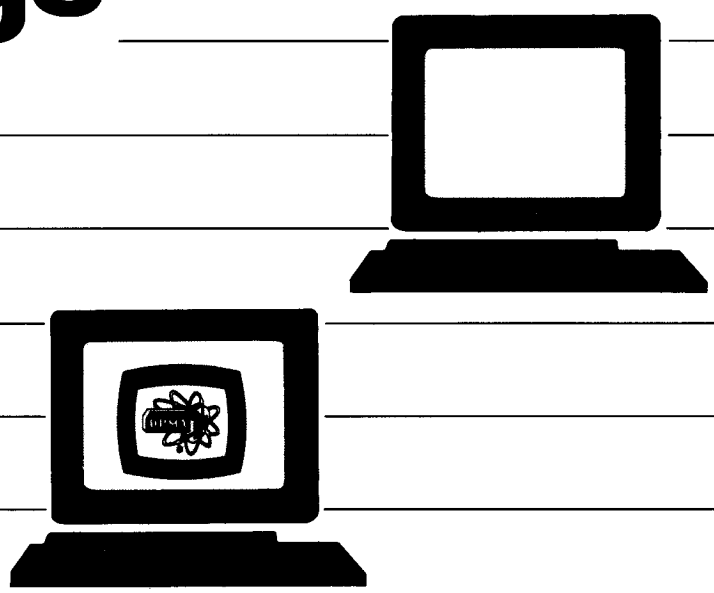
ISECON '87

Proceedings

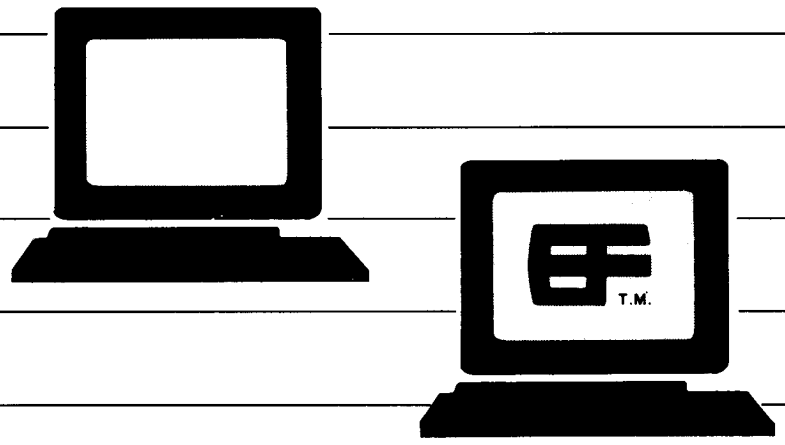
Sixth Annual

Information Systems Education Conference

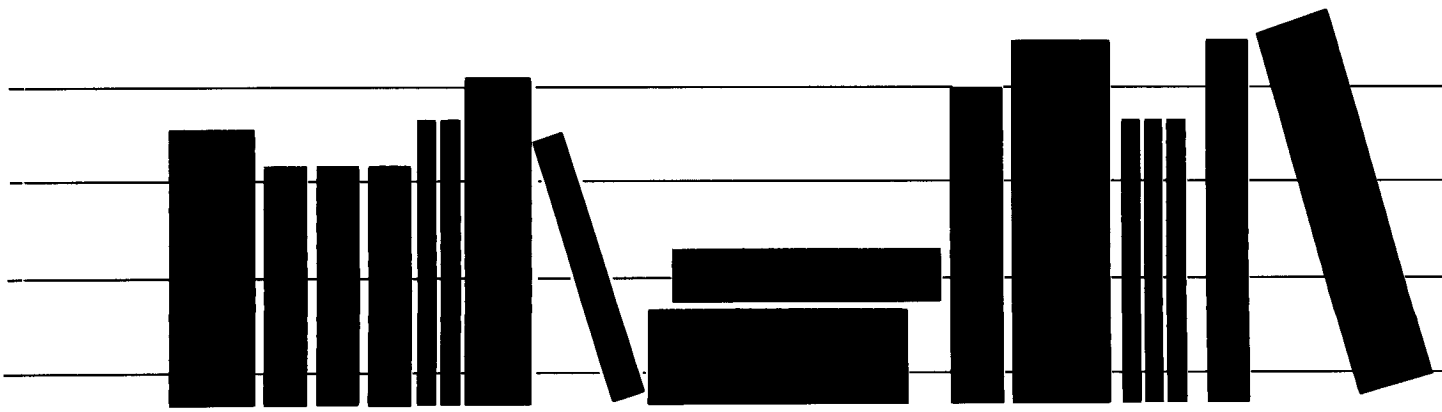


October 31—
November 1, 1987

SAN FRANCISCO,
CALIFORNIA



SPONSORED BY:
Data Processing Management Association
Education Foundation



ISECON '87

Proceedings

Sixth Annual

Information Systems Education Conference

SPONSORED BY:
Data Processing Management Association
Education Foundation

A total of 141 papers and an additional 19 proposals for panels and tutorials were submitted to ISECON '87. Of these, 72 papers (about 50 percent) and 4 proposals for panels and tutorials were accepted (about 24 percent). Each paper was reviewed by at least two referees who submitted individual paper reviews, detailed comments, and rankings of the papers reviewed.

WELCOME TO ISECON '87

Education is the key to our collective national productivity, our competitiveness as a world economy and our quality of life.

As professional educators, you have the most important responsibility of preparing the future systems professionals to meet this challenging and competitive future!

Looking into the future some ten years, we will likely find that many of the skills now taught will continue to be required. Problem solving skills, technical adaptability skills and administrative/leadership skills will continue to be important ingredients in the formula needed for information systems. And, of course, communication, interpersonal and human relations skills will become increasingly important as we envision information systems changing more rapidly and frequently to meet competitive pressures.

Our ISECON is intended to help you become better prepared in your role as educator. At our conference, we hope you discover new ideas, new educational delivery systems and a mutuality of "self-help" subjects in interacting with your peer group.

Welcome to ISECON '87. Let our conference begin.

Warm regards,

Edward A. Otting, President
Education Foundation
Data Processing Management Association

ISECON '87

EXECUTIVE COMMITTEE

CONFERENCE CHAIRPERSON

KATHY BRITTAIN WHITE

Associate Professor of IS
University of North Carolina-Greensboro

PROGRAM CHAIRPERSON

MARY SUMNER

Associate Professor of MIS
Southern Illinois University at Edwardsville

MARKETING CHAIRPERSON

RONALD J. KIZIOR

Assistant Professor of Management Science
Loyola University of Chicago

FACILITIES & LOCAL ARRANGEMENTS CHAIRPERSON

RALEIGH WILSON

Director of Product Development
Mitchell Publishing , Inc.

PLACEMENT CHAIRPERSON

GEORGE FOWLER

Associate Professor of MIS
Texas A&M University

INDUSTRY LIAISON CHAIRPERSON

R. THOMAS BIGELOW

Vice President
Meridian Mutual Insurance

EX-OFFICIO

HERBERT REBHUN

Professor & Coordinator of
Computer Information Systems Program
University of Houston-Downtown

ISECON '87 REVIEWERS

Thomas I. M. Ho
Intelnet Commission

Pete Aleman
Washington University

Judith Solano
University of North Florida

Jack Leitner
University of North Florida

George Jacobson
California State University—LA

Carol Chrisman
Illinois State University

LaVon Green
Purdue University—Calumet

James Courtney
Texas A&M University

Ilene Deutsch
Pace University

Jeff Whitten
Purdue University

Lonnie Bentley
Purdue University

George Fowler
Texas A&M University

Al Lorents
Northern Arizona University

Richard Discenza
University of Colorado

Richard Leifer
Rensselaer Polytechnic Institute

Robert Zant
North Texas State University

Robert Rouse
Washington University

Doris Duncan
California State University at Hayward

Kathy White
University of North Carolina
—Greensboro

James Senn
Georgia State University

Ronald Kizior
Loyola University of Chicago

Herb Rebhun
University of Houston—Downtown

John Cordani
Baruch College

E. Ray Smith
University of Tennessee

Cary T. Hughes
North Texas State University

Richard Vetter
North Texas State University

David Russell
Western New England College

Phil Enns
St. Louis University

Janet M. Cook
Illinois State University

Wita Wojtkowski
Boise State University

Jennifer Wagner
Roosevelt University—Robin Campus

Lloyd Weaver
Purdue University

Curt Hartog
Washington University

Tom Browdy
Washington University

Vicki Sauter
University of Missouri—St. Louis

Donald Chand
Bentley College

James King, Jr.
Baylor University

Jon D. Clark
North Texas State University

Jane Mackay
Texas Christian University

Irv Englander
Bentley College

Stuart Varden
Pace University

Al Lederer
University of Pittsburgh

Mary R. Lind
University of North Carolina

TABLE OF CONTENTS

INNOVATIVE TEACHING METHODS II - CIS ELECTIVES

Software Engineering for Managers . An MBA Level Course

John A. Lehman
Associate Professor of MIS
University of Alaska

J. David Naumann
Associate Professor of MIS
University of Minnesota

ABSTRACT

This paper describes a course in software design which is aimed at MBA students who want to become MIS managers. The course is taught in a single quarter, and is language independent. It combines the Yourdon approach with data structure based approaches, using both lecture and case studies. A modified version of the course has been offered at the AACSB MIS Faculty Development Institute since 1983.

INTRODUCTION

This paper is a description of the Software Design course in the MIS program at the School of Management, University of Minnesota. The course is of interest to others because of the audience, and the course's dissemination.

The audience for the course consists of MBA students rather than engineering or computer science students. Their career goals involve managing programming projects and data processing within commercial organizations. Many of them have non-technical backgrounds and limited exposure to computer technology and programming.

An additional unique aspect of this class involves its dissemination to faculty around the United States. MIS faces a severe faculty shortage, and so the American Association of Collegiate Schools of Business (AACSB - the accrediting agency) has sponsored a Faculty Development Institute since 1983. At this Institute, faculty from business schools throughout the country who are not computer specialists are trained to teach basic MIS courses. The first author has taught this course to around 250 such faculty since the inception of the Institute.

The course divides the programming lifecycle into architectural design, module design, and coding. It uses the Yourdon vocabulary throughout, but also introduce students to other techniques of program design, such as data-structure based, functional decomposition, and object oriented design. The course structure has been shaped by four major constraints:

(1) Many of the students do not have a good background in basic hardware and software technology, but are drawn to the program because of career opportunities in MIS.

(2) Many of the students do not have organizational experience with either computers or with end-users.

(3) The computer systems which the students will use on the job are not the same as the systems used by the University; moreover, the

environment is likely to continue changing at a rapid rate.

(4) We have only ten weeks to present the material, during which some of the students will also have to learn their first programming language.

Our responses to these constraints have been:

(1) Case assignments combined with lecture presentation and extensive reading.

(2) Self-taught programming languages.

(3) Extensive reading of programs

(4) Language-free curriculum which teaches how to apply programming languages in general on computers in general.

CONSTRAINTS

LACK OF TECHNICAL BACKGROUND

MBA programs are aimed at general management, and by design do not require any particular major or sequence of undergraduate courses. The only computer related courses which all MBA students in our program have are an introduction to the University's timeshared computer system, to PCs and a very general introduction to the place of information systems in the organization.

Complicating this issue is the fact that the University of Minnesota has a very large evening MBA program. These students are much more experienced, and the majority of MIS majors currently work in MIS. Exactly the same program is offered in the evening as during the day, due to accreditation requirements. As a result, the same class which works with relatively naive day students must also work with experienced evening students.

LACK OF ORGANIZATIONAL EXPERIENCE.

Many of the students, especially in the day program, are fresh from undergraduate programs and do not have work experience. Since many of the problems with software engineering are people problems rather than technical problems, we have to ensure that the students develop an appreciation of the human side of computer systems.

DIFFERENT SYSTEMS

The University of Minnesota uses Control Data computers, which are designed for numerical processing in FORTRAN. The University of Alaska uses IBM PCs and Unisys machines; most of our students will work with IBM mainframes when they leave school. In addition, we are concerned that our students be able to adapt to changing conditions, since change seems to be the one constant in the computer industry. Thus, we have not chosen the traditional business school route of training students to be immediately productive as programmers on their first job (which means COBOL on IBM mainframes under MVS). Rather, we have taught them to solve programming problems in general.

LIMITED TIME

The University of Minnesota is on a quarter system, which means only ten weeks of classes; at the University of Alaska we can devote only part of a semester to this material. The major impacts in teaching software design are that there is no time for an extensive project, and that those students who are learning their first programming language (or who are learning COBOL for the first time) have to overlap their language learning with class, since there is no time to waste on language syntax.

RESPONSES TO CONSTRAINTS

PRESENTATION OF MATERIAL

Case assignments are combined with lecture presentation and extensive reading. The reason for the case assignments (where the signal to noise ratio is fairly low) is that they give students an opportunity to come to grips with the ambiguity of real world problems. The cases are designed to be similar to assignments which a beginning programmer or designer would face during the first year of work. In order to keep them realistic, the case studies are drawn from actual programming projects, and are reviewed by practicing applications programmers (not managers).

Unfortunately, there is no good text for the course. As a result, we are forced to rely more heavily on lectures than we would like. Classes are thus about two thirds lecture and one third discussion.

SELF-TAUGHT PROGRAMMING LANGUAGES

The ten weeks available does not leave enough time to teach programming language syntax. Examples and assignments are given in a combination of COBOL and various fourth generation languages. Students are told that anyone intelligent enough to be in an MBA program should be able to teach him- or herself COBOL without help from an instructor; few of the students have been self-effacing enough to dispute that contention. One week in the middle of the class is spent discussing how COBOL fits into the conceptual structure for language and system design that we have discussed.

EXTENSIVE READING OF PROGRAMS

One way in which students teach themselves COBOL is through extensive reading and documentation of existing programs. As we present the different forms of notation, students apply them by documenting programs.

LANGUAGE-FREE CURRICULUM

As mentioned above, we use COBOL for many of our examples, but the basic content of the course is language-free. The first author has taught essentially the same course to engineers using assembler, FORTRAN, and C simply by changing the examples.

TOPICS COVERED

The major topics covered in the course are: Introduction, Design, Tools and Notation, Data structures, Control Structures, Measures of quality, Program derivation strategies, Languages, Testing, Implementation, Management, and VHLL.

The Introduction discusses why programming projects tend to be late, over budget, and not what the user expected. This section includes discussion of hardware and software trends, difficulties with coordination and communication as program size increases, and the nature of errors. It includes a good sample of horror stories drawn both from the literature and from personal experience.

The Design lecture includes limitations of the human information processing system, and general principles of design as drawn from rhetoric (1), architecture (2), and engineering.

Tools and Notation introduces the students to the use of structure charts, data dictionaries, data structure diagrams, and data flow diagrams. They apply these tools to documenting one or more COBOL programs

Data structures discusses the hierarchy of data structures from the machine level to the user level concentrating on the idea of data abstraction, with underlying details being hidden as one goes up each level. The lecture also discusses internal and external data

structures, and file organizations and access methods.

Control Structures introduces students to pseudocode, and to modern control structures. Since COBOL has a rather indirect implementation of modern control structures, the students practice extensively with pseudocode before going on to use pseudocode to document the control structures of a COBOL program.

Measures of quality include coupling, cohesion, correspondence, and the various heuristics suggested by Constantine and others such as avoiding decision splitting. The only unique aspect of this lecture is that it divides coupling into design level and implementation level coupling.

Program derivation strategies are the different approaches to deriving a structure chart from a system specification. This lecture includes functional decomposition, transform based design (data flow), data structure based designs, object oriented design, and transaction based design. It discusses the situations under which each is appropriate.

Languages are discussed in two sections. By this point in the class, students have a good grasp of the elements of all programming languages: data structures, control structures, and operations. COBOL is first described using this conceptual framework. An additional lecture compares and contrasts most of the common third generation languages (plus a few fourth generation languages) using the same framework.

The testing and debugging lecture is drawn from Myers (3). It focuses on management of the testing and debugging process and on related organizational issues.

Implementation is divided into two parts: assembling tested modules into a working program, and optimization and efficiency. Optimization and efficiency are treated very briefly.

The lecture on management introduces students to problems of staffing and managing projects. It discusses proposals such as Brook's surgical team approach (4), as well as more conventional approaches. All of these are discussed in the context of how real organizations work, including political problems.

The lecture on Very High Level Languages (VHLL) discusses how such languages fit into the same conceptual frameworks as programming in general. It also discusses areas where the Yourdon approach and Jackson System Design require modification when fourth generation languages are used.

SUMMARY

The course described in this paper has been taught at the University of Minnesota for over six years. It has also been taught for five summers at the AACSB Faculty Development Institute, and within several major corporations. Additional details are available from the first author.

REFERENCES

1. Lehman, J. A., "Program Design and Rhetoric," IEEE Software 3,3 (May, 1976) pp. 71-73.

2. Alexander, C., Notes on the Synthesis of Form, Harvard University Press, 1964.

3. Myers, Glenford J, The Art of Software Testing, John Wiley, New York, 1979.

4. Brooks, F., The Mythical Man Month, Addison-Wesley, 1975.

THE MBA/MIS PROGRAM

OBJECTIVES, CONSTRAINTS, DIRECTIONS

ALDEN C. LORENTS
COLLEGE OF BUSINESS 15066
NORTHERN ARIZONA UNIVERSITY
FLAGSTAFF AZ 86011

602-523-3510

ABSTRACT

MIS as an emphasis in an MBA program has proven to be a successful option for many graduate students. The combination of the MBA courses along with some technical preparation in information systems can be a good compliment to meet the objectives of many employers. Setting up and maintaining a good program has been difficult due to contraints relative to program objectives, number of hours in the program, AACSB considerations, and the philosophy of many faculty toward any emphasis in the MBA program. This paper looks at some of these objectives and constraints using one model as an example.

INTRODUCTION

The MIS emphasis has become quite popular in a number of MBA programs around the country. Students with some MIS background appear to have a competitive edge in the job market over regular MBAs. The design of these programs can vary considerably in terms of their objectives and orientation. Also, there tends to be some constraints in using the MBA program for an MIS emphasis.

The purpose of this paper is to examine some of the objectives, orientation, and constraints related to the MBA/MIS program, and look at some alternative curriculum directions. The program at Northern Arizona University will be used to illustrate an example of one model, and how this model has been recently changed to meet expected demands.

PROGRAM OBJECTIVES:

The objectives vary depending on whether the emphasis is on management and decision making, information systems management, information systems development, or some combination of these. A background in some mix of the above allows a graduate to move into a number of different positions depending on their expertise and interest.

1. Entry level positions as programmer/analysts.
2. Information systems analysts, user based.
3. Information center support.
4. Marketing and marketing support.
5. Management consulting organizations (analysis and development).
6. Positions in the functional areas of the organization where there is a heavy dependence on using information systems.

The program at Northern Arizona University attempts to serve two general objectives.

1. Provide a good background for individuals who want to enter the MIS field as programmer/analysts, information system analysts and system developers. These individuals have a fair amount of technical preparation at the language, system design, and database levels.

2. Provide individuals who plan to enter industry in positions other than those related directly to MIS with enough background to understand and manage their own destiny in using information systems. These individuals may not have as much technical preparation.

PROGRAM CONSTRAINTS:

The MBA program in general has provided a good framework for the MIS emphasis because of the necessity for the information systems specialists to understand the environment they are working in. However, unless the program for the MIS emphasis is expanded, some of the regular MBA program has to be abbreviated to make room for the MIS content. This can be done quite well and still stay within AACSB standards. Some faculty and graduate committees have problems accepting an emphasis in the MBA program, because they feel its no longer an MBA. Some courses become sacred to the MBA program even though the student has had the common body of knowledge in their undergraduate preparation. Another constraint is related to the program layout over semesters. A lot of programs are designed around a one year time frame. It can be very difficult to fit the courses into this time frame with much dependency. The use of summer sessions tend to introduce problems that are not around during the regular semesters.

Some programs are rather small, and the offerings are limited and lock step. For example if it is required of every MBA to take a certain group of courses during one of the semesters, this could rule out the ability to take an MIS course during that semester. If the program is a one year program, this would mean that the MIS content would be limited to the summer and one semester.

PROGRAM STRUCTURE AND IMPLEMENTATION

The program at Northern Arizona University was implemented in 1981. It was patterned somewhat parallel to the undergraduate program to ease the demand on limited faculty resources at the

time. The new MIS emphasis was phased in using the existing MBA program.

The MBA core required the following courses:

Management Information System
Management Theory and Analysis
Managerial Economics
Seminar in Marketing
Quantitative Analysis Probabilistic
Quantitative Analysis Deterministic
Advanced Managerial Accounting
Financial Analysis for Business Decisions
Integrating Seminar

The regular MBA program required the above 27 hours and 6 hours of electives for a total of 33 semester hours. It was set up as a one year program starting in the Fall semester and ending with the second Summer session. It was a two year program for students who did not have an undergraduate degree in business.

The MIS emphasis was implemented as a nine to fifteen hour requirement. The program had the following courses.

Information Systems Development I AND II
Data Base Management Systems
Topics in Information Systems
Internship or Research in Information Systems

The program was originally phased in with a nine hour requirement and a prerequisite of COBOL programming. The MIS students did not take the core Management Information Systems course. Through advising, students would not take two other MBA courses when we implemented the other two MIS courses.

Originally the content of the first three courses was as follows:

Info Systems Development I
Systems development using advanced COBOL methods.

Info Systems Development II
Systems analysis and design methods

Data Base Management Systems
Design and implementation of hierarchical and network type databases.

We have since upgraded the first two courses to exclude COBOL and use relational database and fourth generation languages as the vehicle to teach the development courses. The students are encouraged to have advanced COBOL before entering the program if they plan to enter the field as a programmer/analyst. COBOL is still a part of the database course because we are using IDMS under MVS/TSO.

The first development course introduces a lot of the beginning concepts of structured design methodologies and the design of relational databases. The second course builds on these concepts and covers analysis, implementation and other development related topics. Both courses are heavily project oriented using RBASE SYSTEM V to implement the projects. We are currently reviewing ORACLE and FOCUS.

The topics course is used to give the student a broader perspective of the MIS field. This course gets the students more into the management of

information systems and the directions of MIS. Other topics such as decision support, expert systems, and communications are also covered. The last course is designed to give the student some in-depth exposure to a specific area either through an internship or through independent projects.

The students who are planning on going into MIS take the full 15 hours of MIS courses. Students who do not plan to go into MIS, will take the first two courses and the topics course. The topics course is open as an elective to the regular MBA students after they have taken the Management Information Systems course.

The program is currently being reviewed to look again at the interface with the MBA core and the phasing of courses over the program. There are some thoughts about making it a 36 hour or 39 hour program over three semesters. The summer would be used for internship or research.

PROGRAM SUCCESS

The program has remained small, with a high percentage of the graduates getting jobs in firms with high recruiting standards. A number of the graduates are at Sandia Labs. Other firms hiring have been Lawrence Livermore Labs, IBM, Hewlett Packard, Price Waterhouse, and some of the utilities. The program is very popular with the foreign students.

SELECTED REFERENCES

Cohen, Eli B., and Schultz, David I., "What MBA's Need to Know About MIS 'Computer Literacy' for Managers", Interface, 7:34-39, Winter 85-86.

Duncan, Doris G., "DPMA Graduate Model Curriculum in Management Information Systems - A Preliminary Report", Proceedings, ISECON 85, pp 8-10.

MERGING I.S. CURRICULA WITH BUSINESS CORE SUBJECTS: AN INTEGRATIVE, OPERATIONAL LEVEL CASE PROJECT FOR INFORMATION SYSTEMS MAJORS

R. Brent Gallupe
School of Business, Queen's University
Kingston, Ontario, Canada, K7L 3N6

Paulette L. Shonoski, Ronald A. Nelson & Robert A. Cameron
School of Business Administration, Lakehead University
Thunder Bay, Ontario, Canada, P7B 5E1

ABSTRACT

The study of information systems has typically revolved around courses which concentrate on relatively narrow topic areas (such as decision support systems and data base, etc.) that only indirectly relate to concepts and methods from other subject disciplines (such as production or marketing). Any integration of subject matter is usually left to final year courses such as business policy which typically examines broad strategic issues. A major gap exists in the integration of subject matter at the operational level. Many problems and issues that arise at the operational level are "cross boundary" problems that affect not only MIS policies and procedures but organizational behavior, production, accounting and finance issues as well. This paper reports the experience of one School in its attempt to give 3rd year level business undergraduates an integrative, operational level case that asks them to examine not only information system problems but also production and organizational behavior problems. The results of this experience and the problems encountered are described.

INTRODUCTION

The study of information systems has typically revolved around courses which concentrate on relatively narrow topic areas (such as systems design; decision support systems, etc.) that incorporate few concepts and ideas from other subject disciplines (such as organization behavior, marketing, etc). Any integration of subject matter is usually left to final year courses such as business policy, a course that examines issues on a broad strategic level. A major gap exists in the integration of subject matter at the operational level. Many problems and issues that arise at the operational level are "cross boundary" problems that affect not only MIS but organizational behavior, production, accounting and finance, etc.

Model curriculum for the study of information systems (1) (2) have concentrated on recommending courses in the topics that information systems specialists will need a detailed knowledge of. These curriculums also recommend that the student take a basic set of business subjects such as accounting, marketing, etc., so that they will be able to communicate with individuals in other areas of the organization. Neither the DPMA model curriculums nor the ACM model curriculum have provisions for a means of integrating the information systems knowledge with the applied concepts from

the other discipline areas, particularly at the operational level. Figure 1 shows a graphical representation of this lack of integration.

Entry level positions for information systems graduates are changing. No longer is the graduate limited to a junior programmer position. Information centre consultants, data base analysts, user department liaison positions etc., are now possibilities that the information systems graduate can consider. These jobs require a much broader view of the impact of information systems on organizations. Indeed, as the individuals progress in their careers, they will be increasingly affected by issues and problems both from within information systems and from other departments. By giving information systems students a chance to study a case that examines not just information systems issues but, at the same time, organizational behavior and production issues as well; the instructor is broadening the perspective of the student. The student will be better able to appreciate that what is done in the information systems department affects other areas and vice versa.

This paper reports on the experience of one business school in its attempt to give 3rd year level (penultimate) business undergraduates an integrative, operational level case project that asks them to examine not only information system problems but production and organization behavior problems as well. The paper begins by giving a brief overview of the use of cases in information systems courses. Second, the approach used to give students the integrative case experience is described. Finally, an evaluation using this approach and future plans are discussed.

INTEGRATED CASES IN THE STUDY OF INFORMATION SYSTEMS

Cases are used extensively in teaching information systems concepts and methods. Systems analysis and design courses use cases that describe an information system to be analyzed and redesigned. MIS policy courses use cases that require the student to examine planning and control issues regarding information systems in organizations. These cases focus on issues particular to the study of information systems and only indirectly on problems that might affect other areas. For example, a case might require the student to analyze and design a sales analysis system. The student will focus on information system methods and techniques and only consider marketing concepts and issues if needed. By the same token, the information systems instructor will also concentrate on the information concepts to the virtual exclusion of issues that the case may raise for other discipline areas.

A search of the relevant literature revealed that very little had been written about the use of cases in information systems courses and nothing could be found on the topic of integrated case projects in the study of information systems. The reason for the lack of literature is that the use of integrative cases as a teaching method is quite difficult and usually involves two or more faculty members. A team approach is required that entails extensive co-ordination of courses.

IMPLEMENTING THE INTEGRATED CASE PROJECT

The planning for the implementation of the integrated case project occurred over a period of 4 months in the Spring and Summer. In the early stages of the planning, the objectives for the project were established:

(1) To instill in students appropriate attitudes toward integrative thinking (i.e. focusing on the total organization in addition to functional specialties),
(2) To focus this integrative learning activity at the operational or tactical level of the organization, specifically Information Systems, Production and Organizational Behavior,

THE LACK OF INTEGRATION BETWEEN I.S. AND OTHER BUSINESS SUBJECTS

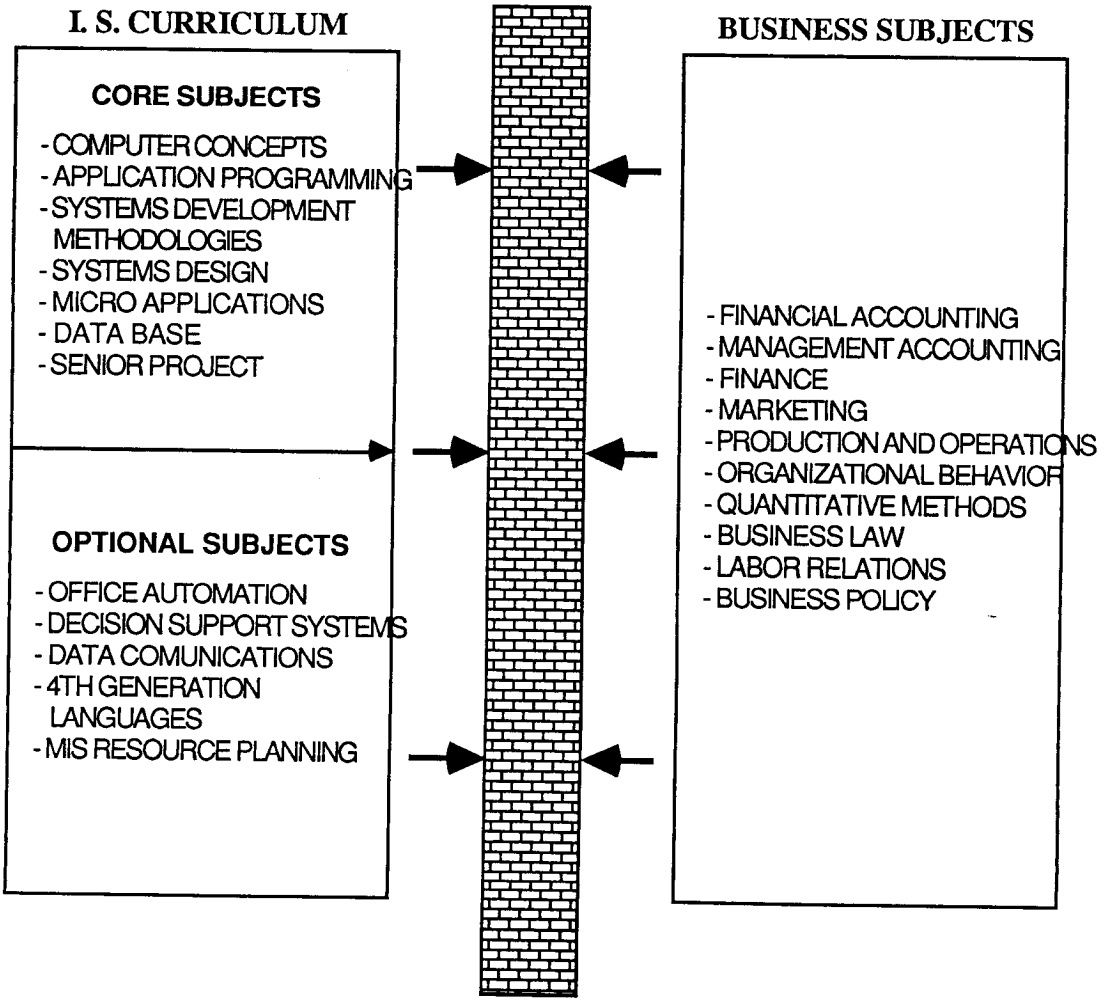


Figure 1: THE LACK OF INTEGRATION BETWEEN I.S. AND OTHER BUSINESS SUBJECTS

(3) To create an organizational simulation that would require students to participate in group work experiences, to complete a formal managerial report and to present, orally, a formal report to a team of senior managers,
(4) To structure a learning situation that would allow the instructors to team teach, and thus role model, the type of behavior we were seeking in our students.

Based on these objectives we developed a schedule of activities to be completed for the implementation of the projects. Figure 2 shows the sequences of activities that were carried out. As Figure 2 indicates, the project required substantial effort on the part of the students, faculty and the simulated team of senior managers. The major activities included the formation of groups, the practice cases, submission of group reports for the final case, the student presentations to the team of managers, and the evaluation of presentation and group reports.

The actual case that was used was a modified version of the SOF-OPTICS Inc., case found in "Cases in Operations Management" published by Irwin. This modified case was felt to be rich in issues of an integrative nature. For example, a decision to use certain information technology had impacts on motivation of employees and types of production processes. Similarly, the choice of production processes and the reporting of the results of those processes had impacts on information systems and individual behavior.
There were three groups of participants in the project. The students were individuals registered in third year of a four-year business program and taking all three of the following courses: (1) Accounting Information Systems, (2) Production, (3) Organizational Behavior. Production is a one-semester course while Accounting Information Systems and Organizational Behavior are two-semester courses. All three courses are required courses in the program. Participation in the project was given a weighting of 10% of the final mark in each course. A total of 48 students took part. Students were divided into groups of four.

The second group of participants were the actual managers that made up the simulated management team. In order to make the simulation as powerful as possible only individuals that met the following criteria were considered:

(1) Senior managers in their respective organizations,
(2) Managers with diverse backgrounds to represent different organizational sub-units,
(3) Managers who showed enthusiasm for becoming involved in an educational experience such as this.

It is interesting to note that no difficulty was encountered in getting highly qualified and enthusiastic managers to participate. It was decided that a team of three managers would be appropriate to provide students with clear and concise feedback.

The third group of participants were the faculty of the three courses. Each faculty member was assigned individual tasks necessary to complete the project. In addition, the faculty members worked as a team in the training sessions (all three faculty members in class at the same time, leading the discussion) and in the evaluation of the group reports and the oral presentations (each faculty had an equal say in the marks assigned).

EVALUATION OF THE INTEGRATED CASE PROJECT

The evaluation of the integrated case project comes from verbal and written comments made by all three groups of participants. Since this was the first time an integrative case project was used, evaluation feedback was open-ended to get positive and negative comments and suggestions for future changes to the project.

SCHEDULE OF ACTIVITIES FOR THE INTEGRATED CASE PROJECT

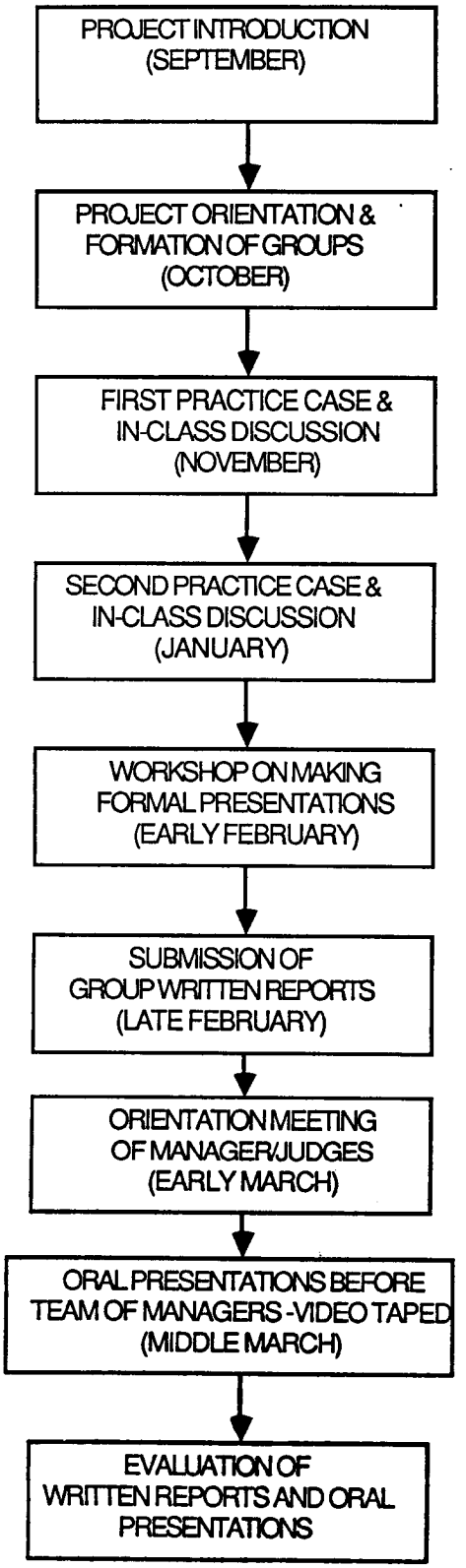


Figure 2: SCHEDULE OF ACTIVITIES FOR THE INTEGRATED CASE PROJECT

STUDENTS

Most of the students perceived the project as a useful learning experience. Most of the students appreciated the value of examining the case from a integrative viewpoint. The typical comment was the following, "I had no idea that the course of action we proposed would have so many implications for other areas. We would not have been aware of this if the case had been given in one course only." The students valued the opportunity to interact with professional managers as well as the chance to make a formal oral presentation.

On the negative side, the students felt that the practice cases did not adequately prepare them for the formal case. Most students thought the workload was excessive for the weighting of the grades in the three courses. In addition, a minority of students felt that they were inadequately prepared to make formal oral presentations and that more instruction and training should be given in public speaking.

MANAGERS

The feedback from the managers was extremely positive. They found the experience exciting and interesting. One manager commented that there should be more use of this kind of integrative teaching approach in business schools. He felt that it gave the student a much better feel for actual business issues and problems. All managers expressed a willingness to participate in the project when it was held again.

FACULTY

Feedback from the three faculty members reflected both positive and negative reactions. On the positive side, they all felt it had been a valuable learning experience for the students. They all gained a new appreciation for the content and teaching methods used in the other two courses. They also felt that the integrated case experience increased their rapport with students as well as the management community outside the school.

On the negative side, all three faculty agreed that the effort in setting up and implementing the project was substantial. It was far greater than they anticipated. The logistics of all three instructors in one class, the conducting of the oral presentations, etc., placed significant extra burdens of time on the faculty. It was felt, however, that much of the effort was a result of first time implementation and that the logistical effort would be reduced the next time. Another negative factor was the cases used. Although both practice and formal cases were selected because they offered an integration of issues, it was felt that better cases could be used.

SUMMARY

The integrative case project for information systems majors has the potential to be an important component in the CIS Curriculum. It will be the major link between information system courses and core business subjects. Faculty must be willing to give the extra effort required to make this learning experience a valuable one for all concerned.

REFERENCES

1. DPMA, "CIS'86, The DPMA Model Curriculum for Undergraduate CIS Education", Park Ridge, Illinois, 1986.
2. ACM, "ACM Curricula Recommendations for Information Systems, Volume II", New York, 1983.

IMPLICATIONS FOR CURRICULUM DEVELOPMENT: RESEARCH TO VALIDATE AN APTITUDE TEST FOR COMPUTER PROGRAMMING

by

George Jacobson, George Burstein, and Ron Cowart
California State University, Los Angeles
5151 State University Drive
Los Angeles, CA 90032
(213)224-2961

ABSTRACT

Scores on a general aptitude test were found to be highly predictive of levels of learning in a second course of instruction in computer programming at a university. Implications of such a correlation on curriculum design and career counseling are discussed. Directions for further research are indicated.

INTRODUCTION

The field of education for information systems is exploding with new technologies, multidisciplinary applications, and a wide variety of students from different backgrounds. Some guidance in differentiating levels of instructional intensity, and career choice seems to be indicated. The course involved in the first phase of this investigation is a second computer programming course for the students at this university. As a prerequisite, these students have been exposed to a combination of CIS/86-1 and CIS/86-2 of the model curriculum of the DMPA. This second course (CIS/86-3) deals with program design and development. Students are required to "...apply a structured multi-phase program development process that features a series of steps involving understanding of a problem, formal problem definition, graphic design methodologies... and program specification through pseudocoding...." At this university COBOL is the language employed for CIS/86-3. This is the first course in which formal programming instruction is undertaken. It is therefore important to offer some form of career guidance to students enrolled in this course. This is a critical career choice point for students who wish to commit themselves as majors in computer programming for business applications. The validation of instruments to assist students in making career choices by furnishing them with predictive cues about their learning ability as computer programmers is a responsibility of the educational or training organization.

For these reasons a controlled study of the differential level of learning of COBOL associated with cutoff scores on a published aptitude test for computer programming was undertaken. The aptitude test used was W-APT, Programming Aptitude Test, 1983 - Wolfe-Spence. Since students may make a series of career decisions, we were concerned with investigating the varied alternatives for impacting individual differences in

learning potential uncovered by W-APT. Among the alternatives to be considered were lengthened courses, enriched courses, tracking individual students, and supportive faculty interventions. This research commenced in the Fall of 1986. Tentative findings and plans for continued research are reported in this paper.

METHODOLOGY

Seventy-eight students in Introduction to COBOL (The CIS/86-3 equivalent) were randomly assigned in three class sections. In the first session of the course, all students were required to complete the W-APT, Programming Aptitude Test, 1983). This test is of a one hour duration. Neither the instructors nor the students were provided with the test results. Instructors were not given the test manual. The course was conducted in the standard manner by all the instructors. Learning was measured in the fifth, eighth, and eleventh weeks. The course was ten weeks long. While the tests were all classroom administered, they were identical in all the course sections and standard scoring keys were employed. Test items were "fill ins", "sentence completion", and "exercises". Answer keys were objective and standard. Students were also evaluated on the quality of completed "homework". This "homework" was comprised of COBOL programs answering certain specifications. The COBOL program component of the student evaluation was subject to external prompting and copying because an underground in previously submitted "homework" had developed. Weighting of the student grades in this course were as follows:

5th week quiz - 20%
8th week quiz - 20%
11th week final exam. - 20%
"Homework" submitted - 40%

Statistical measures of significance (a Pooled Variance Estimate) and Spearman's Rank Correlation (t tests, and Spearman's Correlation Coefficients - Rho) were utilized to analyze the differences in learning for students who scored above and below the cutoff score (67) on W-APT. Levels of learning were defined as grades achieved on the quizzes, the final examination, and the "homework" submitted.

FINDINGS

Because the "homework" programs have been used for the past four years, no correlation was expected between the scores on the aptitude test and the scores on the "homework" since students could copy this work or receive assistance from others in completing it. As expected, there was no statistical significance or meaningful correlation between students passing or failing the aptitude test and "homework" scores (Significant at .204; Rho=.0951). A single night class did however follow the trend found in the daytime classes. This class did show a significant correlation between scores on the W-APT and the course quizzes and final examination (Significant beyond .001; Rho=.6690).

Figures One, Two, Three, and Four show the levels of significance and correlations for the other measures in this study.

DISCUSSION

W-APT appears to be predictive in a university situation, although it is not yet fully validated. This study may serve to assist in such validation. Further replication is being planned. It may then be considered as an aid in selecting, tracking or counseling students in a university information systems setting, or for tracking trainees in a computer programming section of a work organization. The results of this study were confounded by the drop out factor among students who scored less than 67 on W-APT. Correlations would have been greater if all students who started the course had completed all the quizzes and final examination. This drop out rate is understandable, however,

since students doing poorly tend to leave the course. Differences between those students passing and failing W-APT grew from Quiz 1 through the final examination (see t values in Figures One, Two, and Three and growth of the correlations in Figure Four).

Goldstein (1986) discusses instructional approaches to individual learning differences of the type found in this study. This discussion is related to a desire for educating and training large numbers of people from diffrerent backgrounds or with different learning characteristics (such as now enter the field of information systems) without engaging in lock-step educational or training procedures. Many educators and trainers are coming to realize that designing educational programs for the "average" learner might not be the most efficient approach. What is best for one individual (x) at any given time might not be the best for other individuals at that time, or for (x) at a subsequent time. Findings of individual differences in learning, such as those in this study, should therefore be a basis for assisting as many students and trainees as possible to make career or educational choices regarding the speed with which they will learn additional material, or select a specialization. Other decisions may concern which educational majors they will pursue, which branches of occupations they will select, which methods for advancement they might follow, or which methods they might employ to learn new material.

Four approaches have been adopted to deal with individual differences in learning (Goldstein, 1986). The first, the Fixed Program Approach, does not alter the program but sets fixed learning objectives and requires the student/trainee to remain in the program until he or she has achieved those objectives. This can prove to be exceedingly expensive and may ultimately result in learner frustration, since not all people can learn everything needed in an extended period of time. The second approach is the Adapting Goals Approach. Trainers and educators who believe in "tracking" students/trainees based on anticipation of separate needs or goals (selected by these persons or their instructors) are applying this concept. Tracking frequently fails to offer realistic alternatives among the separate choices made available to students/trainees. Separate methods for learning are rarely designated among the tracks developed. Burstein & Jacobson (1986) and Jacobson & Jackson (1986) have suggested that peer review enlarges the level of learning for students in introductory programming courses. This is illustrative of the importance of differentiating methodologies to accompany tracking decisions. The third approach is the Erasing-Individual-Differences Approach. Efforts are made to reduce individual differences by offering remedial instruction. This is again, extremely costly and difficult to administer without some rationalization of alternate methods to use when providing remediation. Some individual differences may persist despite extensive remediation. The fourth approach is the Altering-Instructional-Methods Approach. This is idealy the procedure of choice. Another name for this approach is the Aptitude-Treatment Interaction. This approach seems to relate most closely with the findings in this study. Its application would call for identifying which group of students/trainees (those of high or low aptitude) profit most from a particular method or methods of instruction. If, as is the case in this study, those of high aptitude profit more from a standard methodology, it would seem that they should be offered an enriched and speeded program which uses such a method. Low aptitude students should be offered basic content with extra attention. More laboratory hours, additional exercises, spaced learning and practice, extended explanations, and structured tutorial types of support, seem to be required for low aptitude students/trainees. Such additional supportive methodology should be paired with additional academic or occupational

counseling. We recommend this approach within the constraints of budget and organizational will. Additional research utilizing these concepts will accompany further empirical efforts to validate W-APT.

References

Burstein, G. & G.H. Jacobson (1986). "An Empirical Approach to Studying the Effects of Medial Feedback in an Introductory Programming Course", Proceeding, National meeting of the Association for Human Resource Management and Organizational Behavior (HRMOB), New Orleans, LA, November 19, Vol. 1, pp. 92-96.

Goldstein, I.L. (1986). Training in Organizations, (Second Edition), Monterey, CA: Brooks/Cole Publishing Co., pp. 260-265.

Jacobson, G.H. & D.P. Jackson (1986). "Measurement of Two Teaching Strategies for Strategies for Computer Programming Instruction", Interface: The Computer Education Quarterly, Vol. 8(2), pp. 26-30.

Figure One

Differences in Fifth Week of Learning (Quiz 1) Between Students Passing and Failing

		W-APT	
"N"	W-APT Score	Mean	SD
20	>67	88.2500	12.594
58	<66	72.2414	14.332

t=4.73, significant beyond .001

Figure Two

Differences in Eighth Week of Learning (Quiz 2) Between Students Passing and Failing

		W-APT	
"N"	W-APT Score	MEAN	SD
20	>67	75.1500	14.001
50	<66	55.0517	16.990

t=5.23, significant beyond .001

Figure Three

Differences After Ten Weeks of Learning (Final Examination) Between Students Passing and Failing W-APT

"N"	W-APT Score	MEAN	SD
20	>67	77.3340	15.640
58	<66	52.1953	13.715

t= 5.63, significant beyond .001

Figure Four

Spearman's Rank Correlation: W-APT Scores and Four Measures of Learning Cobol

(N= 78)

Correlation	"Homework"	Quiz 1	Quiz 2	Final Examination
Rho	.0951	.5000	.5525	.6040
Rho Squared	.0090	.2704	.3053	.3648

THE RELATIONSHIP OF ATTITUDES AND PRIOR COMPUTER EXPERIENCE TO PERFORMANCE IN THE INTRODUCTION TO COMPUTER INFORMATION SYSTEMS COURSE

Neil W. Jacobs and Evangeline L. Jacobs
Northern Arizona University

ABSTRACT

Previous research regarding the introduction to computer information systems course has addressed student characteristics such as anxiety; however, research which considers the relationship between student characteristics and performance is particularly sparse. This study addresses two variables which are likely to be associated with performance, i.e., prior computer experience and prior attitudes toward computers. Results indicate a positive relationship between computer experience and performance and significant differences in performance associated with general attitudes toward computers.

INTRODUCTION

Considering the growing popularity of the introductory computer literacy course (CIS/86-1 in the current DPMA model curriculum) there have been surprisingly few studies dealing with student characteristics bearing on effective student performance.

One particular area of study has been students' computer anxiety, i.e., a disproportionate fear of impending interaction with a computer. Howard et al., (4, 5) examined the affects of demographic and psychological variables on computer anxiety. They found anxiety high in one third of the students upon entering the course, and that the proportion with high anxiety dropped to five to eight percent at completion (5). Further, they found computer anxiety to be positively correlated to external locus of control, math anxiety, and trait anxiety, and negatively correlated to societal impact attitude, computer knowledge, computer experience, and class rank. Their earlier study (4) also found that students who owned a computer or terminal were significantly less computer anxious. These studies included no data on student performance.

Some have sought a basis for predicting performance in computer courses. Peterson & Howe (6) sought to predict success in programming. Fowler-Goldfeld (3) developed a model which estimates aptitude based on grade point average, age, ACT score, and number of math courses taken. The primary discriminator in the model was GPA; the other factors were of marginal discriminatory power. Discenza, et al., (2) attempted to validate the model, and found a 71% successful classification rate, e.g., performance was consistent with the model's prediction 71% of the time. While they concluded the model was of some practical value, they recommended that future research focus on identification of variables more appropriate for the specific academic environment.

Support for considering prior computer experience is provided by Yestingsmeier's (7) finding that prior use of a home computer led to higher performance. This result suggests that prior computer experience or expertise might be an appropriate consideration in a model such as the Fowler-Goldfeld model. (Interestingly, Yestingsmeier found no affect associated with having prior introductory computer coursework.)

Colleges can expect a greater proportion of students in the introductory course to have had computer experience, as students' experience increasing exposure to computers in high school and at home (1). It seems logical that prior computer experience would lead to better performance in the course. Yestingsmeier found that previous use of a home computer affected performance, but, what of other types of exposure to computers, such as programming, or the use of common application software packages for wordprocessing or spreadsheet analysis? This study explores this question by developing an index of computer experience and evaluating the relationship of the index to performance in the course.

Another area that remains to be addressed is the relationship of prior attitudes to performance. What is the relationship of attitudes toward computers to performance? Are initial attitudes related to performance? Which specific attitudes relate to performance, e.g., students' general attitudes toward computers, students' attitudes toward learning about computers, and/or students' perceptions about the significance of computers to their future?

The objectives of this study were to examine the relationships of initial attitudes toward computers and prior computer experience to performance while controlling for previously identified significant correlates of performance, namely, general academic ability.

Understanding the relationships between performance and prior conditions such as attitudes toward computers and prior computer experience could aid in designing the course for greater effectiveness. For example, if attitudes are closely related to performance, early sessions of a course could be directed toward getting students to develop positive attitudes. If computer experience has a significant affect, perhaps course activities should vary, e.g., early hands on projects, based on computer experience, or curriculum modifications should allow greater flexibility based on prior experience.

METHODOLOGY

Four sections of an Introduction to Computer Information Systems course were included in this study. The course was taught as three relatively distinct modules, i.e., computer concepts, flowcharting and BASIC, and use of application software packages for wordprocessing, spreadsheet analysis, and database management. Two instructors each taught two sections in the same semester using identical teaching materials, i.e., syllabi, texts, handouts, project assignments, quizzes, and exams. Daily schedules were also the same. No attempt was made to standardize method or style of delivery by the two instructors.

Attitudinal data were collected using multipoint Likert type scales administered at the beginning of the semester. The initial questionnaire also obtained demographic data, as well as self-reports of computer proficiency. GPA scores were obtained from the Registrar. Three questions, common to both questionnaires, addressed students' attitudes toward: (1) computers in general, (2) learning to use computers, and (3) the importance of computers to their future.

To measure prior computer experience a composite index of computer experience was constructed based on students' reported proficiency with microcomputers, wordprocessing, spreadsheets, and programming. Students classified their proficiency with each as "expert--I am quite familiar or very proficient", "extensive use--I can work effectively by myself," "limited use--With some help, I can work well," and "little or no acquaintance." "Expert" responses were assigned a value of 4, "extensive use" a 3, etc., then, the values for all items were summed to give a composite index of computer experience which ranged from 0 to 16.

Performance was measured by overall course grades which were based on ten quizzes and three exams. Eight homework projects were required; successful completion was recorded, and incomplete or erroneous projects were returned, however, no grades were assigned for the projects.

Correlation analysis was used to evaluate the expected positive relationship between computer experience and performance. Analysis of variance was then used to examine for affects of initial attitude on performance, while controlling for grade point average and prior computer experience.

RESULTS

The student population of 183 seemed fairly typical for the computer literacy course in a university. Most were freshman, the average year in school was 1.38. The average age was in the low twenties, and the mean number of high school or college math courses taken was 2.6. Average GPA was 2.51. Most of the students were College of Business majors; however, approximately forty per cent were from other colleges in the university.

PRIOR COMPUTER EXPERIENCE AND PERFORMANCE

The mean score on the index of computer experience was 7.7 with a standard deviation of 3.15. Prior computer experience was positively correlated to performance ($r = .23$, $p = .002$).

ATTITUDES AND PERFORMANCE

Students' general attitude toward computers were positive. Their attitude toward learning about computers was not quite as positive, and they felt that computers were "very important" to their future.

TABLE 1
Descriptive Statistics on
General Attitudes toward Computers and Performance
Note: Means are adjusted for GPA and computer experience

General Attitude	n	Mean	Standard Deviation
Very positive	61	77.9	.96
Somewhat positive	87	76.0	.80
Neither positive nor negative	24	71.8	1.50
Somewhat negative	11	78.6	2.30
Very negative	0		

Results of the one way analysis of variance to examine the affects of initial attitudes on performance (while controlling for grade point average and prior computer experience) indicated a significant affect for general attitude toward computers on performance ($F = 4.19$, d.f. = 3, 177, $p = .007$). Descriptive statistics are shown in Table 1.

The affects of both GPA ($F = 309.74$, d.f. = 1, 177, $p = .0001$) and computer experience ($F = 13.68$, d.f. = 1, 177, $p = 0003$) were significant.

No significant affects were found for attitude toward learning about computers or the perceived importance of computers to students' future.

DISCUSSION

PRIOR COMPUTER EXPERIENCE AND PERFORMANCE

The expected positive correlation with computer experience was found, and the affect of prior computer experience in the analysis of variance was significant. This result is consistent with Yestingsmeier's finding that use of a home computer leads to higher performance. With the index of computer experience, the definition of prior computer experience is extended to include programming, use of common application packages such as wordprocessing and spreadsheet analysis, and other than home use of a microcomputer.

The relatively low value of the correlation coefficient may not be too surprising when one considers the significant relationship known to exist between GPA and performance. Further studies of both the index of computer experience and the relationship between experience and performance may clarify the extent of the relationship.

The result suggests that models, such as the one developed by Fowler-Goldfeld to predict student performance, should include a factor for computer experience. Also, should these preliminary findings be supported by further research, the results suggest that course and curriculum design might well include consideration of an expected positive relationship between prior computer experience and performance.

ATTITUDES AND PERFORMANCE

Results indicate that general attitudes about computers have an affect on performance, however, the nature of the relationship needs further exploration. The descriptive data in Table 1 suggest higher attitudes are associated with higher performance--except for those with a negative attitude whose mean performance was the highest of all attitude groupings. If further study confirms these indications, an early agenda item in the introductory CIS course might well be steps to encourage development of a positive attitude toward computers.

REFERENCES

1. Carey, Regan, & Gall, Meridith. "Patterns of microcomputer use at home and at school by secondary school students," Educational Technology, October, 1986, pp. 29-31.

2. Discenza, Richard., Fowler, George., & Glorfeld, Louis. "An innovative approach to classifying introductory computing students: An opportunity for enhanced resource utilization," Proceedings, Decision Sciences Institute Annual Meeting, Las Vegas, November, 1985, pp. 179-181.

3. Fowler, George C., & Glorfeld, L.W. "Predicting aptitude in introductory computing: A classification model," Association for Education Data Systems, Winter 1981, 14(2), pp. 96-109.

4. Howard, Geoffry S., Lahey, V. Michael, Murphy, Catherine M. & Thomas, Glenn A. "Student profiles and suggestions for improvement in the introductory computer center [course]," Proceedings, Fifth Annual Information Systems Education Conference, Atlanta, Ga., October, 1986, pp. 167-171.

5. Howard, Geoffry S., Thomas, Glenn E., & Murphy, Catherine M. "Computer anxiety considerations for design of introductory computer courses," Proceedings, Decision Sciences Institute Annual Meeting, Honolulu, November, 1986, pp. 630-632.

6. Peterson, C.G., & Howe, T.G. "Predicting academic success in introduction to computers, AEDS Journal, 1979, 12(4), pp. 182-191.

7. Yestingsmeier, Jan. " The effect of prior computing experience on student performance. Interface, Fall 1985, 7(3), pp. 46-49.

Predicting Success in the Introductory COBOL Course

James A. Nelson & Bonnie J. Scranton
College of Business Administration & Economics
New Mexico State University, Las Cruces, NM 88003
Telephone: (505) 646-4901 BITNET: BSA047 @ NMSUVM1

ABSTRACT

Grades in an introductory COBOL course of 230 students were analyzed using a regression model with independent variables of high school grades, college grades, ACT scores, introductory computer science grades, and grades in an algorithmic computations course. The regression model was significant at the .0001 level and accounted for 47% of the variance in COBOL grades. Suggestions are made for further research.

As our society has become more computer dependent, institutions across the country have begun to offer specialized curricula for various computer applications. One can readily find baccalaureate granting colleges and universities that offer degrees in computer science, computer engineering, human-factors engineering, computer aided design and manufacturing, and computer information systems. With such expanded emphasis and opportunities, more college students are choosing to major in computer related disciplines. One such program is the Data Processing Management Association's (DPMA) Computer Information Systems (CIS) curriculum for schools of business incorporating the American Assembly of Collegiate Schools of Business (AACSB) common body of knowledge. This increase in demand, combined with a continuing concern over student retention and limited institutional resources, is causing university personnel to seek valid ways of predicting which students are most likely to succeed as majors in computing related fields.

As early as 1972, Alspaugh (1) did a study to predict performance in a programming course. The independent variables in his study were: scores on the IBM Programmer's Aptitude Test, the Thurstone Temperament Schedule, the SCAT Quantitative and Verbal Subsets, the Watson-Glaser Critical Thinking Appraisal, and a coding system for mathematical background. Alspaugh used 50 students in his study and three measures of programming proficiency: a FORTRAN score, an assembly language score, and a total of the two scores. Using 9 or 10 independent variables selected from a total of 18, he was able to explain from 33 to 40 percent of the variance in programming proficiency. For all the dependent variables, the mathematical background code that attempted to interpret level of high school and college mathematics appeared to be the most important independent variable.

A study by Peterson and Howe (18) found general intelligence score and college GPA to be important predictors as to the grade received in an introductory computer science course. The course covered general information, programming, and computers and society. The researchers built a model using biographical temperament and aptitude data from 113 students enrolled in this course during the fall of 1975. They subsequently used 119 students enrolled the following spring semester to validate the model.

Mazlack (15) concluded in his study that "future programming skill is not predictable" by the IBM Programmer's Aptitude Test. He based this conclusion on results of his study of 63 students enrolled in a FORTRAN programming course. The simple correlation between total PAT score and course grade would only explain about 11 percent of the variation in course grade. Such a small percent validates Mazlack's assertion that the PAT score is not a valid predictor of skill in and of itself.

Several studies reveal that students of high general ability perform well in programming classes. One study of the relationship between ability and

performance in a programming class was carried out by Kurtz (13). He based his study on the Piagetian theory of intellectual development. He used several standard measures of intellectual development to assess student abilities. He then related these scores to student performance on tests and homework in an introductory programming class. Kurtz found that students' intellectual ability did correlate with performance in the course.

A logistical classification model was developed by Fowler and Glorfeld (8) in an attempt to classify students in an introductory programming course. High aptitude students were those receiving an "A" or "B" grade while low aptitude students received "C", "D", or "F" grades. The classification model used college grade point average, number of math courses, SAT math score, and age as independent variables. The model correctly classified 81 percent of the students (122 of 151) and a subsequent validation study using the same model correctly classified 79 percent of the students (41 of 55). The Fowler and Glorfeld results indicated that age was of marginal value while college GPA appeared to be the most important independent variable.

Konvalina, Stephens, and Wileman (12) were able to account for 19 percent of the variation in final exam scores of 165 students enrolled in an introductory computer science course. They concluded in their study that high school mathematics background, overall high school performance, and exposure to high school computer course work were directly related to performance in college level computer science courses. Their conclusion is questionable since their study could only account for about 19 percent of the variance.

In 1984, Campbell and McCabe (5), through discriminant analysis, endeavored to identify the statistical relationship between a student's entrance characteristics and his or her success in the first year of a computer science major. They used high school rank, SAT scores, and high school science and math background as their variables of interest. The results of their statistical study showed that students who persisted in the computer science major, or switched to engineering or another scientific discipline, were significantly different from students who switched to non-scientific majors.

Most recently, Butcher and Muth (3) used first-semester freshmen in a study designed to predict both performance in an introductory computer science course and first-semester college grade point average. High school coursework, GPA, class rank, and ACT scores of 269 freshmen were used as independent variables in the study. Their results showed that 36.6 percent of the variation in course grade could be explained by the relationship of any two of the following three variables: high school GPA, ACT Math, and/or ACT Composite. Most of the variation in exam scores (63.4%) remained unexplained.

Numerous other studies have been done which relate to the psychological aspects of programming. Although this is not an area we can readily measure in college students attempting to choose a major, it is important as it relates to the issue of success as a whole.

One aspect of a psychological analysis of programming is that it requires a kind of abstract thinking sometimes called formal reasoning. This is because most programs are not a single solution to a specific problem, but algorithms that work for a entire class of similar problems. The program must work correctly for a complete range of input values and respond appropriately to a variety of possible circumstances. In Hoc's words (10), the programmer "no longer finds it sufficient to decide what is to be done in a specific situation; he must find a way to make the machine decide what is to be done in any possible situation." Hoc's research showed that learning to program requires internalizing a "mental model" of the computer. Creating a program is specifying how the computer should behave, relying on one's mental model for its functioning and a "device language" for specifying its operation. According to his research, programming design is primarily a planning task. Thus, it demands that people organize their cognitive activity much more carefully (and systematically) than in many other kinds of problem solving.

Coding is implementing a design in a particular programming language. Programming languages are formal languages and have many rules about correct usage. Miller (16) has examined how people describe programming tasks using English. His results show that people use ambiguous natural language to describe procedures, yielding results that are incompatible with the requirements of formal languages. This need for precise expression is, no doubt, an important reason programming languages are difficult to learn. This research lends credibility to the thought that performance in English courses could be as important a predictor to success in programming as performance in mathematics courses.

Along with specific ability, intellect, and experience, gender is frequently hypothesized to co-vary with other factors and therefore to relate to outcomes in computer programming disciplines. Most programming instructors and many research studies

agree that intellectually, females perform as well as males at learning to program (4). However, because of social factors, females may not be as motivated to persist if computer resources are scarce or if the applications presented are not appealing (9). Based on research of this nature it would appear that more females will become successful programmers as interaction between females and computers increase.

It is obvious, from previous research, that many factors contribute to the success of those students who wish to pursue a computer related major. However, not all of those factors can be readily identified in college students. In addition, factors such as sex and ethnicity cannot legally be used when setting admission standards. The literature to date does tend to support the theory that accessible data, such as high school GPA, college GPA, and ACT scores, can be used to predict performance in introductory programming courses. This study hopes to take the research one step further by showing that these factors, in concert with performance in introductory courses, can be used to predict success in the computer information systems major. Two of the courses in this study are similar to CIS courses recommended by the DPMA. The introductory course, CS 110, is modeled after CIS 1, "Introduction to Computer Based Systems" and BCS 217 (COBOL), fits the description of CIS-2 and CIS-3, "Application Program Development I & II".

METHOD

The sample group for this study consisted of 230 students enrolled in the introductory COBOL programming course at New Mexico State University. COBOL is the prerequisite course for any student wishing to major in computer information systems. Students enrolled in this course previously completed an introductory computer course (CIS 1) which uses some PASCAL programming and an algorithmic computations course using PASCAL or Modula 2. They also completed, or demonstrated competency in, intermediate algebra, calculus, finite math, and freshman English composition. The sample consisted of 106 females and 124 males. All but four of the students were either anglo or hispanic, the dominant ethnic groups at our university.

Independent variables included ACT scores for mathematics (ACTMATH), English (ACTENG), social science (ACTSS), natural science (ACTNS), and composite (ACTCO), which were obtained from standardized ACT examination reports. Other variables were the student's high school grade point average (HSGPA), college grade point average (CGPA), and grades in introductory computer courses (CIS 1) and algorithmic computations (ALGCOMP) which were obtained from registrar's records. All grades used in the study were coded numerically using a 4.0 scale.

Multiple regression analysis was done using the Statistical Analysis System (SAS).

RESULTS and DISCUSSION

Table 1 summarizes the Analysis of Variance for the regression model. The variables ACTENG, ACTSS, ACTMATH, and ACTNS were eliminated from the model because of problems of multicollinearity (Maddala, 1977) with the composite ACT score (ACTCO). Pearson correlation coefficients of ACTCO with the other ACT scores ranged from $r = .82$ to $r = .88$, which severely compromises the parameter estimates of the model. Although collinearity does not necessarily interfere with the predictive value of a model, inclusion of ACTSS, ACTNS, ACTENG, and ACTMATH only increased r^2 by .03 from $r^2 = .468$ to $r^2 = .500$, therefore they were eliminated from the analysis.

TABLE 1

ANALYSIS OF VARIANCE

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	7	260.57	37.22	27.90	0.0001
ERROR	222	296.19	1.33		
C TOTAL	229	556.77			
R-SQUARE		0.4680			

Significant parameter estimates included college grades (CGPA), ACT composite scores (ACTCO), and grades in the algorithmic computations course (ALGCOMP). All other variables were not significantly different from zero as parameter estimates as seen in Table 2. Gender, race, CIS 1 scores, and high school G.P.A. did not add to the predictive power of the model.

TABLE 2

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	PROB
INTERCEPT	1	-2.35	0.0001
HSGPA	1	0.02	n.s.
CGPA	1	1.75	0.0001
ACTCO	1	-0.05	0.001
CIS 1	1	0.07	n.s.
ALGCOMP	1	0.15	0.05
GENDER	1	0.01	n.s.
RACE	1	0.07	n.s.

It is interesting to note that the ACT composite score (ACTCO) was a significant parameter but with a negative sign (-0.05). The mean ACT composite score for the sample was ACTCO = 19.84. The negative parameter estimate was due to the fact that many students with relatively low ACT scores were successful in the COBOL course. It appears that students with low ACT scores do master the material. A CHI Square test of independence showed no differences in the frequency of A,B,C,D,F, or W grades for students with ACT scores <15, between 15 and 20, 20 and 25, and greater than 25.

If students with low ACT scores are successful in the COBOL course, maybe they spend more time on programs than students with higher ACT scores. Further research will examine variables such as: logon time, CPU time, and the number of times a program is compiled. It is expected that these variables will significantly increase r^2 above the level found in this study.

In summary, it is feasible for university personnel to identify which students are most likely to succeed as majors in COBOL based computer related disciplines. Students with good college grade point averages, demonstrated competencies in English and Mathematics, and successful completion of a course in algorithmic computations are more likely to succeed in a COBOL based program.

REFERENCES

(1) Alspaugh, C.A. (1972). Identification of some components of computer programming aptitude. Journal of Research in Mathematics Education,3, (9), 89-98.

(2) American College Testing Program, The. (1979). ACT Evaluation Survey Service. Iowa City, Iowa.

(3) Butcher, D.F. and Muth, W.A. (1985). Predicting performance in an introductory computer science course. Communications of the ACM,28, (3), 263-268

(4) Campbell, P.B. (1984, November). Computers in bilingual classes: Effects of sex and language dominance. Paper presented at the American Educational Research Association: Special Interest Group on Women and Education Annual Meeting, Long Beach, CA.

(5) Campbell, P.F. and McCabe, G.P. (1984). Predicting the success of freshmen in a computer science major. Communications of the ACM,27, (11) 1108-1113.

(6) Capstick, C.K., Gordan, J.D. and Salvadori, A. (1975). Predicting performance by university students in introductory computing courses. SIGCSE Bulletin,7, (3), 21-29.

(7) Dalby, J. and Linn, M.C. (1985). The demands and requirements of computer programming: A literature review. Journal of Educational Computing Research,1, (3), 253-274.

(8) Fowler, G.C. and Glorfeld, L.W. (1981). Predicting aptitude in introductory computing: A classification model. AEDS Journal,14, (2), 96-109.

(9) Hess, R. and Muira, I. (1983). Sex differences in computer access. The Forum for Academic Computing and Teaching Systems,2, 91-97.

(10) Hoc, J.M. (1977). Role of mental representations in learning a programming language. International Journal of Man-Machine Studies,15, 87-105.

(11) Hostetler, T.R. (1983). Predicting student success in an introductory programming course. Proceedings of the NECC5. Silver Spring, MD.

(12) Konvalina, J., Stephens, L., and Wileman, S. (1983). Identifying factors influencing computer science aptitude and achievement. AEDS Journal,16, (2), 106-112.

(13) Kurtz, B. (1980). Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. ACM SIGCSE Bulletin,12, 110-117.

(14) Maddala, G.S. (1977). Econometrics. New York: McGraw-Hill.

(15) Mazlack, L.J. (1980). Identifying potential to acquire programming skill. Communications of the ACM,23, (1), 14-17.

(16) Miller, L. (1981). Natural-language programming: Style, strategies, and contrasts. Perspectives in Computing,1, 22-33.

(17) Ott, L. (1978). Admissions management with the focus on retention. New Directions for Student Services,3, 23-28.

(18) Peterson, C.G., and Howe, T.G. (1979). Predicting academic success in introduction to computers. AEDS Journal,12, (4), 182-191.

(19) Ralston, A. and Shaw, M. (1980). Curriculum 78 - is computer science really that unmathematical? Communications of the ACM,23, (2), 67-70.

PROGRAMMING LANGUAGE AS A COMPONENT OF THE CURRICULUM OF AACSB-ACCREDITED DEGREE PROGRAMS

by

Patricia A. Merrier
and
Thomas B. Duff
Department of Finance and Management Information Sciences
University of Minnesota, Duluth

ABSTRACT

This study investigated the method by which AACSB-accredited institutions met the Assembly's general information systems standard/guideline. The findings indicate that the majority of the 201 institutions which formed the sample for the study required completion of a general, introductory computer course. Programming, however, was included within the general course at a majority of the institutions.

The American Assembly of Collegiate Schools of Business (AACSB) is recognized as the official accrediting agency for educational institutions offering programs in business administration and/or accounting. Program accreditation is viewed positively by prospective students, by employers, and by the graduate schools to which those who have completed programs apply.

The accreditation function is carried out by a council representing institutions which have been accredited at the baccalaureate and/or masters level. The statement of philosophy below is taken from the booklet outlining the Council's 1986-87 policies, procedures, and standards.

> AACSB accreditation fosters the attainment and maintenance of excellence in undergraduate and graduate education for business administration and accounting. The process evaluates business administration and accounting programs in terms of standards of performance recognizing the importance of diversity in higher education in management. The process is dynamic, and the interpretation of the standards is subject to change as environments, technology, circumstances and constituencies differ.

While not identical, the standards for programs in business administration and accounting do reflect recognition of the impact technology has had on business operations. For business administration programs, "The curriculum shall be responsive to social, economic, and technological developments . . ." Specifically, the common body of knowledge shall include ". . . a basic understanding of the concepts and applications of accounting, of quantitative methods, and management information systems including computer applications; . . ." Accounting programs are to include at least 25 percent of total coursework drawn from the common body of knowledge. In addition, 15 to 25 percent of coursework should focus on accounting, including "Computerized management information systems."

While DPMA and ACM are straightforward in listing specific courses/content areas, the AACSB Accreditation Council does not prescribe the methods by which institutions must meet its standards/guidelines. Some institutions may require individual courses in each area, others may integrate content into various courses. In either case, courses may be offered within or outside the business school.

PURPOSE OF THE RESEARCH

The experience of and informal research by the authors of this paper indicate that the once well-established pattern of requiring all business students to complete at least one programming language course is being abandoned. This study was conducted to verify that supposition.

Specifically, the researchers sought answers to the following questions for AACSB-accredited programs:

1) Is the accreditation standard related to information systems more likely to be met by requiring completion of a programming course or a general, introductory computer course?

2) If a general course is used, is a programming component included?

3) What programming language(s) is/are taught?

4) Is the course taught within the business school or by another unit?

THE POPULATION

The 244 accredited institutions listed in the 1986-87 AACSB Membership Directory formed the population for this study. Forty-three institutions were eliminated

from consideration--2 Canadian schools; 12 accredited only at the masters level; 8 for which no fiche was available in the library being used; 12 because no course descriptions were included in the bulletin; and 9 because insufficient information was contained in the bulletin to respond to the research questions -- leaving a sample of 201 institutions for which results are reported.

PROCEDURES

Microfiche copies of the bulletins of the 201 AACSB-accredited institutions were examined to gather data necessary to respond to the research questions. In addition, information was obtained about the academic term of the institution and the year(s) covered by the bulletin. The bulletin copy used was the most current available, the 1985-86 copy published by Career Guidance Foundation.

Information was sought only about the method by which the institution met the general AACSB standard/guideline. MIS/CIS concentrations or programs were not reviewed.

Data were analyzed using programs from SPSS-X.

RESULTS

Over three-fourths (161; 80.1 percent) of the schools used the semester as an academic term; the remainder operated on the quarter system.

The bulletins examined for this study covered a variety of years. As shown in Table 1, the majority (184; 91.5 percent) included the 1985-86 academic year.

Table 1
BULLETIN YEARS INCLUDED IN STUDY
(N=201)

Year	Number of Schools
1982	1
1983	6
1984	10
1985	184

Nearly 90 percent (179; 89 percent) of the schools meet the AACSB standard by offering only a general, introductory computer course; 19 (9.5 percent) require completion of only a programming course; and 3 (1.5 percent) require completion of both. Since the number of schools offering both types of courses is so small, data for these institutions has not been treated separately.

As shown in Table 2, the general, introductory courses are most frequently offered as part of the lower division core.

Table 2
PLACEMENT OF GENERAL COURSE OFFERINGS
(N=182)

Placement in Program	Frequency	Percent
General Education	2	1.1
Lower Division Core	160	87.9
Lower Division Business Administration	1	.5
Upper Division Business Administration	1	.5
Lower Division Elective	1	.5
Upper Division Core	17	9.5

Nearly all of the general, introductory computer courses (172; 94.5 percent) contained a language component. BASIC was the most frequently taught language. About one-third (66; 36.3 percent) of the course descriptions were worded to indicate that a language was included, but no language was specified. Additional information is contained in Table 3.

Table 3
LANGUAGES TAUGHT IN
GENERAL, INTRODUCTORY COMPUTER COURSES
(N=182)

Language	Frequency	Percent
None	10	5.5
BASIC	39	21.4
COBOL	4	2.2
FORTRAN	7	3.8
PL/1	2	1.1
Pascal	2	1.1
Survey of Languages	34	18.7
Programming included; language not specified	66	36.3
Other*	18	9.9

*Other includes programs where the course and/or language component differs for business administration and accounting, those in which students were given a choice of language, and other languages.

The general, introductory computer courses are most often offered within the business school. This occurs at over half (133; 73.1 percent) of the accredited institutions. Further details are presented in Table 4.

Table 4
UNITS TEACHING
GENERAL, INTRODUCTORY COMPUTER COURSES
(N=182)

Unit	Frequency	Percent
School/College of Business	133	73.1
Computer Science	37	20.3
Mathematics	2	1.1
Other	10	5.3

Over 80 percent of the schools which meet the AACSB standard/guidelines by requiring completion of a programming course offer the course as part of the lower division core. See Table 5 for details.

Table 5
PLACEMENT OF PROGRAMMING COURSE OFFERINGS
(N=22)

Placement in Program	Frequency	Percent
General Education	3	13.6
Lower Division Core	18	81.8
Lower Division Accounting	1	4.5

As shown in Table 6, half (11; 50 percent) of the institutions which require a programming language of business students offer those students a choice of the language in which they enroll.

Table 6
PROGRAMMING LANGUAGES REQUIRED
(N=22)

Language	Frequency	Percent
BASIC	4	18.2
COBOL	3	13.6
FORTRAN	2	9.1
Choice of Language	11	50.0
Different for Business Admin./Accounting	1	4.5

The programming classes are most frequently taught by units outside the school/college of business. See Table 7 for details.

Table 7
UNITS IN WHICH PROGRAMMING IS TAUGHT
(N=22)

Unit	Frequency	Percent
School/College of Business	5	22.7
Computer Science	17	77.3

SUMMARY/DISCUSSION

The findings of this research support the supposition that general, introductory computer courses have replaced programming courses as a requirement within business programs. Programming has not been completely abolished, however. The findings reported here indicate that it is incorporated as part of the general courses in a majority of the institutions included in this study.

Schools seeking accreditation may wish to follow the pattern set by schools already accredited. Namely, they may wish to meet the accreditation standard/guideline related to computer applications by offering a general, introductory course which includes a high-level programming language component, probably BASIC. To follow the pattern more specifically, such schools may wish to offer the general introductory course as part of the lower-division core taught within the business school.

TEACHING COMPARATIVE PROGRAMMING LANGUAGES TO MIS MAJORS

James E. Benjamin
Southern Illinois University at Edwardsville
Management Information Systems
Campus Box 1106
Edwardsville, Illinois 62026

ABSTRACT

The MIS Department of the SIUE School of Business has developed and offered a new course to further broaden their students grasp of programming tools with a minimum use of the students academic time. The rationale for the offering, student characteristics, methods and experiences are shared.

BACKGROUND

Schools are forced to make tightly constrained decisions on MIS curriculum in the face of an ever increasing supply of software creation tools and the very finite time of students. First, what minimum programming skills must the student possess to be initially employable in the MIS field and second, what can you teach them about programming that will have lasting value beyond their first few years of employment? SIUE School of Business decided some fifteen years ago that in-depth COBOL was essential to meeting the first requirement. (This later came to be a popular decision among Schools of Business but was very open to question when initially made.) During recent years SIUE has tried to address the second requirement with a brief introduction to BASIC during the required Introduction to Data Processing course and elective classes in Fortran Programming, Simulation Programming, Applied Operating Systems Principles, RPG Programming, Pascal Programming, Micro Computer Software tools, Fourth Generation Software seminars, C Programming, etc., as available on campus. This approach led to students who always felt they needed more language classes and who usually exhibited very detail-specific knowledge of certain languages with little appreciation of programming tools in general. The latest and best attempt to address this second issue has been to offer a class in Comparative Programming Languages as discussed in the remainder of this paper.

RATIONALE

The one thing inevitable is that programming tools will evolve (as seen in several generations of FORTRAN and COBOL standards) and/or die (as seen in many other now extinct languages). Also, new and higher levels of tools will evolve and take over the majority of the work done with the older tools. This can be seen historically in the move from machine to assembler to procedural languages and is currently evident in the development of many fourth generation programming tools and the great striving for artificial intelligence or fifth generation tools. Students will work in this rapidly changing environment and need something with more guaranteed longevity of usefulness than simply knowing COBOL. The SIUE School of Business MIS Department approach to this dilemma was

to plan a course explicitly intended to fill the need. The course tries to build a broad generalized understanding of programming tools based on in-depth knowledge of only one language and a broad brush understanding of key features of several others.

STUDENT CHARACTERISTICS

The initial offering of the Comparative Programming Languages class was made Spring Quarter 1986 as an elective MIS seminar available to advanced MIS students and others with the instructors permission (See attached syllabus). Permission was granted based on successful completion of an advanced programming course in at least one language, a hands-on programming experience with a second programming language and an expressed interest in finding out the relationships among various software tools. Twenty one enrolled students proved to include not only MIS majors, but also Computer Science majors and some MBA students! The breadth of background and interest of the students was felt to make a valuable contribution to the seminar as well as making it more challenging to teach. The course became a part of regularly scheduled offerings with the new academic year.

The chief resource language of the class was COBOL as most students had taken the second quarter Advanced COBOL class which required designing, writing and debugging about six or more programs covering file maintenance for three file organizations. The primary variable among the students was their secondary and sometimes tertiary language experience plus their point of view in approaching the computer as a technological artifact. The instructor learned to his dismay that the students almost universally had a very parochial and limited view of the nature of computer software tools in spite of frequent exposure to many different ones. The hoped for osmosis effect in having students learn to appreciate the character of programming through learning to program in several specific languages was indeed very poor. Even minor insights, such as the general ability to program a simple segment of code interpreter for one language using another, came as a major "dawning of the light" for most of the class. Because they had not lived through it the majority of the students had no appreciation for the generally accretive character of software where each successive generation builds upon, if not with, the tools of the prior generations. Even though many of these ideas were mentioned in some of the students prior supporting and introductory materials, the ideas had seldom been seen as germane by the students and were not integrated with their explicit knowledge of programming. The students badly needed some unified frame of reference upon which to attach their current programming skills. This broader perspective would help them continue to grow independently in their knowledge and use of further new tools in the future, beyond school and formal classwork.

COURSE METHODS

Fortunately, the literature available to support such a course is reasonably extensive in textbooks (2,4,8,9,11), historical library references (3,7,10,12) and current events articles (1,5,6,13). PROGRAMMING LANGUAGES - Design and Implementation by Terrence W. Pratt in the 1984 second edition from Prentice-Hall Inc. was the textbook closest to supporting the aims of the course. Many of the students felt the technical detail was a little heavy in some areas, but in general the book served its purpose well. A modification of the authors first alternative approach to the material, which he preferred for graduate classes, rather than his second undergraduate method was used. This first method consisted of going directly through the initial

theoretical groundwork chapters first before tackling the specific languages in anything but minor point supporting details. (Pratt felt that with undergraduates it was better to work back to theory from specific languages.) The theory first approach was chosen on the dual grounds that the students already had some strength in language specifics and that it went most directly to the main purpose of the course.

The keynote idea of the class was centered around a graphic illustration (See Fig. 1) and lecture that provided the first week on the evolutionary, continuing and accretive nature of the development of computer tools. This extends from hardware design, as the zero generation level of software, through the third generation tools that the students were most knowledgeable about, to the fifth generation of software now envisioned. We discussed how initially the only way to express the problem solution method was in hardware wiring such as the ballistic trajectory computer and the plug board wired accounting machines. Next, with the advent of "software" programming as the first generation, code was written in machine language. Subsequently the second generation placed the tool of computer programs in the service of further programming for the first time with assemblers of crude mnemonic languages. The third generation raised the ante on sophistication of computer support with compilers and linkage loaders that expanded individual problem domain procedure statements to as many machine statements as necessary and inserted pre-written code for many cases. In the fourth generation the power of the machine support tools is again raised so that the user can state his problem in terms of a desired solution form rather than as a procedure to create it and the software will deliver. Hopefully in the fifth generation the sophistication of the support tools will be so great that they can be consulted with the problem, as an "expert" that will probably provide an appropriate solution. A major theme in all this is that each successive generation depends on the concepts and tools of the prior generations plus some new ideas of its own. Also, each generation of tools still continues to be used by limited groups of specialists while the vast majority of computer work grows upon and utilizes the newer tools. For example the zero generation is still alive and well among the practitioners of computer architectural design giving rise to micro-codable machines, Programmed Read Only Memories and to such recent items as the Reduced Instruction Set Computers. Some micro-computer users still PEEK and POKE machine language level patches into third generation BASIC programs. Also, much system level software is still coded in second generation assembler and some third generation compilers still create assembler code as an intermediate step during compilation. Even fourth generation tools at present need to have procedural level tools within them to be very general purpose and probably fifth generation tools will need similar "escape hatches".

The hierarchy of virtual computers concept of the author (See Fig. 2) was heavily stressed as a way to look at the implementation details of any specific language embodiment. A central theme was that there was so much to be known about programming languages and so little time to learn details that their only realistic recourse was to attempt to use classification and abstraction as tools to get the "big picture" with only limited detail examples. The purpose of the text was twofold. First, it was to provide us with quickly digestible pre-packaged concept frameworks and terminology for well understood language elements. Second and more importantly, the text was to serve as a guide by example for formulating further new generalities in the future as the necessity arises. In spite of their myriad detail differences, programming

languages do share a great common body of concepts, methods and behaviors when examined from that point of view.

Weeks two through five the students had reading assignments, lectures and attempted discussions on chapters 2 through 7 and 12 through 15. The first group of chapters on general theory were to be studied as the key issue and the second group on individual languages were to be read as background as best they could. On this basis they covered language processors, data types (elementary, structured and programmer-defined), subprograms, sequence control and data control with modest attention to FORTRAN, COBOL, PL/1 and Pascal.

MODIFICATION BASED ON EXPERIENCE

Because class interaction was relatively disappointing, some open ended questions were used on the midterm exam to try to elicit how to achieve more class participation and responsiveness. They were asked what surprised or interested them most of what they had learned in class to date, what they found least interesting, what their particular contributions had been to the seminar and what they suggested to make the second half better. Reading and tabulating the responses led to the following conclusions:

The students identified well with the keynote theme

Very elementary concepts frequently came as revelations to them

They wanted more structure in their assigned requirements

More abstract concepts and details were confusing and threatening

They were rather embarrassed over their participation level.

Based on this information two sessions were held where the class was responsible for determining the structure of the remaining meetings and assignments. In democratic discussion they opted to break the theory into elements and sub-elements of their own designation (which closely paralleled the text but were modified and extended to some extent) for further study. They accepted small group assignments covering these elements which they would prepare and present orally to the class and in writing to the instructor on a schedule. Each group also agreed to be responsible for presenting the applicability of their theory element to each of the eight languages covered in the text. As a final class project they would each individually present verbally and in writing a discussion of how their chosen theory elements applied to dBASE (a language not touched on in the text). They would thus attempt to synthesize a complete view of a "new" language in the form of the theory constructs we had covered. They also decided to bring in outside readings or special knowledge of interest for class presentation for extra credit. The instructor agreed to edit and assemble the written version of their combined investigation of dBASE and make it available to them on request the following quarter. By this process their final submission was converted into a 38 page double spaced draft analysis of dBASE from a comparative point of view.

On the whole the formal and informal feedback from the class was quite positive. The grade span (based on both oral and written contributions) was a bit wider than expected for an advanced elective seminar, but was a reasonable assessment of the acquired knowledge of the students.

BIBLIOGRAPHY

1. D. S. Appleton, "The Technology of Data Integration." Datamation 31, 21, (November 1985), 106-116.

2. D. W. Barron, An Introduction to the Study of Programming Languages. Cambridge, MA, Cambridge University Press, 1977.

3. Hammer, Howe, Krushal and Wladawsky, "A Very High Level Programming Language for Data Processing Applications." Comm ACM 20, 11 (November 1977), 832-840.

4. Higman, Bryan and McDonald, A Comparative Study of Programming Language. New York, NY, London and American Elsevier 1977.

5. Jimenez and King, "Should you Consult a Guru?" Data Based Advisor 4, 6 (June 1986), 8-17.

6. D. Kutnick, "Whither VM?" Datamation 31, 23 (December 1985), 73-78

7. J. A. N. Lee, "Considerations for Future Programming Language Standards Activities." Comm ACM 20, 11 (November 1977), 788-794.

8. B. J. MacLennon, Principles of Programming Languages: Design, Evaluation and Implementation. New York, NY, Holt Rinehart and Winston, 1983.

9. J. Martin, Fourth Generation Languages Volume I Principles. Englewood Cliffs, NJ, Prentice-Hall 1986, pp. 1-31 (Introduction).

10. J. Merritt, "Pascal is Here to Stay," Interface Age (May 1979), 52-54.

11. T. W. Pratt, Programming Languages: Design and Implementation. Englewood Cliffs, NJ, Prentice-Hall 1984.

12. J. E. Sammet, "Roster of Programming Languages for 1974-75." Comm ACM 19, 12 (December 1976) 655-669.

13. L Sigler, "Relating to Database Machines" Datamation 32, 7 (April 1986), 83-89.

SPRING 1986
MIS ELECTIVE COURSE

Day Section TU,TH 12:20-2:30 Night Section TU 6:30-10:20

MIS495
COMPARATIVE PROGRAMMING LANGUAGES

This is the first offering of this MIS elective course. As such it will be done in a seminar format with somewhat more latitude in terms of adherence to details in this syllabus than a standard established course. To permit this developmental flexibility and to sustain a high level of dialogue, we will attempt to keep the class size modest and limited to those with a real interest in the topic.

Who should be interested?

Any MIS major who has completed MIS301 or MIS568 COBOL and has a strong interest in designing, choosing or using computer software tools in their future career should find the topics covered of considerable value.

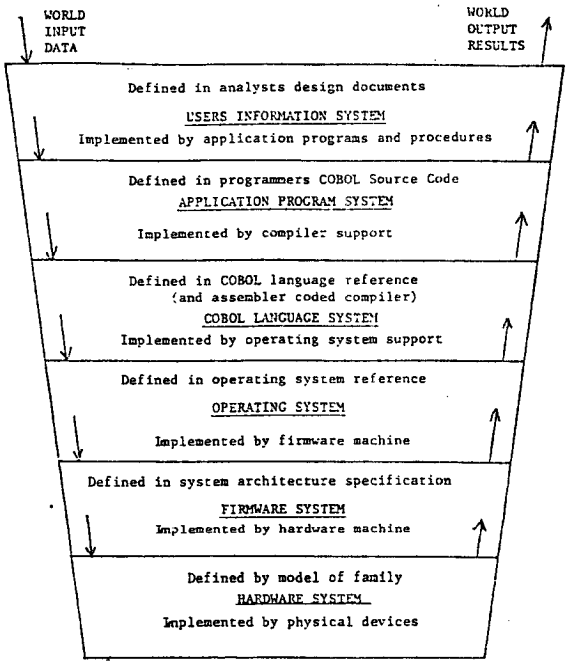
What do we want to do?

We will seek to expand our knowledge about programming languages and software tools by building on the significant specific knowledge that we already have from using COBOL. In doing this we will attempt to abstract what we already know into more general terms which can be applied in making comparisons to other software. We will look at several other programming languages that possess different powers and properties. We will also study the history of this on-going evolution of software so that we can assess where we came from, approximately where we are now, and possibly where we are apt to go next. To a limited extent we will examine the design criteria and trade-offs made in the design and implementation of programming languages. In this process we should achieve a different and deeper perspective on the languages we already know, as well as a much broader perspective on other languages and software.

Why?

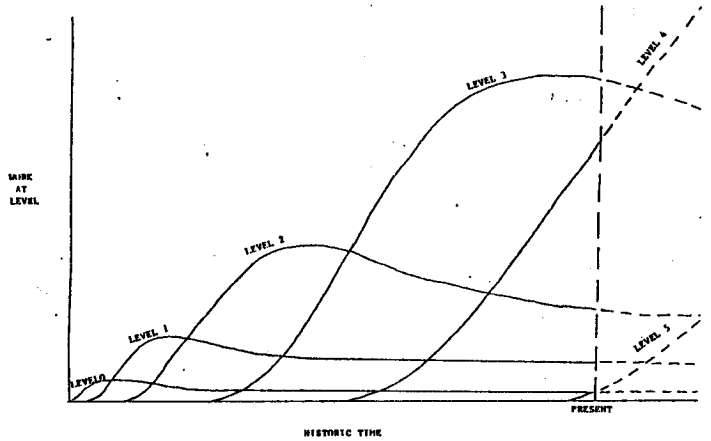
During your careers there will be major on-going changes. New hardware features will be developed, new languages invented and new applications devised. Your university training cannot provide you with the as-yet un-invented specifics of these new things that you will have to cope with. What we can hope to provide you is a frame of reference, some classification tools, some theories and, possibly most importantly, some methods of examining such developments. The principles you learn from the process of this course will be applicable for a much longer time and over a much broader range of topics than the specific current findings.

FIGURE 2 LAYERS OF
VIRTUAL COMPUTERS
IN A COBOL ENVIRONMENT



EACH LAYER CAN BE VIEWED AS A VIRTUAL
MACHINE WITH I/O TO SUCCESSIVE LAYERS

FIGURE 1 EVOLUTION OF
SOFTWARE LEVELS
AND USEAGE



Prolog as an Introduction to Computer Information Systems for Non-Information Systems Majors

David B. Paradice

Department of Business Analysis and Research
Texas A&M University

Abstract

An informal survey of students that have completed an introductory computer information systems course indicates that most non-information systems majors do not perceive any benefit from having taken the course. This paper suggests that the student's perceptions are due to lack of focus on the development of problem-solving skills, and too much focus on the nuances of programming. A recent trend toward packaged software such as database and spreadsheet packages is unlikely to address this problem. Prolog, a logic programming language, has characteristics which may refocus the introductory course on problem solving and hence may be a better vehicle for developing computer-based skills at this level.

INTRODUCTION

A recent informal survey of sixty-nine junior and senior college of business students indicates there may be a need for serious reconsideration of the contents of the university's introductory computer information systems course. When asked to indicate agreement or disagreement with the statement "The things I learned in [the introductory course] will help me in my major," over half of the respondents indicated disagreement with this statement. As might be expected, the percentage of students agreeing with the statement dropped when the responses for information systems majors were removed from the sample.

Although in many cases students do not really know what subjects will be most beneficial later in their careers, such a response is still noteworthy because it indicates a general failure of the course to convey WHY it is an important course in the student's core curriculum. In light of the fact that many business schools are currently redefining the content of their introductory computer course requirement, this may be a most opportune time to re-evaluate the intended goals of such a class.

OTHER SURVEY RESULTS

Space limitations preclude reproducing the survey data here, although some highlights may be mentioned. An overwhelming majority of the students recognize, for example, the importance of learning to use computers to solve problems (ninety-four percent). However, only one-third of the sample indicated that learning to program was important, an opinion shared by many faculty in light of today's plethora of available "off-the-shelf" software. Still, over seventy percent of the students indicated that they learned to program in the introductory course.

At this point, one might reasonably ask "what types of skills should a student acquire as a result of successfully completing the introductory computer information systems course?" First, the student should ultimately recognize the computer as a problem-solving tool. Second, the student should acquire the skills needed to use the computer to solve problems.

The survey seems to indicate that the first objective may be accomplished with little effort. Students already recognize the importance of learning to use computers to solve problems. The survey also indicates that learning to program is not perceived as a valid method for learning to solve problems using a computer. More often, programming is perceived as a means for solving the problem of using a computer.

This latter point is the focus of this paper. An experienced programmer easily "sees" ways of using a computer to solve problems. However, many students in the introductory computer information systems course are not experienced programmers. When given a problem to solve and forced to provide a computer-based solution, the student frequently turns in the first solution which solves the problem of using the computer, not the solution that best solves the problem. Some students become so involved in trying to get the computer to do what they want that they lose sight of whether what they want is worthwhile. Further, a transition from BASIC to spreadsheet and database packages may not address this issue.

THE APPLICABILITY OF BASIC AND SPREADSHEETS

Programming is frequently misused as a surrogate exercise for problem solving, or for teaching problem-solving skills. As such, BASIC and other programming languages have been taught under the belief that "if a student can think through a problem to the extent that a program is written which reflects a sequence of steps which solves the problem, then the student has 'solved' the problem in a logical manner."

When the introductory computer information systems course is considered as a course to teach computer-based problem-solving techniques, especially to non-information systems majors, one may begin to identify concepts that must be developed within the course. The two most fundamental concepts are probably data retrieval and data manipulation. Many of the assignments at the introductory level consist simply of data input followed by data manipulation. "Output" is a third fundamental concept, but one which typically gives students little trouble at this level.

The current trend toward database and spreadsheet packages reflects the importance of these concepts at this level. The database packages address data retrieval; spreadsheets address data manipulation. The student, now confronted with more "languages" to learn, however, may be further detracted from focussing on the real problem to be solved.

ARE PROGRAMMING AND PROBLEM SOLVING SIMILAR?

A close examination of some problem-solving processes would indicate that programming as it occurs in these introductory courses can be antithetical to some problem-solving processes. Consider the problem of reconciling a financial balance sheet.

Ask someone to specify the problem-solving process for this problem and the likely response will be that the total assets must equal the total liabilities. When asked to elaborate, the problem solver may then indicate that total assets is comprised of current assets and long-term assets, and total liabiliies are the sum of current liabilities, long-term liabilities, and owners's equity. Press the problem

solver a little more and you may be told that one must add cash, accounts receivable, and short-term investment income to calculate current assets, and that the other quantities defined thus far may similarly described in terms of specific data items.

Now, how must one program this solution in either BASIC, a database package (with supporting data manipulation language), or a spreadsheet package? One must program the solution in the reverse order of the problem-solving process described above. That is, one must first identify and evaluate the proper data items, then calculations must be made, then the totals must be compared. In this way, the programming process is counter-intuitive to the problem-solving process.

In this light a language such as Prolog may be seen as a viable alternative. Very simple Prolog programs may be constructed which solve problems perceived as very difficult by students at this stage of their college career. Since the programs are easily specified, students can focus on the problem to be solved, instead of the problem of writing the program.

PROLOG AS AN ALTERNATIVE

Prolog has essentially one construct: the rule. Rules are specified in terms of goals to be satisfied. A rule that is always true may be called a fact. The fact that "accounts receivable" in 1986 was $250,000 may be programmed in Prolog as:

```
data (accounts-receivable 1986 250000).
```

Accessing this data is accomplished by the following query:

```
data (accounts-receivable 1986 X).
```

Although dialects vary regarding the specific query format, essentially all that is needed is to supply a variable, in this case X, to receive the value requested.

Rules may be conditional on the truth of other rules, in which case the rule is true when all of the "other" rules are true. The "other" rules represent "goals," since successfully proving their truth proves the truth of the conditional rule. So, a Prolog rule may be conceptually pictured as a set of goals to be achieved. When the conditions are satisfied (i.e., when the goals which represent the conditions are achieved), then the conclusion can be achieved.

By considering the problem of constructing a balance sheet posed earlier, Prolog's data manipulation capabilities may be examined. Notably, this problem might be a "spreadsheet" problem in some introductory courses. Suppose that current assets are defined as the sum of cash, accounts receivable, and inventory. In Prolog, the rule for calculating current assets can be stated as follows:

```
   calculate (current-assets Year Value)
if
   data (cash Year X) and
   data (accounts-receivable Year Y) and
   SUM (X Y Intermediate-value) and
   data (inventory Year Z) and
   SUM ( Intermediate-value Z Value).
```

Words which are entirely capitalized represent functions normally available in Prolog (intrinsic functions). Arguments that begin with a capital letter are variables. The rule can be read:

> The goal of calculating the value of current assets in a given year can be satisfied if the goal of accessing the data for cash for the year supplies a value for x and the goal for accessing the data for the value of acccounts receivable for the year supplies a value for y and x and y may be added to get an intermediate value and the goal of accessing the data for inventory for the year supplies a value for z and the intermediate value and z can be combined to produce the value (of current assets).

At first glance, this appears quite confusing. There is much to be said for this approach, however. Since Prolog is "nearly non-procedural" (in some cases one must specify rules in a specific order to control Prolog's searching procedures) the rules for the calculations can be specified in any order. Consequently, the student is allowed to concentrate on the process of solving the problem. Data for a problem are represented as facts. The program is specified as a set of rules which manipulate the facts.

In Prolog, as in other logic programming languages, a variable's scope is restricted to the rule in which it occurs. For example, in the rule for calculating current assets, the value of the variable Year must be the same for all occurrences of the variable within the rule, but Year may assume different values in other rules. Essentially, every rule is a subroutine. With Prolog, "subroutines" (and hence structured programming) are taught on the first day. With BASIC, weeks may pass before these concepts are developed.

Since the scope of Prolog variables is restricted to the rule in which the variable occurs and rules may be written in any order, each rule represents a complete specification. Therefore, the correctness of each rule may be determined separately from the occurrence of any other rule. This characteristic allows instructors to points out flaws in the problem-solving process instead of the programming process. Once a rule has been correctly specified, the addition or deletion of any other rule cannot alter the correctness of an existing rule.

SUMMARY

Prolog provides an essentially database environment where data are represented as facts and data are manipulated through rules. Further, most Prolog implementations currently provide all spreadsheet type calculations. Consequently, all data driven applications can be implemented in Prolog, although some extended features (e.g., report writing facilities) have yet to be implemented.

Additionally, Prolog provides an uniquely incremental, verifiable means for teaching students to use computers to solve problems. Consequently, instructors of introductory computer information systems courses could emphasize problem-solving techniques over programming techniques with Prolog. Such an approach may be more beneficial to non-information systems majors in a college of business.

REFERENCES

(1) Barnes, Cynthia C. "Teaching Computer Literacy: A Nontraditional Approach." Journal of Education for Business, (April 1986) 311-314.

(2) Clark, K.L. and McCabe, F.G. micro-Prolog: Programming in Logic, Englewood Cliffs NJ: Prentice-Hall, 1984.

(3) Clocksin, W.F. and Mellish, C.S. Programming in Prolog, Berlin: Springer-Verlag, 1981.

(4) Paradice, David B. "Prolog: A Language for Teaching Computer-Based Business Problem Solving." Journal of Education for Business. Forthcoming.

(5) Mayer, R.E., Dyck, J.L., and Vilberg, W. "Learning to Program and Learning to Think: What's the Connnection?" Communications of the ACM, (July 1986) 605-610.

(6) Tetenbaum, Toby J. and Mulkeen, Thomas A. "LOGO and the Teaching of Problem Solving: A Call for a Moratorium." Educational Technology, (November 1984) 16-19.

(7) Watterman, Mick. "Logic to Code Pedagogy in BASIC Programming Classes." Journal of Education for Business, (May 1986) 374-377.

MIS INTERNSHIP: A MUTUALLY BENEFICIAL PROGRAM

Byron Dangerfield, University of Idaho
Norman Pendegraft, University of Idaho

ABSTRACT

The College of Business and Economics at the University of Idaho faces a challenge common to many business schools: meaningfully incorporating computer education into the curriculum. In addition, the college has other tasks, such as advising, which might be enhanced by some computer-based aid. At the same time students in our MIS program frequently lack meaningful professional experience before graduation. The creation of an MIS Internship Program has enabled us to address both issues at relatively low cost. This paper reports on our internship program and how it has improved the education of the students and provided service to the college.

INTRODUCTION

The development of good systems analysts requires hands-on experience. To the extent that that experience can be provided in the university environment, the quality of education received by the student is enhanced. In the following sections we discuss the needs of students and the college and the internship program. We then discuss how using student interns to solve the college systems problems provides a solution that is mutually beneficial to both the students and the college.

NEEDS

STUDENT

The systems analysis curriculum trains students in the principles and practices of the profession. Much classwork, however, is conceptual in nature and contains principles often stated in generalities. Without concrete examples or the application of these principles, the student would be left without a real understanding of the principle and what the various applications might be. Application of the principle aids in retention as well as understanding. Without actual application, retention of the material is also poor, extending slightly beyond the classroom exit on the day of the final exam. Teachers are discouraged to find in subsequent classes that prerequisite material must be dredged up and rehashed to lay the proper foundation for new material. One needs only imagine how that feeling is amplified for a new employer.

Case studies and in-class projects are generally used to fill the need for real applications. Case studies can be difficult for illustration, however, unless sufficient time is spend helping the student understand all facets of the application; there is not time enough to develop cases for each principle and few cases are comprehensive enough to illustrate all of the necessary points. Short cases typically are incomplete and require several assumptions, some critical to the formulation of the design.

Projects are more useful, especially with real clients. The investigation process can be directly experienced, and

unknown facts discovered. With larger classes and project teams, however, teachers don't have time to deal with students individually. Also, a student on a project team generally sees only a small part of the project, experiencing only a subset of the total project activities. A common student complaint at the end of the semester is "I didn't know what was going on," a failure of the project leader but still a problem for the student who has missed the experience. The result is that some students can graduate from a four-year institution with minimal actual contact with the problems and processes of systems analysis.

Another effect of lack of actual experience is low self confidence. Some MIS students occasionally express some fear and reservation about going into the job market--because they "can't do anything." What athlete could possibly attend a tryout having only read about football? While it is true that a major function of a college education is to teach students good work habits and problem-solving techniques, it is also true that they should have enough direct contact with their major field to know that they have chosen the right profession, one that is enjoyable and challenging, and that they have some real experience practicing it. They should, at the least, feel mildly confident that they can do the job.

Employers understand the nature of education, and most expect that a new employee will require some initial training period to become productive. But how much more attractive would a prospective employee be if some positive, direct experience were shown on the resume? One need only check the Position Announcements section of Computerworld.

COLLEGE

Applications are continually expanding, and technology is continually changing. The College of Business and Economics has a number of technical or system problems that await resolution: for example, COMPUSTAT tapes need access routines compatible with the current computer system, IFPS requires instruction of both faculty and student for use in classes, faculty research material is not directly portable between microcomputers and the mainframe, new facilities require installation of cables or circuit boards....New hardware and software demand continual attention. New technology and problem-solving techniques should be integrated into the classrooms, faculty office, and staff offices.

The resolution of these problems leads to inefficient use of faculty time. The university Computer User Services has enough staff to handle the more important problems. Many of the smaller problems are left incumbant on the colleges to resolve, a situation not unlike other MIS facilities. Those tasked with solving these problems are usually faculty members; in the College of Business this is usually the MIS faculty. While some problems are interesting, most faculty members are not interested in serving as systems analysts for the college. It is one of those collateral duties that yields little reward and consumes large quantities of time, time better used for other activities. One solution is to have a full-time member of the college staff trained in the area; this alternative is somewhat a luxury in today's world of shrinking budgets and an alternative many can not afford.

Another problem is that of retaining "corporate knowledge" in the technical areas. As faculty members change institutions, they take with the knowledge base accumulated at the college. Students who have worked on certain projects also graduate and take their accumulated technical skills elsewhere. If no one else on the faculty or staff shares those skills, the skills are

lost--a deletion anomaly, if you will. Such a loss forces a relearning of those skills at the high cost of entry-level learning.

The results for the college are ineffective use of technical resources (sometimes non-use), misdirected faculty activities, and poor utilization of scarce college funds; the cumulative effect of which diminishes thelevels of student education and faculty output.

IMPLEMENTATION OF THE INTERNSHIP PROGRAM

Our solution to the problems was to establish an MIS Internship Program. Since we do not have a graduate program in Information Systems, we turned to undergraduates to solve some of the problems discussed above and hired MIS students to serve as systems analysts for the college under the supervision of MIS faculty. With minimal funding we found we could provide challenging, real-world problems for our students and at the same time alleviate some of the problems of the college.

An MIS Internship Committee was formed to select students for the internship as well as to select the problems to be solved. The committee consists of the Dean of the College as chairman and two of the MIS faculty as members, one of whom serves as the co-ordinator of the program and directly supervises the activities of the interns. As such the co-ordinator becomes more of a project leader than a systems analyst.

Organizationally, the committee stands between the faculty (the clients) and the students who work for them. Isolating the students protects them from making decisions regarding the choice of projects and direction of work. This organization precludes conflicts which may prejudice student relationships with future teachers. The presence of the dean similarly helps protect the MIS faculty members and guarantees to them his support of the program.

Students are selected for the internship based on demonstrated classroom ability as well as possession of both critical skills which are of use to the college. Both juniors and seniors are selected. The juniors initially serve more as apprentices, having little technical background; the seniors have enough training to be more productive analysts. Ideally senior/junior pairs are assigned to specific projects as an analyst and assistant analyst. By the end of an academic year the graduating seniors should have passed their corporate knowledge base to the juniors. Specific activities are scheduled to ensure this transmittal occurs. Prospective projects are submitted to committee members on user request forms (designed by the system analysis class) and screened by the committee. Selected projects are assigned to the interns as they become available.

Table 1

	Biweekly Period								
Student	1	2	3	4	5	6	7	8	Avg.
Senior A	21	20	19	-	-	-	14	-	4.63
Senior B	5	4	4	-	4	5	4	1	1.69
Senior C	-	11	4	7	5	2	16	6	3.87
Senior D	20	4	-	-	-	-	-	-	1.50
Junior A	2	12	4	1	14	16	5	6	3.80
Junior B	13	12	15	4	7	10	10	-	4.44
Junior C	21	-	4	-	-	13	-	-	2.38
Average									3.19

The workload a full-time student can support is a major concern. The seniors initially attempted to work 20 hours per week and the juniors 10 hours per week. Experience has shown, however, that rarely do they actually have that much time to devote to the internship. Priority is given to normal school work with remaining time allocated to the internship. The table below shows the average hours reported for a semester as biweekly totals.

Note that work is reported sporadically. Students sometimes forget to report for one period and accumulate hours over multiple periods. The high variability is also due to the nature of the project assigned; some projects have deadlines, some are more interesting than others, ... A third reason for the variability is the natural interference of midterm, finals, and recruiting interviews which all tend to occur at given times during the semester. We have found that weekly project walkthroughs tend to level intern activity.

The current maximum allocation for seniors is 10 hours per week and for juniors 8 hours per week. Actual workloads are closer to three hours per week, but students are able to adjust when the project demands more time.

Specific projects assigned the students are classified below.

Requesting faculty ranged from those with little computer background wanting access to COMPUSTAT to MIS faculty wanting assistance in a computer-based experiment. Note that some projects provide problem solutions that give long-term results, that is, they are of benefit to many individuals over a long period of time. These we plan to solve once and then maintain for other faculty or students. Other projects are one-time, ad hoc requests that once satisfied provide no continuing requirement on the program.

COSTS

There is currently sufficient budget for four interns. The program as envisioned may eventually include a graduate who will work as the college systems analyst. Funding, however, has not been sufficient to support a full-time analyst.

The interns are paid at a rate more competitive than most university part-time employees. Salary schedules for our current annual budget are shown on the next page.

The actual costs are much less than budgeted costs because students are not able to sustain the maximum level of activity over the whole semester.

There are no extra hardware costs since

Table 2

Term	Project Name	Time to Complete
Long	COMPUSTAT FORTRAN Access Modules	Complete
Long	COMPUSTAT Handholding	Complete
Long	COMPUSTAT Users Reference Manual	Complete
Long	COMPUSTAT SAS Macros	Complete
Long	IFPS Student Manual	Complete
Long	IFPS Capabilities Faculty Presentation	Complete
Short	IFPS Operations Management Simulation	In progress
Long	College Advising Database	In progress
Short	BASIC Program Conversion	Complete
Long	College Pre-registration Database	In progress
Short	Classroom/Computer Feasibility Study	In progress

Table 3

Position	Rate/hour	Hours/week	Hours/year	Annual Cost
Senior A	$6.67	10	300	$2000
Senior B	$6.67	10	300	$2000
Junior A	$5.50	8	240	$1320
Junior B	$5.50	8	240	$1320
Totals			1080	$6640

there is sufficient excess capacity on the college mainframe terminals and microcomputers. There is some marginal increase in costs for mainframe access, supplies, manuals, and copies, but it is difficult to estimate what the marginal costs would be over normal faculty usage to solve the same problems. The real difference is that more problems are solved, so the costs are higher.

There is faculty time involved in serving on the committee and in serving as the coordinator. The coordinator is compensated with release time from one class per year.

BENEFITS

The students benefit in several ways. Foremost, they get real-job experience in systems analysis. They take projects from initial investigation through implementation and evaluation. They experience a wide range of problems and users with widely diverse backgrounds.

They also benefit in the process of job hunting. They have stronger resumes because they can offer employers meaningful experience. They also get better written recommendations from faculty members who have observed first-hand their performance as employees. Finally, those having financial difficulties find a source of funding their educations at a reasonable rate in a meaningful, work-related area--beats selling hamburgers!

Until now a faculty member who needed computer work done could do it himself, hire a workstudy student, or sponsor a senior project for a computer science major. Having tried all three, the authors have found that only the first gives dependable and satisfactory results. A good workstudy student may be unavailable and senior-project students cannot promise to finish the task since their first priority is to the class requirements, not the customer. Doing it oneself is frequently sufficiently time consuming to preclude doing it at all. This program has enabled us to screen prospective interns in advance and pay them enough to be confident the work will be done properly and in reasonable time.

Faculty response thus far has been very favorable. The economics faculty have been able to begin using COMPUSTAT in their classes as a result of our first major project and several others have been able to quickly learn IFPS enhancements through a series of intern presentations. One task involved translation of a program from one BASIC dialect to another--saving the faculty member several hours and providing the student a graphic lession in maintenance and transportability.

Current projects involving advising and preregistration databases should enhance the productivity of both faculty and staff. Projects such as the ongoing IFPS education program will enable greater use of computer-based tools by faculty in their courses, demonstrating to the students their value (and limitations) as well as making them available as research tools.

Thus, the college gets more problems solved more rapidly. Teachers can

integrate new techniques into classroom presentations. Researchers have better access to databases and organizational tools as well as analytical ones. The faculty in general has more time to devote to appropriate research and teaching activities. Staff efficiency is also enhanced.

We expect that the program will bring the MIS program greater visibility both on and off campus. We expect to attract more recruiters to the university as firms become familiar with the goals and training offered at the college. We also expect that the increase in recruiter visits and the availability of internship opportunities will bring the MIS area greater visibility on-campus and attract better students to the program.

CONCLUSIONS

The MIS internship program has greatly enhanced the quality of education in the college. It also provides an econmical means for a college that is basically an undergraduate institution to fund systems analysis projects and provide technical assistance to the faculty.

FOURTH GENERATION LANGUAGE AS A REQUIRED CIS COURSE

W. Gregory Wojtkowski, Ph.D., Boise State University, Boise, Idaho 83725
Wita Wojtkowski, Ph.D., Boise State University, Boise, Idaho 83725

ABSTRACT

The approach to teaching "high productivity languages" -- Fourth Generation Languages is described. The principles of the development of computer information systems through the justified use of fourth-generations langugaes are stressed.

INTRODUCTION

Nowadays more computer applications are implemented directly by the end-users. With the fourth-generation languages (4GLs) computer power becomes accessible to those without extensive training in data processing. The impact of 4GLs will shape the development of data processing in a very fundamental way [5]. That is why it is so important to rethink our attitudes about programming and how we teach it, especially in Colleges of Business.

In the Computer Systems & Decision Sciences Department at Boise State University students are exposed on two levels to 4GLs: in the introductory CIS course, which is required for all College of Business students, we stress the end-user orientation. Students are required to solve simple computer information system problems using 4GLs. Specific Fourth Generation Language is used in this introductory course in place of programming in BASIC [15]. This first level of exposure to 4GLs leaves the student with a skill in applications programming and an appreciation of the process of creation of simple information systems.

On the second level, which is oriented toward CIS professionals, the principles of the development of computer information systems using 4GLs are examined in depth. The state of the art design techniques appropriate for those languages are explored. Students learn to recognize the advantages and disadvantages of the use of 4GLs in major CIS application areas. On this level students undertake an intensive study of a specific 4GL -- POWERHOUSE -- a 4GL for HP3000. The emphasis is on problem solving and students are required to build complete computer information systems. Advantage is taken of rapid system development using 4GL. Students are also made aware of how important it is not to misuse applications of the 4GLs.

The course objectives are:

- -introduction to 4GLs and to the design techniques associated with 4GLs;
- -exposure to the capabilities of the 4GLs and their use in solving common business computer information problems;
- -practice of the prototyping technique throughout the entire course; and
- -introduction to the process of building decision support systems for business.

The content of the course and the rationale for the topics selection and placement in the CIS curriculum are described in detail below.

RATIONALE FOR THE PLACEMENT IN THE CIS CURRICULUM AND THE TOPICS SELECTION

Pedagogical reasons for teaching 4GLs have been already explored [1]. Fourth Generation Languages improve students productivity so that complex, real-life computer information systems problems can be used as CIS class projects [14].

Boise State University students majoring in CIS are required to take second level course in the first semester of the senior year. This course provides senior-year students with theoretical and practical orientation for subsequent, more advanced, software design course, in which complete system together with completed documentation, is built. It is our contention that this better prepares our students to be

CIS professionals in the 1990s as by that time ninety-five percent of the major organizations will have implemented 4GLs [1].

We read a lot about productivity improvement when 4GLs are used [3][7]. We are also aware of horror stories describing situations in which systems written in 4GLs are extremely inefficient. Therefore, it is very important to expose students to advantages and disadvantages of such languages. Very often, in the real world, the computer information systems written in 4GLs are designed using old design techniques which in reality are not appropriate and can cause unsuccessful system implementation [1]. Therefore, it is essential that the technique of prototyping is introduced to the students as one of the most crucial new tools in designing systems with 4GLs.

The topic selection is as follows: first, the survey of 4GLs is considered next, principles of prototyping and then specific 4GL (in our case the POWERHOUSE -- the 4GL for HP3000) is taught in depth using the prototyping approach. The training schema is represented on Figure 1.

In this way all capabilities of the specific 4GL are introduced using cases of complete computer information systems and gradually increasing level of difficulty so that all the time students are improving their problem-solving skills.

This way our students become better prepared CIS professionals: they have deeper understanding of the underlying processes involved in the creation of computer information systems.

COURSE CONTENT

Unit 1, Basic Definitions and Survey of 4GLs, provides students with the definition of the languages in this category. It also introduces the student to many different kinds of 4GLs and their capabilities. Having completed this part the student will have been taught how to reorganize the 4GL, and its appropriateness for the specific computer information problem.

Unit 2, Prototyping -- New Tool for Systems Development, focuses on helping students with mastery of the prototyping technique. This serves as an aid to creation of complex computer information systems. The first part of this unit introduces prototyping and points out advantages and disadvantages of such a system development technique. The second part teaches students how to apply the prototyping in his work. Throughout this unit, students are provided with numerous examples of prototyping with 4GLs.

Unit 3, Introduction to POWERHOUSE, teaches students the skill of building simple computer systems. As the unit continues, students are taught to define and build simple files, to generate screens for data entry and test them, and to generate reports. A series of simple case problems are defined so that students can practice building simple prototypes. That will strengthen students' ability to understand prototyping and to clarify how systems are created using 4GL.

Unit 4, Formulating and Analyzing Business Computer Information Systems, presents students with four problems which they can develop into integrated computer prototypes. Which the examples of problems are: state park campsite reservation system, budgeting system, on-line inventory system, and payroll system for small businesses. This unit begins with the introduction of direct access files and options in the dictionary for defining detailed attributes of data elements. A series of screen enhancement and customizing capabilities are introduced, so that students are capable of designing professional looking screens with layout control, assigning field ID and labels, providing operator feedback, format options and operator conveniences such as line drawing and highlighting. The menu screens are introduced so that students will be able to integrate their separate programs into computer systems. Report capabilities are enhanced by introducing multiple-file access, custom formatting such as initial heading, page heading, summary operations, control

footing, page footing, and final footing.

This set of 4GL capabilities was chosen because it illustrates the most common types of business information processing needs. At the completion of this unit, students should be able to formulate and implement standard business computer information systems.

Unit 5, Formulating More Complex Computer Information Systems, deepens the discussion begun in Unit 4. Students are also introduced to some more advanced topics on 4GLs such as security, database design with multiple files, split screens, and multiple screens, subfiles processing, calculations, and processing with multiple passes through the data. Prototypes build in Unit 4 are enhanced so they look as professionally built complete computer information systems.

Unit 6, Decision Support Systems, introduces students to 4GLs option such as graphics and statistical analysis so students can integrate computer information system into complete decision support systems.

CONCLUSIONS

Given the results from our pilot courses data, preliminary indications are that the course design thus far is meeting the design objectives of providing an (1) introduction to 4GLs and design techniques with 4GLs; (2) in-depth coverage of 4GL and its capabilities in solving common business computer information problems; (3) a practice of the prototyping techniques throughout the entire course; and (4) introduction to the process of building decision support systems for businesses.

Prototyping approach allows one to introduce technical concepts more rapidly, so that students may undertake a programming task at greater depth more quickly. This is of particular importance given the wide scope of material to be covered within the short time of a single semester.

In summary, this training schema brings about:

- -basic understanding of 4GLs to allow one to recognize potential 4GL applications and ascertain justification for their proper use;
- -recognition of the capabilities as well as limitations of 4GLs;
- -change in the way of doing design and programming by utilizing prototyping and 4GLs.

We believe that most important is experience gained by the use of 4GLs: new programmers write and implement applications themselves -- 4GLs are the way of the future.

REFERENCES

1. Bhaisdell, J.H., "Teaching Fourth-Generation Languages Within the CIS Curriculum," ISECON '86 Proceedings, October 1986, pp. 297-301.
2. Chrysler, W., "Some Basic Determinants of Computer Productivity," Communications of the ACM, Vol. 21, No. 6, June 1978, pp. 472-483.
3. Harel, E. C., McLean, E. R., "The Effects of Using a Nonprocedural Computer Language on Programmer Productivity," MIS Quarterly, June 1985, pp. 109-120.
4. Holtz, H., "A Nonprocedural Language for Online Applications," Datamation, Vol. 25, No 4, April 1979, pp. 167-176.
5. Martin, J., "Fourth-Generation Languages, Principles," Prentice-Hall, Inc., 1985.
6. Martin, J., "Fourth-Generation Languages, Representative 4GLs," Prentice-Hall, Inc., 1986.
7. McCracken, D. D., "The Changing Face of Applications Programming," Datamation, Vol. 24, No. 12, November 15, 1978, pp. 25-30.

8. Munnecke, T., "A Linguistic Comparison of MUMPS and COBOL," Proceedings of the National Computer Conference, AFIPS, Montvale, New Jersey, Vol. 49, 1980, pp. 723-729.

9. Perror, R. G., Raja, A. K., and O'Kauet, P. C., "A Simulation Experiment Using Two Languages," The Computer Journal, Vol. 23, No. 2, 1979, pp. 142-146.

10. Prywes, N. S., Pnueli, A., and Shastry, S., "Use of Nonprocedural Specification Language and Associated Program Generator in Software Development," ACM Transactions on Programming Languages and Systems, Vol. 1, No. 2, October 1979, pp. 196-217.

11. Reisner, P., "Human Factors Studies of Database Query Languages: A Survey and Assessment," A.C.M. Computing Surveys, Vol. 13, No. 1, March 1981, pp. 13-31.

12. Shneiderman, B., "Experimental Testing in Programming Languages: Stylistic Consideration and Design Techniques," Proceedings of the National Computer Conference AFIPS, Montvale, New Jersey, Vol. 44, 1975, pp. 653-656.

13. Welty, C., "A Comparison of Procedural and a Nonprocedural Query Language: Syntactic Metrics and Human Factor," unpublished Ph.D. dissertation, University of Massachusetts, Amherst, Massachusetts, May 1979.

14. Welty, C., and Stemple, D., "Human Factors Comparison of a Procedural and a Nonprocedural Query Language," ACM Transactions on Database Systems, Vol. 6, No. 4, December 1981, pp. 626-649.

15. Wojtkowski, W.G., and Wojtkowski, W., "Assessing The Fourth-Generation Language as a Productivity Tool: An Example of Use for the Budgeting System for the Idaho Legislative Office," ISECON '86 Proceedings, October 1986, pp. 307-309.

16. Wojtkowski, W., and Wojtkowski, W.G., "Introduction to Fourth-Generation Language: End-User Approach," ISECON '87 Proceedings, October 1987, submitted.

PREPARING STUDENTS FOR CAREERS IN INFORMATION CENTERS

Barbara Beccue
Carol Chrisman

Applied Computer Science Department
Illinois State University
Normal, Illinois 61761
(309) 438-8338

Abstract

The dramatic increase in the number of companies with Information Centers (ICs) is providing new career opportunities for Information Systems (IS) students. The skills needed for a career in an IC are different from those needed for more traditional data processing careers. A successful IC consultant must possess a combination of communication, interpersonal and technical skills, as well as general business knowledge. This paper will discuss the skills a student should obtain as preparation for a career in an IC and how a University can help the student obtain these skills.

Introduction

Increasing numbers of companies are attempting to support end user computing through ICs. The purpose of an IC is to provide the computer tools and support services that allow end users to effectively and efficiently fulfill their information requirements. ICs will have a significant role in ensuring the proper utilization of an organization's information resources since they provide a bridge between technology and the users.

The number of ICs has increased dramatically over the last decade. The first IC was formed by IBM Canada in 1974. Currently most large organizations have implemented an IC [1,2]. The 1985 American Management Association Report on ICs estimated that 80% of billion-dollar companies and 40% of small businesses have ICs.

Many people are needed to staff these new ICs. In fact, this need has fostered an emerging career as an IC consultant. Companies have attempted to staff these centers from within their data processing organization. From experience with ICs, it has become apparent that the most successful personnel in this setting are not necessarily those that were successful in the traditional data processing environment. The career of IC consultant requires more than just technical skills.

Information Center Services

An IC which provides a full line of services will support the end user in a number of areas such as product use, data access, consulting, and training. To support product use an IC needs to supply information about hardware and software products. This information would include the capabilities and limitations of each product. Most ICs offer additional product support through a hotline to provide troubleshooting help. The data support function addresses

issues involved with the access, extraction, and use of data residing in corporate or centralized data bases. The training services typically include both organized classes and one-on-one teaching sessions. ICs also provide end users with consulting services. They help end users identify problems and information needs and then devise practical ways to solve the problems or acquire the necessary information.

Staff Skills

Since an IC is a service department, its staff is especially critical to the Center's success. In order to offer the variety of services expected of an IC, the staff, individually and collectively, must possess a combination of communication, interpersonal, and technical skills as well as general business knowledge [3,4]. Generally a person with the attributes of both a successful salesperson and teacher would make a good IC consultant.

Communication and interpersonal skills are necessary since the consultant interacts with a wide variety of users at different levels within the organization. To be effective, the consultant must get along well with these people. In addition, the consultant must be able to communicate with them using an appropriate vocabulary and be a good listener so that he hears the underlying needs and fears of the user. The consultant must understand the learning process so that he can begin at the user's level and introduce concepts as the user is ready for them. Training users requires patience and organization. Furthermore, the consultant must be enthusiastic enough to motivate the users.

Technical skills are also important, but the necessary level of expertise is not as high as for communication and interpersonal skills. An understanding of the business functions and the way a particular company operates gives the IC consultant a basis for understanding the users' needs and recognizing appropriate solutions to their problems.

Staffing ICs

Staffing ICs with individuals having the necessary skills is a major concern for companies. Staffing is complicated by the fact that the supply of individuals with appropriate skills is limited and locating these individuals is difficult. Students from Information Systems programs represent a potential source of candidates for IC jobs. Since the emphasis in most Information Systems programs is on technical skills, the students' communication and interpersonal skills are frequently not as strong as needed. With relatively little effort Universities could better prepare a student for a possible career in an IC. This enhancement of needed skills could come from modifications to the Information Systems courses and from advising students to take existing courses from other departments.

Preparing Students

There can be many opportunities within Information Systems courses to provide students with experiences relevant for an IC consultant. Systems development courses already promote an understanding of business through discussions of the feasibility, justification, and functionality of a system.

Working in groups gives students opportunities to develop and practice interpersonal skills. Systems Development courses provide a natural setting for many group activities since students frequently participate in development projects. Students in programming courses can also be required to work in teams on a few assignments. They learn about cooperation and communication through this type of experience.

Group work is only one of the many activities which can be used to strengthen a student's communication skills. Students can make oral presentations and prepare written reports. They can be asked to make explanations to the rest of the class or to a small group of peers. Instructors should make use of available opportunities to have students practice their communication skills.

There are additional class activities that would prove useful to a future IC consultant. For example, creating a users' manual or a reference manual can help a student gain insight into the different needs of novice and experienced users. Incorporating a help feature into a piece of software can encourage a student to try to think like a user. The student would have to determine the type and quantity of information that would assist the user in a variety of situations. In addition to class activities, some students can gain valuable experience from being a tutor, lab debugger, or lab teaching assistant.

With slight changes to courses the emphasis within an IS program can be adjusted to a more appropriate balance of technical and non-technical skills. It will require taking advantage of natural opportunities in the various courses, not major changes to the curriculum or a lessening of technical content.

In addition to modifying Information Systems courses to encourage the development of relevant skills, students could be advised to take courses from other departments. Some courses could be selected for their ability to help the student acquire and strengthen communication and interpersonal skills. Other courses could be selected to help develop an awareness of business functions and practices. Information Systems programs may already require some beginning level courses in speech, business organization, and technical writing. While these requirements provide a start in helping the student develop appropriate skills for a career in ICs, more expertise in the relevant areas is needed.

To demonstrate the types of courses that might be appropriate, specific topic areas will be discussed and specific courses from our University that cover these topics will be listed. Although the courses are specific to our University, most Universities have similar courses. The intent is to create an awareness of the potential courses that could better prepare an Information Systems student for an IC career.

In order to gain knowledge and understanding of the business environment, students can take courses in business. Possible courses from our University include, but are not limited to, the following:

Business and Its Environment
Business Communications
Business Finance

Business Organization and Management
Human Resources Management
Instructional Techniques for Business
Legal Environment of Business
Organizational Behavior and Administration
Principles of Economics

The student can enhance his communication and interpersonal skills through courses typically offered in departments of English and Communication. For example, our University offers the following courses from these two areas:

Interpersonal Communication
Introduction to Interviews
Message Composition
Nonverbal Communication
Organizational and Professional Speaking
Report Writing for Business
Small Group Processes
Speech
Technical Writing
Utilization of Audiovisual Materials

The development of interpersonal skills can be fostered by many courses in the Psychology Department. These courses are often overlooked in advising IS students. Possible psychology courses from our University include:

Applied Human Relations in the Workplace
Dynamics of Social Behavior
Intro to Psychology
Learning
Motivation
Organizational Psychology
Perception
Personnel Psychology
Principles of Behavior Modification

The various courses listed demonstrate some possibilities that could better prepare a student for an IC job. In addition, beneficial courses might be found in other departments such as Education or Sociology. A student who was interested in working in an IC could be advised to minor in one of these relevant areas. For example, a minor in Communications would make a student especially attractive as a prospective employee.

Conclusion

The establishment of ICs has created new career opportunities for Information Systems students. This type of career requires a combination of general business knowledge, communication skills, and interpersonal skills, as well as technical expertise. Universities can help students obtain these skills. They can provide appropriate activities in IS courses and encourage students to take relevant courses in other departments.

References

[1] Arnoudse, D. M. and Ouellette, L. P., "An Introduction to the Information Center Concept", Information Strategy (Winter 1986) Vol 3:2, pp. 9-12.

[2] Greenberg, E., "What Drives IC Growth?", Information Center (July 1986) Vol 2:7, pp.55-59.

[3] Perry, W., "Information Center Staff Selection and Development" in The Information Center, Prentice Hall, 1987, pp. 85-105.

[4] Arnoudse, D. M. and Ouellette, P., "Staffing the Information Center for Long-Term Success", Journal of Information Systems Management (Winter 1986) Vol 3:1, pp. 78-80.

A SURVEY OF A COLLEGE OF BUSINESS' ADVISORY COUNCIL CONCERNING THE DEVELOPMENT OF THE CIS CURRICULUM

by

Dr. Andrew J. Winnick
Associate Professor of Computer Information Systems
and Quantitative Analysis
Department of Business Administration
Central State University
Wilberforce, Ohio

ABSTRACT

During the summer of 1986, as part of a larger curriculum development study for the CIS program at Central State University, I conducted a survey of the Advisory Council of the College of Business to ascertain their views on our current curriculum and their recommendations for needed changes. Since these persons represent organizations, many of which hire our graduates, their views were thought to be a relevant and useful component to the study. The results indicate that there was a clear consensus that the systems design and implementation courses and the database management courses were the most important. COBOL programming was felt to be necessary, but not a very high priority. The introductory course was among the least important. Their recommendations included a focus on the role of microcomputers, the introduction of Fourth Generation Languages, data communications, and oral and written communications. Recommendations were also solicited concerning the best software to use for teaching purposes.

INTRODUCTION

The survey was sent to 30 Advisory Council members, plus the two most frequent employers of our CIS graduates. Altogether, we received 13 responses, for a 40.6% response rate, including: ALCOA, Babcock & Wilcox, Defense Logistics Agency, General Foods, Huntington National Bank, IBM, Mead, NCR, Reynolds & Reynolds, and XEROX. Despite the low response, rate, the pattern of responses was sufficiently clear and distinct as to provide useful information. The respondents were instructed: "In answering all of the following questions, please do so in the context of the criteria you and your organization would be likely to use in the hiring of a new employee with a B.A. degree in Business Administration with an area concentration in Computer Information Systems."

Conceptually, the survey can be divided into three main focus areas: an evaluation of the importance of the courses as we now teach them, suggestions for improving our curriculum, and information about the sort of software applications packages upon which we should be giving our students experience.

EVALUATING OUR CURRENT COURSES

The first question was: "Looking at the list of courses currently offered at CSU, please indicate, in rank order, the ones which are most important," with three numbered lines being provided. The results were:

Course	Rankings First	Second	Third	Total
Introduction	2	0	0	2
COBOL Prog I & II	2	1	2	5
Systems Anal & Design	7	3	1	11
System Design & Implementation	1	6	5	12
Database Management	1	3	4	8
Topics (PASCAL)	0	0	1	1

Thus, the courses in systems development were clearly mentioned most often, with the Database Management course, being the next most often cited. The topics and introductory courses were the least cited. Interestingly, the COBOL programming courses were mentioned only five times.

As checks on the validity and reliability of these answers, the respondents were asked two related questions. First, identify which two courses were the least important (with two unnumbered lines upon which to answer). The results were:

Course	Times Mentioned
Introduction	4
COBOL Prog I & II	5
System Anal & Design	0
System design & Implementation	0
Database Management	2
Topics (PASCAL)	9

The fact that neither of the systems development courses were mentioned even once, and that the database management course was mentioned only twice, confirms the results of the first question. It is again interesting to note that the programming courses were mentioned even more often than the introductory course, though less than the topics course.

Then, the survey asked Please rank each course on the following scale:

1. VERY IMPORTANT that the potential employee have this
2. FAIRLY IMPORTANT ...
3. NOT VERY IMPORTANT ...
4. QUITE UNIMPORTANT ...

No. of Times Ranked at:

Course	1	2	3	4	Average
Introduction	7	3	2	4	1.76
COBOL I	5	4	4	0	1.92
COBOL II	4	5	4	0	2.00
Sys A & D	11	1	1	0	1.23
Sys D & I	10	2	1	0	1.31
Data Mng	9	3	0	1	1.46
Topics	1.5	2.5	5	4	2.58

Again, we find that the system development courses receive the highest rankings, followed closely by the database management course. What is interesting about the rankings of the programming courses is that while they score lower, they are still considered to be relatively important, though less so than the introductory course. Clearly, the topics course, as we are now teaching it, is not considered very important at all.

Thus, our respondents were clearly quite reliable in their answers and the responses are also apparently quite valid, given the different way this question was posed and the consistent pattern of the answers.

RECOMMENDATIONS FOR MODIFICATIONS OR NEW MATERIAL

The next area explored was what new courses the respondents thought we should offer. The first such question was directed in nature and asked: "Using the scale given, how important do you think it is that the potential employee have a hands-on Personal Computer course?" The answers were:

1.	Very Important:	7
2.	Fairly Important:	5
3.	Not Very Important:	1
4.	Quite Unimportant:	0

for an average ranking of 1.54. Clearly, the respondents think such a course should be offered.

They were then asked an open-ended question about new courses: "What other course(s) would you like to see a potential employee have? Please indicate the topic area and the degree of importance you place on this suggestion, using the above mentioned scale.

There was a wide diversity of suggested new courses with only two topic areas occurring frequently. Six different people suggested the need for a course that dealt with data communications. Two used that term, others referred to telecommunications, local area networks, or distributed processing. Another area that was referred to by four of the

respondents was the need to better prepare students in oral and written communications and presentations. No other topic was mentioned more than once. Among those that were mentioned once were office automation, IBM job control language, Fourth Generation Languages (4GL), and basic assembler language. In almost every case, when a suggestion was made it was weighted 1 or 2. It was interesting to me that no one mentioned the topic of Information Center Functions, which was the subject of one of the DPMA's recommendations for a new course.

The second question in this area was: "Looking back over the course descriptions, please indicate ANY changes you would like to see made, either topics added, dropped or emphasized." The primary suggestions were that we introduce into our systems and database management courses the use of 4GL's and that we emphasize data analysis and data architecture. It was also suggested that we make more use of projects wherein the students would actually have to conduct interviews outside the class, work in teams, and develop solution proposals. There were suggestions that we introduce artificial intelligence and expert systems and the use of networks and that we make more use of PC application packages and less use of the VAX.

Then, the question was posed as to: "What courses outside the area of Computer Information Systems do you think are the most important for such a potential employee." The most common responses were courses on human resource management, effective communications and presentations (oral or written), accounting, and philosophy and ethics. When mentioned, all were weighted with either 1 or 2.

SOFTWARE RECOMMENDATIONS

Another focus of the survey was to ascertain how important the respondents considered training in various software application areas, they were asked the following question (rank scores are listed to the right): "How important do you think experience using each of the following types of software is?"

	Rank				
	1	2	3	4	Ave.
Work Processing	3	4	5	1	2.31
Spreadsheets (General Use)	6	3	4	0	1.85
Database Management	10	3	0	0	1.23
Decision Support Systems					
a. Linear Programming	1	3	8	1	2.69
b. Network Models (CPM, PERK, etc.)	0	7	6	0	2.46
Graphics (Charts & Graphs)	1	8	4	0	2.23
Graphics (CAD Packages)	0	5	6	2	2.69
Accounting Statement Design Package	1	3	7	2	2.77
Marketing Survey Packages	1	3	7	2	2.77
Comp. Assis. Production Mgt.	1	3	7	2	2.77
Other than those specified above)					

From the above, it is clear that the primary concern was that students have experience in the use of database management and spreadsheet software application packages, followed by a lesser concern that they are experienced with word processing and graphics (charts and graphs). Among the other options, only Network Models received more rankings of 1 or 2 than of 3 and 4.

Then, I wanted to find out if the respondents had specific software packages upon which they wanted our students trained. With one or two exceptions, recommendations were made only with regard to word processing, spreadsheets, and database management. Even here, two respondents did not make any specific recommendations, one of them stating that: "The version is not important--opportunity to have in-depth use is important." Among those who did respond, the pattern was fairly clear. Two word processing systems were mentioned most often: Wordstar - 5 times, and Displaywrite - 4 times. No other system was mentioned more than once. In the case of the spreadsheets, there was amazing unanimity, everyone responding mentioned Lotus 1-2-3. Excell and 20/20 were also each mentioned once. There was little agreement on the database management system. One or

another version of dBase received four mentions, with IDMS, Oricle, Nomad and IBM's IMS each being mentioned once.

Then, they were asked which software does your organization primarily use for each of the specified purposes. There was even less consensus at this point. Clearly most of the firms use more than one of each type of software, many use three or more. What this says about the compatibility of their databases and their ability to move data around the firm is rather interesting. But the relevant point for our analysis is that students will have to be able to adapt to a variety of different software packages. This confirms the point made that the key is that they have some in-depth, hands-on experience with at least one typical package in each of the key areas: spreadsheets, databases, word processing and graphics. After that, learning a new system is not a major problem.

Finally, they were asked: "Do your lower and middle managers have regular access to a personal computer? Do those personnel tend to have direct access themselves to a mini or main frame computer?" Every respondent indicated that the answer to the first question was yes and in all but one case mentioned an IBM compatible system. Moreover, almost everyone answered yes to the second part of the question, but with a much greater variety of hardware systems mentioned.

BUSINESS/EDUCATION COLLABORATION: COMPETENCIES RANKED BY CIS PROFESSIONALS

Dr. Thomas A. Pollack
Dr. John C. Shepherd

Duquesne University
School of Business and Administration
Pittsburgh, PA 15282

ABSTRACT

The objective of this study is to determine the perceived importance of various identified competencies inherent in a typical undergraduate Computer Information Systems (CIS) program of studies. Information systems professionals were surveyed and asked to rank the importance of competencies developed by most CIS programs. A major finding of this study is that business professionals perceive written and oral communication skills as the most important competencies for an entry level person to possess. In the technical area, the information systems professionals ranked proficiency in COBOL programming and structured programming techniques as the most essential competencies. The audience of survey respondents was least interested in EDP Auditing.

INTRODUCTION

Educators from schools of business administration have typically expressed concern for development of competencies desired by businesses which will ultimately employ their graduates. This attitude is often contrary to that of fellow educators from the typical college of arts and sciences who tend to ignore the vocational orientation and advocate education for the sake of education. Even so, the business community, for many years, has contended that schools of business do not promote and teach the competencies which the corporate community deems vital. Curricula that an academician perceives as emphasizing fundamentals often appears as being out of touch with the realities of industry jobs to the information systems professionals (3, p. 68).

The DPMA Model Curriculum is intended to be a resource document which is both dynamic and responsive (2, p. 3). A significant aspect of the responsiveness of programs of study in the computer information systems field is the direct result of business/university collaboration. This business/university collaboration can result in excellent community relations for the corporations and desirable skills, attitudes, and motivation on the part of the students (4, pp. 63-64). In a recent Datamation article by Curt Hartog, it was stated that current undergraduate students are the most career-oriented of any group since the 1950's (3, p. 68). Since the highly publicized report, "A Nation at Risk", the corporate community has become more actively involved in a partnership role with education. They have come to the realization that quality education is an investment in their work force, their companies, and in general, the nation's economy (1, p. 16). The corporate sector has also recognized that they can either train employees after they are hired or be somewhat influential in training them before they are hired. Partnerships with schools prove to be much less costly than post-hiring training programs and yield maximum use of resources for schools and corporations (4, p. 65).

PROCEDURE

A questionnaire listing CIS/MIS competencies was designed and distributed to information systems professionals through organization meetings of the Pittsburgh Chapters of the Association for Systems Management and the Data Processing Management Association. The cooperation of those holding management or recruiting positions was sought. Thirty valid responses were returned, and the results were compiled. The response rate was approximately 25% of the number of questionnaires distributed. This figure is misleading in that the questionnaire was not selectively distributed, but instead a specific verbal request asked those holding management or recruiting positions to complete the questionnaire.

THE QUESTIONNAIRE

In designing the questionnaire an effort was made to extrapolate as many general competencies as possible from the DPMA Model Curriculum (2, pp. 6-29). The questionnaire was actually divided into three segments. In the first segment, the respondents were asked to evaluate on a scale of one to five, where one is least important and five is most important, a list of fifteen general CIS/MIS competencies. An open space was provided for the respondent to identify competencies deemed important, but not listed.

In the second segment of the questionnaire, the respondents were asked to evaluate specific areas of CIS/MIS. Included in this section, one will find categories for programming languages, systems software, database management systems, statistical packages, fourth generation languages, artificial intelligence languages, financial modeling software, and microcomputer software. In addition to specifically designated items, each of the above areas had a space for the respondent to identify software of importance which was not listed.

In the third segment of the questionnaire, the general business and other remaining educational requirements typical of an AACSB-accredited business school program were listed. Again, the respondents were asked to assess the importance of each of these areas. Space was provided for the respondent to identify areas deemed important but not listed specifically.

The specific numerical scale used to rank each competency is as follows:

1. Least Important
2. Moderately Important
3. Important
4. Very Important
5. Most Important

A response summary, by rank order, of the average rating for each competency listed on the questionnaire follows:

General MIS Competencies	Mean Rating
1. Structured Programming Techniques	4.1
2. Documentation and Maintenance	3.8
3. File Organization and Design	3.8
4. Data Communications	3.7
5. Structured Analysis and Design	3.7
6. Database Management Systems	3.6
7. Fourth Generation Languages	3.1
8. Decision Support Systems	2.9
9. Job Control Language (JCL)	2.9
10. Microcomputer Operating Systems	2.9
11. Mainframe Operating Systems	2.8
12. Artificial Intelligence/Expert Systems	2.4
13. Computer Graphics	2.3
14. Computer Simulation Models	2.3
15. EDP Auditing	2.2

Programming Languages	
1. COBOL	4.3
2. BASIC	2.8
3. Mainframe Assembler	2.4
4. C	2.3
5. FORTRAN	2.3

Systems Software	
1. MVS-TSO	3.7
2. CICS	3.3
3. LAN System Software	3.3
4 UNIX	2.6
5. CMS	2.3

Data Base Management Systems	
1. SQL/DB 2	3.2
2. IMS	2.9
3. IDMS	2.5

Statistical Packages	Mean Rating
1. SAS	2.8
2. SPSS	1.9
Fourth Generation Languages	
1. FOCUS	2.9
2. NOMAD 2	2.1
Artificial Intelligence Languages	
1. LISP	2.6
2. PROLOG	2.5
Financial Modeling	
1. IFPS	2.4
2. PSG	1.8
Microcomputer Software	
1. dBASE III PLUS	3.8
2. Lotus 1-2-3	3.7
3. Word Processing	3.3
Other Educational Competencies	
1. Written Communication	4.4
2. Oral Communication	4.3
3. Interpersonal Relationships	3.8
4. Management Principles	3.5
5. General Business Theory	3.4
6. Liberal Arts Core	3.3
7. Organizational Behavior	3.3
8 Group Dynamics	3.1
9. Accounting Principles	2.9
10. Finance Principles	2.7
11. Marketing Principles	2.4
12. Production Management	2.1
Top 12 - All Categories	
1. Written Communication	4.4
2. COBOL	4.3
3 Oral Communication	4.3
4. Structured Programming Techniques	4.1
5. dBASE III PLUS	3.8
6. Documentation and Maintenance	3.8
7. File Organization and Design	3.8
8. Interpersonal Relationships	3.8
9. Data Communications	3.7
10. Lotus 1-2-3	3.7
11. MVS-TSO	3.7
12. Structured Analysis and Design	3.7

FINDINGS, IMPLICATIONS, AND INTERPRETATIONS

The study has basically confirmed that long-established fundamental CIS competencies remain important as potential entry level employees are assessed. It is not surprising that business professionals feel that written and oral communication skills and interpersonal relationship skills are extremely important. Interestingly, Hartog reported that although technical preparation is good, leadership, communication skills, and business knowledge are generally lacking in the typical computer science graduate (3, p. 70). It is also not surprising that the respondents deem competencies in structured programming techniques, documentation and maintenance, database management systems, file organization and design, and data communications as being very important. This general group of competencies has been considered essential in CIS education programs for some time (5, pp. 152-153). This group of competencies also enables an entry level employee to contribute immediately to his/her organization in a productive fashion.

In the area of specific software, COBOL remains as the dominant programming language in the business environment. Somewhat surprising is the fact that SQL/DB 2 and FOCUS were considered most important in the database management systems and fourth generation language categories, respectively. LISP is the most desirable of the artificial intelligence languages. There were absolutely no surprises in the microcomputer software category. LOTUS 1-2-3 and dBASE III PLUS are the most desirable of the microcomputer software packages. Feelings are mixed on the choice of a specific microcomputer word processing package, but WORDPERFECT seems to be emerging as the favorite in this area. Perhaps the most surprising and somewhat disappointing aspect revealed by this study is the lack of a forward, futuristic look at the CIS/MIS field and the competencies which will be especially important in the years ahead. Few respondents chose to identify other competen-

cies or specific software of importance. The results presented in this paper are somewhat limited due to several factors. First of all, the questionnaire was not distributed selectively. Greater control as well as a higher response rate might be achieved by strictly controlling distribution of the questionnaire to the desired audience. Primarily, the authors were interested in the opinions of managers and recruiters of CIS/MIS professionals. This qualification for respondents was delivered verbally when the questionnaire was distributed. Secondly, validity will be improved with a larger sample size and subsequent re-evaluation of the average ratings. The authors do not feel that the sample size is large enough to validate by hypothesis testing at this time. Thirdly, the ranked needs may differ significantly in another geographic area or with another group of information processing managers. Fourthly, ranked needs may differ by the type of equipment which predominates in a given geographic area.

It appears that the academic community, based on research being done in leading CIS/MIS schools, must assume the lead in anticipating which competencies will be of particular importance in the future. It is relatively safe to assume that significant advances in such areas as fourth generation languages, artificial intelligence, expert systems, and decision support systems will represent the new direction for the information systems field. Leading institutions will predominate in developing competencies in these and other areas, and their graduates will prosper in the job market as a result. It is extremely important to be both current and progressive in program development and revision efforts. Successful programs in CIS/MIS will continue to place emphasis on the ever so important area of developing logical-thinking skills, sound problem-solving techniques, and effective communication skills. As CIS/MIS educators, we can be assured that our graduates will be successful if they possess the always important ability to methodically assess and solve a problem.

REFERENCES

1. Armistead, Lew and Martin, Elizabeth M., "Business/Education Partnerships," Public Relations Journal, September 1985, pp. 16-20.

2. CIS '86 "The DPMA Model Curriculum for Undergraduate Computer Information Systems," Second Edition, Data Processing Management Association, July 1986.

3. Hartog, Curt, "Of Commerce and Academic," Datamation , September 1, 1985, pp. 68-70.

4. Jasso, Gayle, "A Partnership with Schools To Train Job Prospects," Personnel, May-June 1984, pp. 63-68.

5. Pollack, Thomas A. "The Rationale, Design, and Assessment of a Business Information Systems Curriculum to Fulfill the Needs of Large Computer Users," Dissertation, University of Pittsburgh, 1981.

ASSESSING THE QUALITY OF AN APPLIED COMPUTER SCIENCE PROGRAM FOR PROGRAM REVIEW AND CURRICULUM DEVELOPMENT

David F. Kephart
Applied Computer Science Department
Illinois State University
Normal, Illinois 61761
(309) 438-8338

Abstract

The Applied Computer Science (ACS) Department was created in the Fall of 1978 to provide graduates who could supply computer solutions to business problems. The department differed from a traditional computer science department in that curriculum and research were application driven with many of the courses based on the DPMA Model Curriculum. The department has to report on the effectiveness of it's program to a state agency every five years. Also because of this orientation, the ACS Department has to maintain curriculum that reflects the 'state of the art' in computer applications in business and industry. One method of determining the quality of it's curriculum and what was happening in the real world of computer applications was to survey ACS graduates.

Introduction

In the Fall semester of 1985 the author was asked to develop a survey of graduates since 1980 for the Applied Computer Science Department. The objectives of the survey were to aid in the curriculum development process at both the undergraduate and graduate level and to provide input into a periodic program review process required by the governing board of the university.

Questions raised by the Curriculum Committee during the past two years concerning the direction that the curriculum should be going have included:
"Should we change our IBM mainframe, COBOL emphasis?"
"Should we increase the emphasis on microcomputer applications?"
"Is it important for graduates of our program to have a course in Assembler language?"
"Is IBM CICS Command Level language on the wane?"
"Is our curriculum conceptual enough to provide our graduates with the knowledge they will need to grow and advance in their careers?"
Questions concerning the quality of the Applied Computer Science curriculum have been:
"How well are our graduates doing in finding a job in the computer applications area after graduation?"
"Is our program preparing our graduates for the long term?"
"What kinds of skills do the ACS graduates possess?"
"How does the ACS program compare to programs of a similar nature?"

The Survey Instrument

The survey form was divided into seven parts. The parts were concerned with demographics, first job, career advancement, current job, cooperative education experiences, further education experiences, educational experiences in the Applied Computer Science program, strong points of the

ACS curriculum, and perceived deficiencies in the ACS curriculum.

The Procedure

The surveys were mailed in November 1985, to all known graduates of the Applied Computer Science Department, which numbered 535. A total of 196 responses were received during the spring 1986 semester. The data was coded, recorded, and analyzed during the summer and fall of 1986. Since some questions required a write-in response, the author had to categorize responses based on his knowledge of the ACS program and computer career fields.

The Results

Background of the respondents was determined by job classification and primary job duties. Of 196 respondents, 193 were working in a computer related field with the majority (127) describing their primary job function as either programmer, programmer/analyst, or systems analyst/designer. The rest were in computer operations, computer management, project management, data base management, or data processing training.

Several questions addressed program quality. The graduate was asked to check the appropriate answer. The questions and responses are listed below.

23. How well did your undergraduate studies prepare you for your job?

RESPONSE	COUNT
1. Good	141
2. Adequate	47
3. Inadequate	4

These responses indicate that the ACS program provided the necessary knowledge and skills to prepare the graduates for a career in computer applications. The next question was designed to determine if long term knowledge and skills were being provided by the program.

24. Please indicate how well the following phrases reflect the continuing usefulness of your undergraduate course work in Applied Computer Science.

RESPONSE	COUNT
1. Still useful on the job	122
2. Replaced by specialized training for present job	101
3. No longer relevant to work but contributes to quality of life	75
4. Enabled me to continue to upgrade my skills to their present level	19

Again it appears that the ACS program has been addressing the long term skills that will allow the graduates to continue to be successful in a computer applications career.

The purpose of question 40 was to assess the short term preparedness of an ACS graduate.

40. How well were you prepared for work after completing the ACS curriculum?

RESPONSE	COUNT
1. Completely prepared	36
2. Very prepared	128
3. Somewhat prepared	30
4. Not very prepared	2
5. Completely unprepared	0

The responses indicate that ACS graduates were well prepared to compete in the job market upon graduation. This correlates with the job placement rate of graduates.

The next question was designed to compare the Applied computer Science program against similar programs.

47. How well would you rate yourself in comparison to co-workers from other schools or training backgrounds?

RESPONSE	COUNT
1. Much better	78
2. Somewhat better	83
3. Equal	33
4. Somewhat worse	0
5. Much worse	0

The responses indicate that the Applied Computer Science program is as good or better than other programs that prepare graduates for a career in computer applications.

The next set of questions address the curriculum development issues.

21. What processing equipment do you work with?

A total of 162 responses were IBM or IBM plug compatible mainframe systems.

This answers the question addressing the issue of whether or not to reduce the IBM orientation of our courses. While there is a bias to the responses, that is, we would expect organizations that have IBM hardware to recruit graduates of an IBM oriented program, it appears that a hefty segment of the ACS graduates job market will continue to be IBM shops.

The next question deals with specific courses in the undergraduate curriculum. I have provided the generic name for the courses specified. This question was open-ended so a respondent could list more than one course.

41. What courses do you feel were the most important in achieving success in your chosen area of work?

COURSE	COUNT
External Data Structures	145
COBOL Language	96
Systems Analysis and Design	93
Software Design Methodologies	82
On-line System Design	72
Assembler Language	45
Data Base Management Systems	44
Microsystems	13

Since our Applied Computer Science program has been large scale IBM COBOL oriented, it was not surprising that the external data structures course was indicated so many times. The course covers IBM utilities, job control language, and VSAM programming with a strong background in COBOL being a prerequisite for the course.

The number of responses for on-line system design indicate that a significant number of the graduates are working with on-line systems.

On the question of long term preparation of the graduates, the responses for data base management systems, systems analysis and design, and software design methodologies seem to indicate that these courses improve job success even though they are not central to the job.

Only nine respondents classified their job function as systems programmer and seven more as assembler language programmers. The number of responses seems to indicate that there is some value to understanding assembler beyond just using it on the job.

The small number of respondents who selected micros as one of the most important courses

correlates with the answer to a subsequent question on the least valuable courses taken.

A question dealing with the least important course was also asked.

42. What ACS courses to you feel were least important for achieving success in your chosen area of work?

COURSE	COUNT
Microsystems	45
PL/I language	44
Systems design	25
Systems analysis	12
Assembler	9
COBOL	9

These were the only courses that had more than one or two responses. Again, there was an indication that our graduates were not heavily involved in microsystems.

Outcomes of an academic program are important. Two questions dealt with specific skills attained in the ACS program.

12. Please check all appropriate columns to indicate whether you acquired this skill in undergraduate course work.

RESPONSE	COUNT
Knowledge of subject matter	175
Critical thinking skills	160
Research and investigative skills	137
Ability to learn new ideas/skills	167
Management and administrative skills	67
Interpersonal skills	135
Oral communication skills	144
Written communication skills	144
Design and planning skills	177
Information management skills	115

If a major objective of any four year degree program is to prepare students for life, it would seem that the Applied Computer Science program students for life, it would seem that the Applied Computer Science program is successful. It has been the author's contention that the ACS program produces problem solvers and communicators with the technical skills to deal with computer problems and solutions. The responses to question twelve seem to reinforce this statement.

The second question on program outcomes required a write in answer.

43. What was the major benefit you gained from your course work in Applied Computer Science?
The two primary responses were "Acquisition of technical skills" with 117 and "Problem solving/logical thinking skills" with 41.

Conclusions

Given limited resources, it is important for an academic program to be able to concentrate on those courses and computing resources that will be the most beneficial to the graduates. A survey of graduates can be one significant source of data about program outcomes and curriculum effectiveness. Based on the results of this survey, the Applied Computer Science program and curriculum have been producing graduates who have problem solving skills, can communicate, and can apply computer solutions to real world problems. It would seem that the large scale IBM COBOL orientation of the curriculum has been appropriate and that it will be appropriate for the near future.

THE DEVELOPMENT OF AN UNDERGRADUATE BUSINESS AI COURSE

Norman E. Sondak and Richard A. Hatch
Information and Decision Systems Department
College of Business Administration
San Diego State University
San Diego, CA 92182-0127

ABSTRACT

Until recently the topic of Artificial Intelligence has been the sole purview of Computer Science departments at colleges and universities. However, the development of Expert Systems has changed all that. Major curricula bodies, such as the DPMA, have indicated that AI course work be a required part of the education of information systems graduates. This paper contrasts the type of Artificial Intelligence course available in the computer science area with the needs and demands for AI education of the business student. An undergraduate course AI designed for Information Systems majors at San Diego State University is presented. Literature and text support for such a course is reviewed. The available AI courseware is examined and discussed.

AI AND DATA PROCESSING

Artificial Intelligence, as a discipline, has its roots in academia. The very phrase "Artificial Intelligence" was coined as a result of the Dartmouth Conference of 1956 [1]. AI was nurtured and developed at major university laboratories in MIT, Stanford, Cal Berkeley, Carnegie-Mellon, and Yale. AI courses and seminars were available in computer science departments at most colleges and universities. But, with the development of MYCIN [2] in 1973 there was a fundamental and profound change in the AI milieu.

The first true "Expert System," MYCIN was the direct descendent of DENDRAL, a system for identifying chemical structures. MYCIN was designed to act as an intelligent consultant to physicians, advising them on the diagnosis and treatment of bacterial infections. Within a short period of time, the enormous potential of rule-based expert systems was recognized by computer scientists and there was a "transformation in the field of AI." [3] The development of a number of commercially successful Expert Systems in the '80s caused the business community, which had considered AI a computer science toy, to rethink its attitudes towards AI.

This engendered a new appreciation of the relationship of AI and traditional data processing. And recently the curricula committees of professional computer and data processing societies, such as the DPMA and the ACM, issued recommendations that AI education be required in college and university programs designed to educate computer professionals [4,5].

Information Systems departments are now in a much better position to support undergraduate AI education. There are two important reasons for this. First, there is now a strong body of literature and a number of excellent texts available for business-oriented AI courses at the college level. And second, and possibly even more important, AI courseware and programs are beginning to appear that can be used to give a business student the experiential and project component required in an AI course. This courseware is targeted towards personal and desktop computers.

TRADITIONAL AI COURSES

The traditional AI computer science course is based on ACM (Association for Computing Machinery) recommendations [4]. It is offered at the senior level and requires a considerable number of mathematics prerequisites. The course coverage is approximately as follows [4]:

1. Representation of problems and problem domains. The representational capabilities of programming languages and logical systems.

2. Searches and Searching. Search strategies for exploring alternatives. Tree searching algorithms. Finding the best path. Heuristic approaches.

3. Control choices. Knowledge storage, serial and parallel processing modes. Production oriented systems.

4. Natural language processing. Understanding language, parsing sentences, grammars. Thematic-role frames.

5. Computer Vision. Image recognition. Surface and edge determination. Robotics.

6. AI Applications. Expert Systems, robots, computer assisted instruction, medical applications.

The programming component of the course is LISP-oriented. Programming activities include the implementation of search algorithms, game-playing programs, and the actual development of all or portions of an Expert System [7]. PROLOG [11] has recently become a substantive alternative to LISP in these AI courses.

AI IN THE BUSINESS SCHOOL

AI education in the Business School necessitates a restructuring of course priorities and pedagogy [6]. The business students do not have the mathematical or programming backgrounds of computer science students nor do they have the need to examine the AI topics in such technical depth.

A more meaningful philosophy for AI courses offered in the business schools is to have the student master the underlying conceptual framework of AI, not the implementation details. The student should be able to utilize AI concepts in the solution of practical business problems. The course goal for the student is to be "AI Literate" and capable of understanding the relationship of AI, data processing, and decision systems, and to be able to manage AI projects.

THE UNDERGRADUATE BUSINESS AI COURSE AT SAN DIEGO STATE UNIVERSITY

During the 1985-86 academic year, the Information and Decision Systems Department of the College of Business Administration at San Diego State University undertook a major curriculum reevaluation. The curriculum recommendations of the DPMA were studied, as well as offerings and degree requirements at similar institutions throughout the country.

As a result of this analysis, it was determined that an undergraduate course in AI should be part of the core (required) courses for the Information Systems degree and a graduate AI seminar be added to the graduate offerings. A syllabus was developed (by one of the authors of this paper) for the course and after the appropriate departmental, college, and university review (and revision) approved for inclusion in the University Catalog and offering to students.

The catalog description of the undergraduate course is as follows:

Artificial Intelligence Applications In Business -- Basic concepts of Artificial Intelligence, knowledge acquisition and representation, automated problem solving and goal seeking techniques, application of Artificial Intelligence in business, expert systems, data processing versus Artificial Intelligence methodologies.

A brief course outline with approximate course weights is:

1. History of AI, notion of "intelligent" computer applications, the potential for the use of AI in business situations ... 10%

2. AI terminology, human versus machine problem solving, machine acquisition and storage of knowledge, matching and goal reduction, tree structures and search techniques, heuristics ... 25%

3. Expert systems, productions, control strategies, commercial software and packages, list processing, logic programming ... 25%

4. Knowledge engineering, identifying the problem, objects and domain, creating representational theory, structural hierarchy, deep and shallow knowledge, implementing knowledge, decisions and uncertainty ... 20%

5. Building knowledge systems for business, case studies, management models and control, integrating AI in the data processing environment, AI in the Information Center ... 20%

Selected texts and references for the course include:

1. Designing and Programing Personal Expert Systems, Townsend, C., and D. Feucht, TAB Books, 1986

2. A Guide to Expert Systems in Business, Donald Waterman, Addison-Wesley, 1986

3. Rule-Based Expert Systems, Buchanan, B., and E. Shortliffe, Addison-Wesley, 1984

4. The Cognitive Computer, Schank, R., and P. Childers, Addison-Wesley, 1984

5. Artificial Intelligence, 2nd Ed., Patrick Winston, Addison-Wesley, 1984

6. Expert Systems, Harmon, P., and D. King, John Wiley, 1985

7. Introduction to Expert Systems, Peter Jackson, Addison-Wesley, 1986

8. Artificial Intelligence, Tools, Techniques, and Applications, O'Shea, T., and M. Eisenstadt

9. The Fifth Generation, Feigenbaum E. and McCorduck, P., Addison-Wesley, 1983

10. Understanding Artificial Intelligence Mishoff, N.F., Texas Instruments, Dallas, TX, 1985

11. Artificial Intelligence 2nd Edition, Winston, P., Addison-Wesley, 1983

12. Intelligence: The Eye, the Brain, and the Computer, Fischler, M., and Firschein, O., Addison-Wesley, 1987

13. What Computers Can't Do, Dreyfus, H., Harper & Row, 1979

14. Artificial Intelligence: A Personal, Commonsense Journey, Arnold, W., and Bowie, J. S., Prentice-Hall, 1986

15. Texas Instruments Engineering Journal, AI Issue, v3 n1, January-Feburary 1986

16. IEEE EXPERT, IEEE Magazine devoted to Intelligent Systems and their Applications -- Started in 1986, ISSN 0885-9000

17. COMPUTER, Expert Systems Issue, July 1986

18. Zeide, J., and Liebowitz, J., "Using Expert Systems: The Legal Perspective," Expert, v2 n1, Spr 1987, p19-21

19. Williams, C., "Expert Systems, Knowledge Engineering, and AI Tools -- An Overview," Expert, v1 n4, Wntr 1986, p66-70

20. Myers, W., "Introduction to Expert Systems," Expert, v1 n2, Srg 1986, p100-109

21. Expert Systems: Principles and Case Studies, Forsyth, ed., Chapman and Hall, 1984

22. Microexpert, Thompson, R., and Thompson, W., McGraw-Hill, 1985

23. The AI Business: The Commercial Uses of Artificial Intelligence, Winston, P., and Prendergast, K., eds., MIT Press 1984

24. Artificial Intelligence: An Applications-oriented Approach, Schutzer, D., Van Nostrand Reinhold, 1987

The grading standards for the course are:

Projects and presentations	25%
Examinations	50%
Participation	25%

AI COURSEWARE

The experiential component is a key element of this course. Business-oriented AI courses can introduce PROLOG, but they also require an expert shell to complete the experiential component of the course. The shell should operate on a personal computer since many senior Information System students already have their own PCs. Therefore they can complete their course assignments at home and not impact the university's computer system [8].

There are several powerful expert system shells available for personal computers including GURU [12], Personal Consultant [13], M.1 [14], Expert Ease [17], and EXSYS [18]. There are also several lower-cost shells such as Micro Expert [15] and ESIE [16]. VP-EXPERT, an expert system shell by Paperback Software, a company dedicated to offering professional level software at a low cost, is also being advertised.

There is an emerging trend in the software industry to offer very low-cost "student" or "academic" versions of large-scale commercial software. It is expected that student versions of the powerful expert system shells will soon be available too. Such student AI software will play a vital role in establishing business AI courses, provided that these versions have sufficient capacity to handle reasonably sized knowledge-bases.

SUMMARY

Business has finally realized that AI is more than a research curiosity, it is a viable part of the data processing scene. AI courses, once only found in computer science departments, are now appearing in the business school. Such courses differ markedly from those in computer science, in that they focus on "AI Literacy" and rely on the use of expert system shells as the experiential and laboratory component. There is still a need for low-cost, powerful AI courseware. However, there is every reason to believe that the major shell developers will follow the example of other commercial software vendors and offer very low-cost, but still powerful, student versions of their software to support AI courses.

BIBLIOGRAPHY

[1] Mishoff, N.F., Understanding Artificial Intelligence Texas Instruments, Dallas, TX, 1985, p31-36
[2] Buchanan, B., and Shortliffe, E., Eds., Rule-Based Expert Systems, Addison-Wesley, 1984
[3] ibid., page xi
[4] Artificial Intelligence Course, CS 12, Austing et. al., CACM, Mar 1979, p147-66
[5] The DPMA Model Curriculum for Undergraduate Computer Information Systems (CIS 86), DPMA 1986
[6] Eliot, L., "Expert Systems in a Graduate-Level Course: A Business Oriented Approach," AIDS Conference, 1986, p22-24
[7] Bahill, T., and Ferall, W., "Teaching an Introductory Course in Expert Systems," EXPERT, v1n4, Wntr 1986, p59-63
[8] Georgeff, M., and Gupta, G., "Resources Needed for a University Computer Science Department", The Computer Journal, v29 n1, 1986, p90-95
[8] Hong S. J., "Expert Systems: Guest Editor's Introduction," Computer, June 1986, p12-15
[10] Faught, W., "Applications of AI in Engineering," Computer, June 1986, p17-27
[11] TURBO PROLOG, Borland International, 4585 Scotts Valley Drive, Scotts Valley, CA 95066
[12] GURU, MDBS, PO Box 248, Lafayette, IN 47902
[13] Personal Consultant Plus, Texas Instruments, PO Box 809063, Dallas, Tx 75380
[14] M.1, Teknowledge, 525 University Ave, Suite 200, Palo Alto, CA 94301
[15] Micro Expert, McGraw-Hill, NY, NY
[16] ESIE, Lightwave Consultants, P.O. Box 290539, Tampa, FL 33617
[17] Expert Ease, Human Edge Software, 2445 Faber Place, Palo Alto, CA 94303
[18] EXSYS, EXSYS, PO Box 75158, Contract Station 14, Albuquerque, NM 87194

THEORY AND PRACTICE: BRIDGING THE GAP FOR EFFECTIVE TEACHING

James D. McKeen
School of Business, Queen's University
Kingston, Ontario, Canada, K7L 3N6

This paper describes an approach used to teach the capstone MIS course at a major Canadian Business School. By effectively bridging the gap between theory and practice, the approach has resulted in increased student motivation, enhanced classroom participation and top teaching and course evaluations. Even participating organizations are enthusiastic about their involvement with the MBA students enrolled in the course.

THE COURSE

The MBA program at Queen's University has a general management orientation. As a result, MBA students may take elective courses in IS beyond their compulsory introductory IS course but are not granted degrees in IS per se. Because of this orientation, the final capstone IS course deals with "management issues in IS" in order to attract a wider audience than the more typical "management of IS" course. The course is offered to appeal to general managers and not just those intending to be IS professionals.

The course investigates issues relevant to managers who will be assuming positions in organizations where they will have to deal effectively with information technology in all its facets. It is expected that these managers will develop the capability to address technological problems/opportunities with the same business acumen that they would bring to bear in other (eg., accounting or marketing) situations. It is assumed that these managers will interact with this technology as end-users, as members of project development teams, as members of steering committees, in liaison roles with the IS department, and possibly as members of special task forces to investigate potential strategic uses of existing and emergent technologies. In order to be effective in these roles, managers need to have an understanding of (1) the relevant technologies (eg., hardware, software, communications), (2) usage of these technologies (eg. networks, systems, distributed data processing), (3) organizational issues (eg. centralization/decentralization, structure of the IS department, reporting role of CIO), (4) IS development and application (eg., SDLC methodologies, prototyping, end-user development), and (5) strategic issues (eg., IS planning, IS as competitive weapon, personnel recruitment/ development). The course was designed to fulfil this need.

The Business School is on the semester system; all courses last 13 weeks with 3 class hours per week. The course uses a readings package because of the particular blend of topics, the currency of the issues, and the shortage of suitable textbooks.

THE APPROACH

Students taking the course are formed into groups of three; each group selects a company to work with for the duration of the course. Selections are made from a list of organizations which have agreed to participate in the course. The company contact is the CIO (or equivalent under a different title). These contacts are predominantly vice-

presidential level. The companies are medium to large organizations representing a variety of industries. Company selection is on the basis of their willingness to participate with the course - there are no other criteria.

Approximately two weeks of class time are devoted to the discussion of each major topic. This discussion concentrates on existing theory and conceptual foundations. At the end of this two-week session, students visit their companies to analyze how these companies have approached this particular issue. The results of this analysis are written up as a report and presented to the class. This cycle of two weeks in the classroom followed by company analysis and report is repeated five times.

This approach offers a balanced mix of theory and practice by extending the learning environment beyond the walls of the classroom. In a sense, the students take the theory from the classroom and "test" it in the workplace. Sometimes the students return to the classroom with the successful application of the theory. Sometimes they encounter difficulty in applying the theory because (1) they really didn't understand the theory well enough to apply it, (2) they encounter a situation where the theory does not seem to apply, or (3) they discover a situation where the theory breaks down. In all cases, the experience of application benefits the learning process immeasurably.

In addition to the five reports already described, the student groups must prepare a major report at the end of the semester. It is accompanied by an oral presentation as well. This report takes the form of a "management audit" of the IS function at the students' particular company. The student's report must follow a strict format consisting of the following parts:

(1) an intra-company analysis based on the results of the five previous reports

(2) an inter-company analysis comparing the group's company with at least two other companies in the sample (again based on material covered by the five reports)

(3) recommendations to the senior officers of the company for the enhancement of their IS function

(4) a detailed implementation plan for the above recommendations and

(5) a description of the expected benefits to be gained through the implementation of the recommendations.

There are two primary bases for reward on these final reports -- the integration of course materials and the demonstrated ability to support the stated recommendations. This ensures that the students use this opportunity to pull the entire course together. The fact that most of the participating companies request to see these final reports provides a little extra incentive for the students to be conscientious. The grading structure for the course is as follows:

. Five reports (each 10%) ---- 50%
. Final report ---------- 25%
. Course participation ------- 25%
100%

COURSE BENEFITS

The format for this course has evolved over a number of years. During that evolution, different approaches have been used including cases (instead of real companies), textbooks (instead of readings packages), lectures (instead of seminars), guest speakers and even organized debates on specific topics. All of these approaches have advantages in certain situations but they must be coordinated into a consistent and cohesive framework. The approach described in this paper has such a framework and, as such, derives some definite benefits. These benefits are

outlined below.

Meaningful participation. Within a few weeks of the beginning of the semester, MBA students become very knowledgeable about their companies and certainly know much more than the other students. In some cases, they may know more than the professor. That is, they soon become the authority regarding their particular company. Furthermore, other students expect each group to assume this role. This makes the classroom discussion come alive with "meaningful participation" as opposed to participation based on a cursory analysis of an abridged case. The students have a "stake" in these companies. Discussions can become very heated. One example of this occurred where one company decided to abolish their chargeback scheme for exactly the same reason that another company adopted it! These flagrant inconsistencies generate the motivation for students to seek resolutions of the issue. It is interesting to see the students develop ways of defending and indeed justifying the actions of their companies. They do not remain disinterested observers for very long!

In-depth Learning. It is one thing to understand a theory; it is quite another to understand it well enough to be able to apply it. It is in the application that one really learns the theory -- its strengths, its weaknesses, and its applicability. Through the design and operation of this course, the theory must become an analysis tool understood well enough to be applied.

Deductive/Inductive Learning. Starting with a theory and discussing where and how it might apply is deductive. In contrast, starting with practice and attempting to generalize to a common statement is inductive. The approach adopted in this course balances theory with practice, induction with deduction. It is felt that the blend of these polemic approaches towards investigation offers the greatest insight for the students. Where the theory is lacking or underdeveloped, use is made of more inductive strategies by students; where the theory is well developed and accepted, students use more of a deductive approach to the analysis of their organizations.

Multi-dimensional issue analysis. If this course is taught using the case method, a single case is discussed for each management issue (eg., the "BARCO A" case for steering committees). In the course under discussion, each company's approach to a given issue (for example, use of steering committees) is examined. This means that you have one case per company per issue which often leads to a multi-dimensional analysis of each of the management issues of the course.

Issue perspective. The real-world exposure builds a perspective for the issues under discussion. When students learn by reading articles and textbooks, they tend to treat the topics of the course as if they were all of uniform importance. This is reasonable for them to do given that they may have little real-world experience (or little real-world experience in MIS) to draw on to help them sort this out. The benefit of working with managers, who spend their days dealing with issues identical to those under discussion in the course, is that it provides a perspective for the relative importance and magnitude of these issues.

Confidence boosting. This point follows from the previous one. Students gain confidence in dealing with managers--confidence about themselves, about the course material, and confidence that the issues which they are discussing are of vital interest to the majority of organizations. Perhaps most importantly, students develop the confidence to ask "dumb" questions--the ones for which they don't already have (or think they have) answers.

Unabridged Complexity. The real world

is complex. This course does not hide this complexity from the students. They must face the issues in their natural form unaltered by editing and retelling. This realism is at times bewildering but is always fascinating and this provides the incentive for students to wade through the detail to sift out the critical facts. It encourages and rewards thoughtful and reflective analysis as opposed to the "shoot from the hip" style of management that is often encouraged (either explicitly or implicitly) by other pedagogical approaches.

CONCLUSIONS

This course bridges the gap between theory and practice in the IS area to create a dynamic classroom environment that is reflective of the actual environment of IS. The benefits to be reaped by such a classroom environment are significant. Perhaps the greatest single benefit is that students actually "learn" the course instead of being "taught" the course.

Model Curricula Implementation Project

Mary Jo Haught, CDP
Training Coordinator, Hospital Computing Services
University of Virginia, Charlottesville, VA

ABSTRACT: A region project for DPMA members to work more closely with educators centered on the DPMA curricula models. Personal interaction was emphasized and educators' responses became the basis for expanding the scope of the project. After a year, the time frame was extended several years and the area was increased to include all regions of North America through a 1987 DPMA committee. Benefits to both DPMA and educators resulted.

Educators and professional organizations both share a common primary objective. It is the development of human resources--the opportunity for people to fulfill themselves by improving themselves. However, in many cases these two social institutions do not make the best use of the support they can gain from each other. During the last several years, the Data Processing Management Association (DPMA) has made a major commitment to the development of various levels of model curricula. This effort represents a direct contribution of professionals in support of the educational process.

In 1981, the first curriculum model sponsored by DPMA was published: Undergraduate Computer Information Systems, the result of the three-year development. This was followed in 1982-1984 by the drafting of the secondary curriculum. In 1985, DPMA published its associate level curriculum aimed at imparting entry-level job skills distinct from the first two years of the undergraduate model. Finally, in keeping with the concept that these must be living models to serve the changing field of computers, a revised CIS '86 model for undergraduate education was released. Work on a graduate document is currently in progress.

These documents were the result of cooperative effort between educators and industry through many iterations of feedback and change. But unless the curricula are actually implemented in the educational process, little has been gained--by either DPMA or the schools and colleges meant to benefit from the effort. DPMA tracking indicates that over 800 institutions have at least partly implemented its models and the entire state of New York is using its secondary curriculum extensively.

Because of the importance DPMA as an organization places on education, one region decided to spend 1986 working more closely with educators. The goal was to establish closer relationships with educational institutions and the perfect point of contact seemed to be a ready-to-go product of the association: published copies of the model curricula. The South Central Atlantic Region of DPMA (Region 8) therefore

established a committee to ensure that these documents would be distributed to all educational institutions throughout East Tennessee, North Carolina and Virginia. Moreover, it was determined that personal contact would be the method of distribution.

The proposed benefits of such a project included:

1. More interaction and better understanding between industry and education.

2. Better and more relevant education for students.

3. Better employees and users for DP managers.

4. Strengthening of DPMA through:

   a. Educator participation in the association.

   b. Student awareness of the association.

In addition, the project offered two opportunities for the region:

1. A way to implement a major objective of the long range plan of the association: "To develop, maintain and integrate curricula models related to information processing, so that there is an annual increase of 20 percent in the number of educational institutions using them." (1)

2. A way to gather feedback from educational institutions about the model curricula and the association itself.

Obtaining copies of the curricula and distributing them personally was the first objective of the committee. The region chairman estimated the number of copies needed and began to work toward obtaining them. It was considered important for each level school to be aware of the other levels of curricula. Therefore, bulk copies of each (secondary, associate and undergraduate) were needed to reach every 4-year college/university, 2-year college and secondary school district/private school.

To provide local leadership in each state, a state coordinator was named. This individual was located in the state capital. Therefore, the state coordinator also was charged with maintaining contacts at the state level for public educational institutions and with monitoring statewide computer education thrusts. These coordinators determined what institutions existed and made individual packets for each. Within each chapter a special contact was named by the chapter to carry out the actual distribution. Groups of packets prepared by the state coordinator were given to each chapter contact. Then the contact or other supportive chapter members met with a representative from each educational institution.

To assist in presentations for groups (such as chapter educators' nights) the region authorized the development of a slide show by the committee. In addition, the committee developed supportive material for use by chapters. Included were lists of:

1. Educational institutions which had already adopted the model.

2. Experts to contact with curricula questions.

3. Publishers who support the curricula with books.

Also provided were data sheets to summarize educators' reponses; furthermore, samples of publicity and contact letters were shared. Finally, a PC graphics presentation was produced.

The educators contacted included administrators and those teaching at

MODEL CURRICULA IMPLEMENTATION PROJECT

Phase I - DISTRIBUTION

A. Set up committee
   1. Name region chair
   2. Select state coordinators
   3. Form chapter committee

B. Obtain materials
   1. Obtain supplementary materials relating to project from headquarters
   2. Estimate numbers and obtain copies of curricula
   3. Obtain additional materials from publishers, industry

C. Integrate industrial material
   1. Return chapter data to region
   2. Receive region summaries in chapters

D. Distribute materials

E. Obtain feedback
   1. At time of distribution
   2. Follow-up

Phase II - ADOPTION

A. Distribute evaluation materials

B. Obtain response
   1. Verbal estimate and copy of present/proposed curriculum
   2. Consultant contacts
   3. Evaluation document completed

C. Determine adoption status

D. Make written repsonse in each specific instance

Phase III - TRACKING AND REPORTING

A. Distribute updates to model

B. Evaluate adoption status if new model or new institution

C. Supply feedback to association

D. Obtain updates from association or other regions

Figure 1.

junior high, high schools, technical schools, community colleges, colleges and universities. Some chapters met educators individually at their institutions by appointment. Others sponsored free special dinners for groups of educators at a particular educational level or included this distribution as part of an educators' night for all levels of education in their area.

Several strong reponses were immediately evident as DPMA members and educators met. Two relevant areas were of great importance to those in education:

1. Educators have a serious concern about accreditation and whether the DPMA curriculum for their level with be a plus in the accreditation process.

2. Educators want to know if the specific employers who hire most of their graduates approve of the curricula.

Two findings of particular interest to DPMA were:

1. Some educators know little or nothing about DPMA--even that it exists.

2. Some educators have previously obtained copies of the model and used it in planning their curriculum--but DPMA is unaware it was adopted.

To continue to improve the dialog with educators and to address their particular concerns, it was recommended that the project be extended for three years and that its scope be expanded. The region chairman and state coordinators made a three-year commitment with the backing of the region officers and directors. A database was designed to implement more organized feedback. Chapters began to contact industry sources in their area and request formal endorsement of the curricula. And, finally, a region process to evaluate and formally recognize educational institutions which adopt the curriculum was developed since DPMA does not accredit as such. Moreover, it was recommended that the region project be carried out throughout the association.

The recommended changes were incorporated and the Curriculum Implementation Committee was formed to carry out the project throughout all regions of DPMA. In February, 1987, Dr. Madhu Trivedi, DPMA Vice President, Education Services, contacted the region presidents to request their assistance in the committee's work thoughout North America. A network of members was formed to carry out the outline for the project in each region as shown in Figure 1.

One of the outcomes of the project is a database which will be coordinated with other functions, such as updating the curricula or planning new projects for education (like the associate level inservice). Already through contacts in the project, educators have joined or rejoined the association because of this visible effort to support their needs. Within the association, many managers have become more aware of educational institutions and have joined advisory committees as a result of contacts initiated in the project.

These are only a few of the far reaching results of the project. Although other more specific results could be discussed, the most important outcome is that an ongoing atmosphere of cooperation has evolved which is only the beginning of many joint ventures between DPMA and educational institutions.

REFERENCE

(1) "DPMA 1986 Business Plan published," INSIDE DPMA, Spring, 1986, p.15.

A PROJECT-BASED SYSTEMS ANALYSIS AND DESIGN COURSE

Albert L. Lederer
The Joseph M. Katz
Graduate School of Business
University of Pittsburgh

A systems analysis and design course integrates conventional text and lecture material with real-world projects from local businesses. The course uses the projects to enhance the development of skills related to business problem solving, planning and control, and organizational and political sensitivity. The project-based course offers benefits to both its students and its sponsors.

In recent years, significant improvements have taken place in the education of entry-level business systems professionals. Data processing and computing associations have developed information systems curriculums which many colleges and universities have found useful (1, 2). Students can now learn techniques and develop skills in such areas as business programming, data management, systems analysis, systems design, and data communications. This represents a considerable advance over earlier years. When business and industry needed entry-level analysts and programmers, their best option was often to hire computer science majors whose expertise was seemingly limited to FORTRAN programming, compiler development and operating systems design.

However, one difficulty in the education of entry-level business systems professionals remains the development of an appreciation of the applicability of the techniques. Students need to understand how business problems motivate computer systems solutions. It is all too easy for the techniques to become ends in and of themselves and for commitment to the technology to exceed commitment to solving business problems.

Likewise, organizational and political sensitivity is extremely critical for systems professionals. Personal communications skills may, in many instances, outweigh data processing skills. However, such skills are very difficult to teach in a meaningful fashion within a conventional classroom setting.

This report describes a Systems Analysis and Design course which attempts to overcome these difficulties by requiring students to complete a real-world information systems project.

STUDENTS' BACKGROUND

The Systems Analysis and Design course is an elective in the final term of a one year M.B.A. degree. Hence all

students have earned an undergraduate degree although their fields of concentration vary widely and are usually unrelated to computing. Some have had work experience.

All students have completed two M.B.A. prerequisite courses related to information systems. The first, Computer-Based Information Systems, covers such elementary computing skills as BASIC programming, spreadsheets and statistical analysis. The second, Management Information Systems, introduces the uses of computing in business and industry, organizational and political issues, hardware, software, database management systems, systems analysis and design, and the systems development life cycle. Students might also have had additional courses in programmming or in database management systems.

CONDUCT OF THE COURSE

The first session of the course begins with the instructor's brief description of several projects. All projects represent actual problems of local firms assembled by the instructor from contacts in MIS departments, functional departments or even top management. The students then individually list the projects in order of their preference. The instructor assigns the students to the projects in groups of two to six students based on the students' preferences and background. The students contact the sponsors to set up an initial meeting.

The projects are the focal point of the course. Throughout the term, each project team gives a weekly, 15 to 20 minute presentation of its progress to the rest of the class. During the presentations, the students must respond to questions dealing with issues covered in the prerequisite MIS course. For example, how does the project contribute to the overall objectives of the firm? Who might resist the project and why? How would you associate costs and benefits with the project? At the conclusion of the project, what should the sponsor do and why, and what will the sponsor do and why?

Around the third week of the project, the project teams prepare a schedule with tasks and target dates using Gantt charts. They also prepare one-page biweekly status reports outlining three items: their achievements during the past two weeks, their anticipated problems, and their plans to overcome the problems. The schedule and status reports are written to be shared with the sponsor. Also, there is a final comprehensive written report and often an oral presentation to the sponsor.

Supervision of each project varies although it is generally not extensive. In some cases, the sponsor plays a significant role in coordinating the project while in other cases the instructor does so. In all cases, the students maintain considerable independence in planning and carrying out their project.

INTEGRATION WITH SYSTEMS ANALYSIS AND DESIGN

While the project comprises a major portion of the course, it also offers an opportunity to include conventional educational material. Hence, students read several chapters from a textbook on systems analysis and design, and they attend lectures on the same topics. Two take-home exams require the students to write essays about how they might apply the techniques in the textbook to their projects or to their sponsors' organizations.

EXAMPLES OF RECENT PROJECTS

All projects must meet a few basic requirements. They must make a genuine contribution to the sponsor's organization; "make-work" projects are not considered. They must be completable during the fourteen week semester. And of course, their

necessary skill levels must not exceed those of the students.

Some examples of the projects have included the following.

o An automobile association's membership system permitted terminal access only at its headquarters. Its branch managers also wanted online access. A student project team prepared a feasibility study for branch automation. The study included two alternative telecommunication system designs and a detailed analysis of the financial benefits and costs associated with each. The recommendations helped the association make a decision regarding its course of action for branch automation.

o A leading producer of mining equipment had installed a human resources information system software package. The company had never implemented the Affirmative Action module which was to produce a job group analysis and summary report. A student project team analyzed the organization's Affirmative Action needs, studied the package, determined its requirements, provided the necessary data and parameters, tested and debugged the module, and outlined a procedure for regularly producing the report. The firm now uses the students' work when it prepares its annual Affirmative Action plan.

o A small metals wholesaler decided to computerize its order entry and inventory control processing. A group of students analyzed and flowcharted the present manual system and evaluated three alternatives: the acquisition of a software package with no computer program modifications, the acquisition of a package with modifications, and the development of customized programs. After comparing the firm's needs to the features of several packages, the students recommended the customized programs. Because the firm had no in-house programmers, the students evaluated proposals from five software developers and recommended one of them.

o A large metals processor observed a decline in the demand for the services of its Information Center (IC) wondered if it was reaching all possible users. A student team analyzed personal computer purchase orders, studied training class attendance rosters, and interviewed 81 users to better determine their support requirements, concerns and system usage. The students discovered that the demand for service was very great but was not being met because it had changed since the inception of the IC. The students' recommendations included a marketing and communications plan, the expansion of services to meet the changing needs of the users, and the expansion of the IC staff.

o A large chemical manufacturer had undergone considerable growth in the acquisition of hardware and software. A student project team analyzed the current equipment request process and charted it using data flow diagrams. The students also interviewed users to determine the desirable features of a new automated inventory system and recommended a course of action for the organization. This helped prepare the manufacturer for the selection of an inventory package.

o The local branch of a large international bank suffered from excessive paperwork due to the duplication of client information, from inordinate manual calculation of routine financial information, and from insufficient control of client information especially in loan proposal and letter of credit processing. A student project team analyzed the existing system, drew up reports and screen layouts, and designed a database that would produce the reports. Systems analysts from the bank's North American headquarters in New York attended the team's intermediate and final presentations.

o A large food retailer decided to

install a job accounting system for its computer resources. Students studied the job accounting system and the alternatives available to the retailer. The students prepared recommendations regarding staffing, the appropriate chargeback method for various resources (computer time, personnel, supplies, etc.), the potential uses of the accounting information, a coding system for identifying program flows, and an implementation plan. The recommendations helped lay the groundwork for the installation of the job accounting system.

WHAT THE STUDENTS LEARN FROM THE COURSE

The course reinforces skills and techniques learned from textbooks, lectures and assignments in this and previous courses both by actual experience and by listening to the accounts of other project teams' experiences. It also enables the students to appreciate how real business problems create the need for computer systems.

Furthermore, the project exposes the students to organizational and political issues. Frequently, the students must work with users and others who may be very busy, indifferent, or even resistant to the project. Communications skills and their importance are thus learned first hand.

The project also offers considerable opportunity for initiative and independence. This contrasts many classroom assignments which are highly structured. Hence, students frequently describe the project on their resumes and discuss it with potential employers.

WHAT THE SPONSORS GAIN FROM THE PROJECT

For the sponsor, an important task is completed. Generally, this includes a fresh perspective on its problem. Sponsors have frequently shown their appreciation for this in various ways.

Sponsors also have the opportunity to evaluate the students and consider them for employment. In one case, a sponsor created a new position in order to hire one of the students as a systems analyst after graduation. Finally, the project gives the sponsor opportunities for additional ties to the business school.

REFERENCES

1. DPMA. DPMA's Model Curriculum DPMA, Park Ridge, Ill., Oct., 1980.

2. Nunamaker, J.F., Couger, J.D., and Davis, G.B. "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs, Communications of the ACM, 25(11), November 1982, pp. 781-805.

PROJECT COURSE IN SYSTEMS ANALYSIS

Robert A. Barrett
Dale K. Hockensmith
Computer Technology Department
Indiana University - Purdue University
Fort Wayne, Indiana

ABSTRACT:

Project oriented courses have been a recognized approach in programming for enhancing student learning and comprehension. A project-oriented course must be complex and yet not so large that the project can't be completed in one semester (1). In a degree program dealing with Information Systems a key component must be Systems Analysis and Design - the skills that an analyst will need for employment (2). The student in learning computer science and programming skills gains experience from the generation of programs and completion of projects. This same technique and approach can be applied, quite successfully, to courses dealing with Systems Analysis and Design.

INTRODUCTION:

We have previously reported about our five course sequence approach to the entire topic area of Systems Analysis and Design and Information Systems (4). A key component of the sequence is the second course which is taken in the first semester of the Sophomore year. This is the course dealing with detail systems design. The student has had a freshman- level introduction to Systems Analysis and Design but has not had the opportunity to put all of the pieces together into a structured project. We also treat this course as a possible "capstone" course for those students who will obtain their Associate Degree and go into the workplace and not continue their education. We recognize that students who choose this option will be entry level programmers at first and will most likely be working in programming teams and working with the systems analyst or programmer analyst. We want the student to be prepared for this workplace environment.

In order to provide the student with knowledge that goes beyond programming skills (the ability to think analytically) we have provided the systems design course to expose the students to the processes necessary to investigate an existing business problem and to develop a proposed computerized solution to that problem. With the completion of the course the student is not expected to have all the background to go into the business world and design large systems, but he is provided with the opportunity to perform the processes and to try his hand at system design and learn from his successes and failures (and those of the rest of the class as well).

The course has been developed to provide the students with more in-depth experience in system design techniques. In addition to the normal textbook

studies of various standard business systems, the student is put into a team design project to permit him to learn, not only how to design an effective system, but how to interrelate with others involved in the system design project. The systems chosen for the project are fairly straightforward so that the relatively inexperienced students can understand the applications and develop a reasonably good design.

The students are assigned to act as programmer analysts in that they are provided with system functional specifications and are to generate the detail system specifications -- including system files and report layouts. Currently the class is being exposed to manufacturing systems. Upon completion of the design project the students will have not only extended their knowledge of the specific system area, but will also have grown in their knowledge of systems design in general and will have experienced the advantages and difficulties of working within the team environment. The need for original ideas in system design is stressed heavily, and the teams are encouraged to consider nonconventional ideas in providing for the informational needs of the company - although the proposed solutions must be practical and functional.

One idea that is repeatedly emphasized to the students is the importance of the KISS method of software development. We feel that by developing a simple and straightforward system, the problems involved with modification and maintenance of that system will be reduced considerably, therefore saving time and money (and frustration) for the organization over the long run. In addition, by dealing with a non-complicated -- but complex -- system, the programming of it and the testing and debugging of it will be easier; the ultimate result being that the system should be more reliable and more usable.

The project is composed of nine different, but related, areas:

PROBLEM OVERVIEW

The first area to be developed by the student is an overview of the problem that the system is to solve: a statement of the problem (the real problem, not just the symptoms) and the goals and objectives of the proposed system that is intended to solve those indicated problems. It is necessary for the students to understand fully that the "problems" that are identified by the potential users of the system may not really be the problems to be resolved but merely symptoms of underlying but unidentified malaise and that failure to ferret out the cause of the stated symptoms can lead to disastrous results not only for the system under design, but for the company as well.

DOCUMENT FLOW

The second project area is composed of the document flow and process flow charts depicting the current system used by our hypothetical company. Here, the teams are encouraged to identify what they understand to be a "typical" computer system that has been in existence for several years or to depict a manual system. In some cases enterprising development teams have contacted local businesses and analyzed actual systems presenting copies of the completed projects to those businesses. The teams have shown remarkable ingenuity in defining the potential flaws in existing systems and have identified deficiencies that are not readily apparent.

CURRENT SYSTEM ANALYSIS

Strong points of the current system are noted so that they can be incorporated into the new system and deficiencies are identified so that they will be remedied. In this manner the students are reassured that all is not chaos in the business community and that there are many examples of extremely well designed systems currently in use. The reason for new system development is not necessarily that the prior systems were poorly designed originally, but because conditions have changed.

NEW SYSTEM DESIGN

As we have instructed the students, the design is started with the required output from the system as indicated in the limited functional specifications. As we had mentioned earlier, this task requires the teams to investigate systems depicted in textbooks and to investigate canned software that can be purchased for mainframe or micro based systems. We have repeatedly emphasized that the user may not completely understand what he actually needs or wants. It then becomes the task of the system designer not only to provide the needed information but also to suggest improvements to the proposed system. One of the ways that we have indicated for obtaining the detailed systems experience that permits the analyst to make those suggestions is to obtain literature from software houses and computer manufacturers dealing with systems they provide. Although the teams are warned that they will not want to include every feature from these various systems in their design, they should certainly consider features that are included in most of the purchased packages they investigate. Report layouts are then generated for batch reports and CRT screen layouts are designed for any on-line inquiries planned.

Once the design team has identified the required output, they can then proceed to making the determination of where the input will be obtained. As much as possible they are to get the raw data from existing computer systems rather than rely on manual input. Existing master files are expected to be utilized, if possible, in editing any manual input brought into the system. The concept of integrated systems is heavily stressed not only to reduce the manual input but to provide consistent data among all systems utilizing the central database files. Where manual input cannot be avoided, the teams are required to provide input forms for any batch input and CRT screen layouts for any on-line data entry. Here, we stress the importance of designing an entry screen that the user will feel comfortable with and not one that is convenient for the analyst or programmers.

Lastly, the teams can make the determination of the master data files required by the system. The internal data files are defined not only by the input and output but also by any other data the teams determine may be required for future system enhancements. We have stressed that the systems being designed must be flexible and expandable.

COST/BENEFIT ANALYSIS

The teams are to provide an economic cost/benefit analysis of the proposed design taking into consideration any proposed hardware changes that may be required for efficient functioning of the new system as well as costs associated with the design and programming of it. They are expected to investigate the cost effects of replacing hardware if necessary and also to consider the costs of installing any hardware features recommended for the new system. Although it may be easy to provide cost justification if only

system development and programming costs are considered, the teams must also include costs of developing proper internal documentation and user manuals as well as the costs of actually training the users in the use of the new system.

The Project Control function cannot be neglected. Utilizing the ubiquitous Gannt and Pert charts, the design teams are required to allocate their resources so that they can commit themselves to a realistic target date. They have been instructed to consider as many potential trouble areas as possible to ensure that the project can be completed within a reasonable time of the target date.

PROGRAM SPECIFICATIONS

Finally, the individual programs themselves are designed - although no coding is done. The logic is assumed not to be obvious and each function must be defined along with any special routines or decision tables considered to be necessary for proper program definition. Input and output file layouts must be provided as well as any report layouts and input/output CRT screen layouts. In the event that processing logic is quite complex and understanding of the function by the programmer may be difficult, examples of processing are encouraged so that the completed program can be tested utilizing the provided sample data.

BELLS & WHISTLES

Although the system definition is complete after the sixth phase of the project, the project itself is far from complete. Another phase is developed to describe what is referred to as "Bells and Whistles" - functions that may not be required for effective system utilization but can be readily added into the system at a relatively small cost in time and effort. Here the design team is exhorted to stretch their imaginations to provide some special functions that can yield benefits to the users that may not be readily apparent. Special features built into the design that the team considers unusual are noted and the potential impact on the company indicated. Here, our objective is to encourage creativity within the design teams as long as the proposals do not require undue complexity within the system.

CURRENT SYSTEM IMPACT

Here the teams are required to assess the impact of the new system on the personnel of the company -- both labor and management. The expected benefits of the system must be examined and critiqued along with the expected negative impacts. Since no system can be expected to provide only benefits, the teams are instructed to anticipate some negative reaction to the system and be prepared for it.

GOALS AND OBJECTIVES

In summary, the design team will again indicate the basic goals and objectives of the proposed system. They will restate targets that they have established and how they will organize their design team to meet those targets. Features of the system will be used to determine whether the objectives will be met and how the system will assist the organization in reaching the established goals as defined by top management.

OUR PERSONAL OBJECTIVES

Although our objectives are to provide extensive experience to the students in systems development as well as programming skills, the question always

arises as to whether our objectives are being met. We feel that the answer is that we are accomplishing what we have attempted to do. The individual faculty member teaching this course works very closely with the student project teams. The students in this course must utilize the communication skills (writing and speaking) that will be required in the workplace (5). The individual faculty member must be a facilitator for the student if the course objectives are to be achieved. This does require, in our opinion, extra work in course preparation, course guidance, and time outside the classroom for meetings with the project teams. In effect, the faculty member must play the role of the project leader or Information Systems manager.

CONCLUSION:

Our approach to the project course in teaching Systems Analysis and Design provides an experience opportunity for the student to pull together the concepts of Systems Analysis and Design. The benefits of doing a project from beginning to end brings the textbook material into focus. We feel that this prepares the student for his first position as a programmer working with analysts and gives him the background to understand more fully what care must be taken in the design and programming phases to ensure that the newly proposed system is indeed an undertaking worthy of the time and effort expended to bring it into being.

REFERENCES

1. "The Anatomy of a Project Oriented Second Course for Computer Science Majors", Will Gillett, SIGCSE Bulletin 12, 1 (February 1980), pp. 25-31.

2. "A Major in Information Systems", Fred J. Maryanski and Elizabeth A. Unger, SIGCSE Bulletin 12, 1 (February 1980), pp. 25-31.

3. "Critique and Evaluation of the Cal Poly/DPMA Model Curriculum for Computer Information Systems", William Mitchell and James Westfall, SIGCSE Bulletin 13, 1 (February 1981), pp. 153-60.

4. "A Five Course Sequence for Information Systems", Robert A. Barrett, SIGCSE Bulletin 14, 1 (February 1983), pp. 114-122.

5. "Methods and Approaches for Teaching Systems Analysis", Panel Discussion, Robert A. Barrett, Moderator, SIGCSE Bulletin 15, 1 (February 1984), pp. 86-

THE SENIOR PROJECT - EXPERIENCES & IDEAS

by

Dr. John Maniotes

Professor

Information Systems and

Computer Programming Department

Purdue University Calumet

Hammond, IN 46323

Track: Innovative Teaching Method I - CIS Core Area

Abstract

This paper presents ideas and actual experiences accumulated over the past seven years on conducting a Senior Project course for CIS majors. The paper also is intended to serve as a guide for those faculty who are interested in conducting a Senior Project course.

PURPOSE OF THE COURSE

The senior project course at Purdue University Calumet is a capstone course which serves to integrate the knowledge and abilities gained through other computer related courses in the curriculum within a comprehensive system development project. Currently, this course is titled CIS 426 Applied Software Development Project (3 credits) and is an 8th semester course in the Information System and Computer Programming Curriculum.

PRE-REQUISITES

Students participating in the senior project are expected to apply concepts mastered in prior computer and non-computer courses. The following is a minimum list of topics that the student is expected to have mastered:

* Security and audit control
* Structured systems analysis & design
* Classical systems analysis & design
* Data/file structures & design
* A specific data base package
* On-line programming techniques
* Structured and top-down methodologies

TEAM RESPONSIBILITIES

All students in the course participate in the assigned project. Students are grouped into various teams. Each team, composed of three students, faces the following tasks for the project: analysis, design, programming ,test data creation, key (data) entry, testing and debugging, implementation, and documentation

Each team is expected to choose a Project Leader who will provide the necessary leadership and who will foster group discussions and decision making. Furthermore, the Project Leader is expected to handle the administrative aspects associated with this project and to report to the instructor the progress, problems, etc., related to this project.

ENVIRONMENT

Over the past 7 years, the senior

project has been designed, developed, and processed on the following systems at Purdue University Calumet:

Computer systems - DEC VAX 8600, DEC VAX 11/780, IBM PC, IBM 4341, IBM 370, and PRIME 300

DBMS - 1032, Data Boss, Adabase, and dBASE III Plus

Secondary Languages - COBOL, PL/I, BASIC, FORTRAN.

PHASE 1 OF THE SENIOR PROJECT

Phase 1 is concerned with the development and implementation of directories, security measures, and high level modules and menus.

Figure 1 represents the directory scheme for each team. Additional sub-directories are created in Phase 2. Students use the Test sub-directory to develop their modules. As the modules are de-bugged and implemented, these are moved over to the Production sub-directory. During the final team interview/presentation, the project is demonstrated from the Production sub-directory.

All high level modules and menus are implemented using a data base management system package as defined in the Environment section of this paper. From a user's point of view, the project must be easy to use and user friendly with appropriate prompts and help screens.

In preparing menus for this project, students are expected to highlight menu items for all major screens via appropriate video and sound attributes. Menus are either of the "pull down" or "push up" type. The split screen capability of the monitor is utilized for displaying variable information on the upper 2/3 of the screen (known as the upper window) and other information on the lower 1/3 of the screen (known as the lower window). In the lower window, the students display the programmer's name who created this menu/module; module name and identification number; current date, time, and day of week dynamically; and any other useful information.

The high level modules and menus for Phase 1 are represented in Figure 2. The Initial module simply identifies the course, type of project, the names of the members of the team, and the team number. The rest of the modules in Figure 2 are self-explanatory.

When the project terminates, the final screen displays some statistical information such as the name of the application terminated; start and stop time/date for the session; the elapsed time; and an appropriate graphical oriented sign-off.

PHASE 2 OF THE SENIOR PROJECT

This phase deals with the implementation of one of the lower level modules identified in Phase 1 and shown in Figure 2. Over the past seven years, the following applications have been implemented: A comprehensive payroll package for a service bureau, an inventory control package, a comprehensive statistical package, a modeling and engine combustion package, and a COBOL verb analyzer.

The most popular applications with CIS majors are the financial application packages. This part of the paper describes the comprehensive payroll package that was recently implemented.

The specifications for the payroll system are approximately 120 pages (8 1/2 x 11 and double spaced). The specifications are considered to be minimal and incomplete. Based on their experience, students are expected to research, expand, enhance, and implement these specifications accordingly.

For the Senior Project, students are

assumed to be employed by a service bureau as programmers/analysts. Furthermore, it has been assumed that they have been requested to implement in-house a system using a DBMS to handle the service bureau's payrolls of their existing clients and to streamline, enhance, and consolidate the payroll master files. The new system must be able to process payrolls from the construction oriented companies as well as non-construction oriented ones.

The amount and type of test data for testing all inputs, files, boundary conditions, and processes is the responsibility of the students. Their grade, in part, is determined by the quality of their test data.

The important modules for the comprehensive payroll package for a service bureau environment are represented in Figure 3. Appropriate security measures are expected to be implemented wherever feasible.

FINAL TEAM INTERVIEW/PRESENTATION

During the 13th week of the semester, two hour appointments are scheduled with each team. The number of teams during the past 7 years has varied between 12 to 18 teams per semester

The presentations are open to the faculty of our department. The instructor and faculty ask the questions. Students are expected to "defend" their project design and demonstrate that the modules work according to the specifications.

The presentations take place in our Computer Resource Center which is a large room equipped with an appropriate number of color terminals, printers and personal computers. The images on the terminals are displayed on a large 6' by 6' screen.

The final presentation for each team lasts approximately 2 hours. Prior to the final presentation, other interviews/presentations have taken place with each team early during the semester. For example, during week 6, a one-hour initial interview/presentation takes place regarding Phase 1. Another one-hour session takes place during week 9.

Students are expected also to prepare in a professional manner a formal report. All narrative documentation is prepared double-spaced using either a word processing system or a text editor.

Students place in a separate binder for each application all appropriate directory listings, DBMS and program listings, printer outputs, menus and screens. All programs are expected to be well documented with comment lines.

TIPS ON CONDUCTING THE SENIOR PROJECT COURSE

* Deadlines

It is important that the instructor establish reasonable deadlines and adhere to them. Otherwise, students tend to "coast" and allow their project assignments to "slide by". Project Milestones are one of the ways to establish these deadlines for the students. The appointment of a Project Leader for each team also serves that purpose.

* Presentations by Each Team

The number and frequency of the presentations is also very important. We have found, form past experiences, that if each team partakes in 3 presentations during the semester, the project is more apt to succeed than fail. Furthermore, during the early presentations, serious design flaws and errors can easily be detected and later corrected.

During these presentations, the students demonstrate the various working modules and discuss their progress and problems. The instructor

tests the various modules to insure that they work according to the specifications and provides suggestions on improving the design and implementation.

The final presentation by each team is held about 4 weeks later toward the end of the semester. This is a formal 2-hour session where the students are asked to "defend" the implementation of their project. The defense effort is concentrated around Phase 2. Faculty members from the department are invited to critique and test the various modules of Phase 2 and partake in the discussion.

During each presentation, the Project Leader (or someone else on the team) records any problems, errors or flaws that may be discovered. These "loose ends" are later written and copies are distributed the next day to the instructor and to the rest of the team members for implementation and fine tuning of the project.

* Meetings

The instructor for the senior project should be prepared to spend additional time meeting informally with the various teams in order to answer their questions, clarify the project specifications, and occasionally settle "disputes". From time to time, the instructor should also hold "surprise" 20 to 30 minute meetings where a selected team is required to demonstrate their progress to-date. These surprise meetings also serve as a means to keep to project on schedule.

* Sources of Materials for the Project

Good senior projects are difficult to come by. Although some case studies exist, we have found the materials and ideas from most of our senior projects come from the instructor's industrial experience or consulting assignments.

* Faculty and Student Feedback

Early in the semester the senior project is distributed to the departmental faculty for their review and comments. Before the 12th week, each faculty member receives a schedule of when the teams are expected to make their final presentations. The Faculty is encouraged to attend as many of the final 2-hour presentations, ask questions, and critique the implementation of the senior project.

Student feedback is also solicited during the semester regarding the specifications, goals for the project, and future improvements.

SOME FINAL THOUGHTS

The advice I would give to those faculty members who are thinking about teaching a Senior Project course is as follows: Be prepared to spend a lot of time with your students. It is important that close supervision be maintain over the teams; otherwise the quality of the senior project will suffer.

Whatever senior project you initially assigned, be prepared to revise and enhance it accordingly. Your students and faculty are your best sources for these revisions and enhancements. Also change the senior project from year to year, and let another faculty member participate in this course.

I endorse the concept of a Senior Project course for any CIS curriculum. This should be one of the final courses for the CIS majors. The course is a challenge to both the students and the instructor. The opportunities for learning and developing a real world project are certainly there.

FIGURE 1

LOGICAL HIERARCHY OF DIRECTORIES

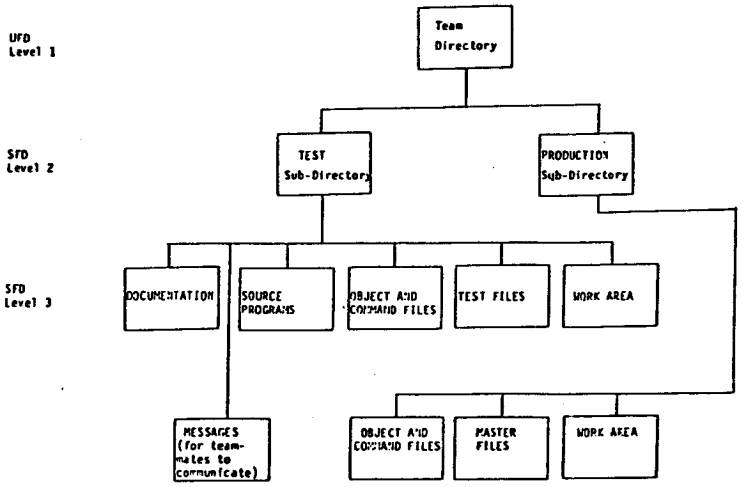


FIGURE 2

A LOGICAL REPRESENTATION OF THE PROJECT HIGH LEVEL MODULES AND MENUS FOR PHASE I

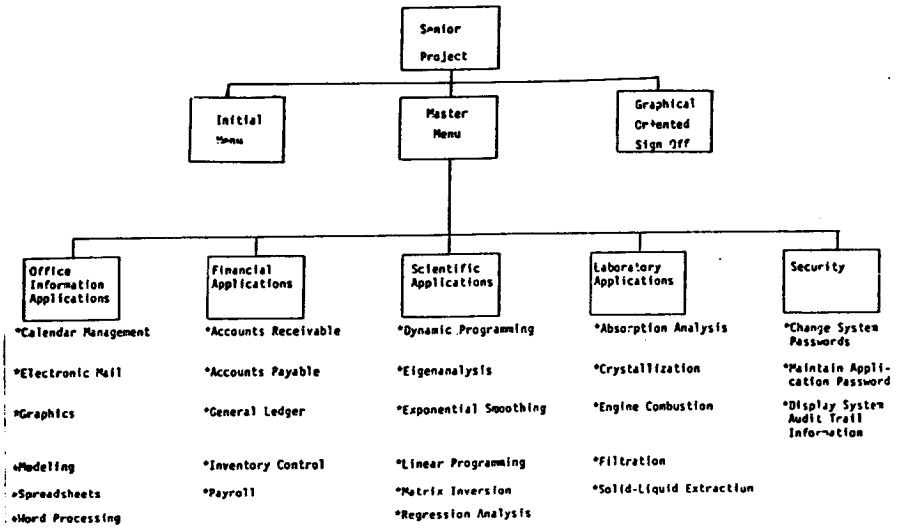
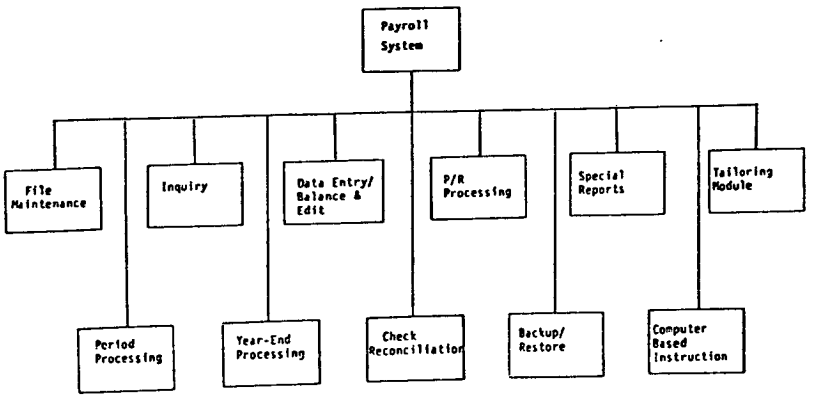


FIGURE 3

LOGICAL HIERARCHY OF THE VARIOUS PAYROLL MODULES FOR PHASE 2



BEHAVIORALLY BASED TEACHING METHODOLOGIES FOR CIS COURSES

Richard Leifer
School of Management
Rensselaer Polytechnic Institute
Troy, New York 12180

Abstract

Behavioral concerns impact CIS effectiveness to a considerable degree and yet there seems to be a limited emphasis on teaching behavioral issues using behavioral methodologies. Part of the reason is that there are few behavioral activities available for the CIS curricula. This paper presents four behavioral activities that we have used with success in our CIS courses. Hopefully, these will begin a movement to bring behavioral issues into CIS courses as well as to motivate technically oriented instructors to try behavioral activities.

The importance of different modes of learning is often overlooked in teaching CIS materials. There is a preponderance of emphasis on more cognitive skills, while, at the same time, writers in the field suggest that behavioral problems are becoming more important to eventual success or failure of actual information systems. Part of the problem is with our educational system, which does not educate for the "right side" of the brain, and part of the problem is with our MIS curricula which emphasizes technology to the detriment of the behavioral aspects of systems.

The emphasis on more cognitively oriented teaching styles can be referred from an analysis of the data in a recent paper by McLeod (1985) (2). There seems to be an unstated assumption that better systems are those that are more technically sophisticated.

This paper presents four behaviorally oriented experiential exercises for CIS courses. Each exercise is described in the following format:

a. Name and objective of each exercise.

b. A brief description of the methodology of each exercise.

c. Discussion and suggestions for carrying out the exercise.

1. The Effect of System Designer Characteristics on System Design

Objective: To demonstrate that personality characteristics of systems designers influence their designs.

Method: Students are given the Myers-Briggs Type Indicator (short form) which measures Jungian dimensions. On the basis of their

scores, the class is divided into four groups corresponding to the four Jungian types. These homogeneous groups are then given two cases describing companies which require information systems. One company needs a more centralized, control oriented system while the other company needs a more open, networked kind of solution to their problem. Typically the Jungian group that is more detailed oriented will try to implement more control oriented systems to both problems than the more intuitive group who will suggest more network oriented solutions to both problems. The groups will defend their positions using terminology oriented to their personality characteristics.

Discussion: Most students regard IS design as "Objective" in the sense that any problem will determine the solution. Here however, it becomes apparent that since groups come up with different solutions, it may be that systems design is not as "objective" as students think. This exercise suggests that not only must the systems designer know what his or her own biases are with regard to systems. Finally, some organizations are control oriented and need control oriented systems, while some other organizations are innovation oriented and need the greater information processing capabilities that promote innovation.

2. Designers and Users

Objective: The need to involve users in design is an old saw with regard to design. This exercise demonstrates the consequences of not involving users in the design process.

Method: The class is divided into groups of eight or ten. The following discussion pertains to this smaller group. Half the group is sent out of the room with instructions to wait. The other half is seated around a table with pieces of a puzzle and an answer sheet of how the pieces fit together. This is the planning group (systems designers). They have forty minutes to come up with a plan to give to the group out in the hall (the users). No later than 5 minutes before the end of the 40 minute planning period the users must come in to be given the plan. At the end of 40 minutes, construction begins. The users must complete the puzzle with no help from the planners. The planners cannot show the puzzle solution to the users and they cannot draw a picture of the solution. However, during the 40 minute planning time, the users and planners are allowed to communicate in writing to the planners about anything that concerns them and further, the planners can call the users in at any time to discuss the plans that were made, but they cannot show the puzzle solution to the users.

Discussion: Typically, the planners become so involved in writing the plan that the users come in 5 minutes before the end of the planning session. When the users come in they are frustrated and angry. They have little motivation to solve the problem and many of the groups never do solve the problem. However, there are other groups who do call their users in early, spend a lot of time with them trying to get them to understand the problem and how they derived the solution. These groups typically solve the problem quickly (one group took 45 seconds) with lots of motivation. This is a powerful exercise. Users are usually quite vocal at the end and the planners are usually quite sheepish because they "know better." I have done this exercise in some consulting assignments and find that information systems personnel treat users in this exercise the same way that the

students did. The IS personnel start blaming the users and think that they're dumb for not being able to follow their terrific plans. Does this sound familiar?

3. Group Effectiveness: and Planning

Objective: This exercise is designed to demonstrate the effective use of skills and planning when faced with a complex task in order to ensure effective group performance.

Method: Groups of about 5 students are given a bag of tinkertoy parts. The parts are spread out on the table top or floor and the group is instructed that they have thirty-five minutes to plan to build the tallest free-standing tower that they can. After the planning period they will have 40 seconds to build the tower. They are not allowed to practice with the pieces or touch them in any way. At the end of the 35 minutes planning phase, the groups are given the ready, set, go instructions and the instructor counts off the seconds. The tallest tower as measured with a tape measure, or other device, is the winner.

Discussion: This demonstrates the need to not only effectively plan in terms of the skills and abilities of the people involved, but it also requires teamwork and coordination. The best results come from those that rehearse the construction and are realistic about construction difficulties. Many teams construct too complex a structure where the best solution is a single straight line up with a simple base. Most CIS systems are designed in teams with constraints on resources and time and at the end of their analysis and development time they often build overly complex systems. They need to budget their time and plan for their activities to finish within constraints. This exercise engenders a great deal of enthusiasm and participation.

3. Goals and CIS Effectiveness

Objective: The importance of clearly stated and measurable outcomes for systems are demonstrated. It is an old saying but true, if you cannot measure what your goals are, you will not know if you have achieved them.

Method: Students are divided into groups of about 5. They are asked to come up with a means for ascertaining the effectiveness of the information system at their university or college. Most students will have some experience with the IS center. They have 25 minutes to work on this. As they are working, I go around to ask if the items they have listed are concrete enough to be measurable.

Discussion: The impact of this exercise hits home when the goals across groups are compared. Usually there is a wide diversity of opinions. It soon becomes apparent that who in the organization you ask (operators, managers, administrators, users) or when you ask (before budget time, during the summer or right before finals when all the computer science and CIS projects are due) influences the results. This leads into a discussion about the meaning of IS effectiveness when there are multivariate goals which, in many instances are contradictory to each other. If there are contradictory goals, how do we judge effectiveness?

4. Organizational Impacts of Information Systems

Objective: Many times analysts take for granted the organizational implications of the informations systems they work on. It is often said that systems analysts are change agents. Nevertheless, there is less

than satisfactory understanding of the implications on the organization of the CIS's that are being designed. This exercise is concerned with the dynamics and needs of organizational inertia and CIS change.

Method: The technique to be employed here is called force field analysis. This approach derives from the assumption that current behavior is a resolution of two opposing forces: driving and restraining forces. Driving forces are those which mitigate for change while restraining forces are those which tend to keep the system in its current state. The exercise is to have students define the driving and restraining forces for a new systems application and then using these to suggest how to increase the driving force and how to reduce the restraining force so that the new system will not meet with resistance. The exercise is further complicated by dividing the class into two groups and separated. One group, the users, defines their version of the driving and restraining forces, while the other group, the systems designers, defines their versions of the driving and restraining forces. Virtually any short case illustrating systems design can be used. The two sets of forces are compared and contrasted. A third group can also be formed, these are the managers or administrators.

Discussion: The two (or three) sets of forces are generally different. This sets the stage for discussing expectations and perceptual differences between the groups. Effective system implementation can only occur when all parties agree about what the system is supposed to accomplish and how. This starts a discussion about perceptions and organizational realities.

CONCLUSION

The strong behavioral component of CIS continues to generate attention and discussion in the academic as well as the popular literature. However, students are often times shielded from this in their courses. More importantly though, the values and beliefs of most IS programs tends toward designing "elegant" systems. Unfortunately, this has little organizational reality associated with it. It is difficult for IS students to seemingly compromise IS design for the sake of organizational realities. Hopefully, behavioral exercises such as these will help them view IS and organizations as belonging to the same organizational system wherein the success of the organization takes precedence over the design of the system.

REFERENCES

(1) Kolb, D.; I. Rubin, and J. McIntyre, Organizational Psychology: An Experiential Approach, Englewood Cliffs, NJ, Prentice-Hall, Inc., 1984.

(2) McLeod, R., The Undergraduate MIS Course in AACSB Schools, Journal of Information Systems, 2, Fall 1985, p. 73-85.

(3) White, K. and R. Leifer, Information Systems Development Success: Perspectives from Project Team Participants, MIS Quarterly, September, 1986, p. 215-223.

DEVELOPING EFFECTIVE WRITTEN COMMUNICATION SKILLS IN AN ADVANCED MIS COURSE

Dr. Karen A. Forcht
Assistant Professor
Department of Information and Decision Sciences
James Madison University

INTRODUCTION

One of the first questions that prospective employers invariably ask when checking with a faculty member about a student is, "How well does this student communicate?" In order to be able to adequately respond to this question, professors must provide activities that aid the student in "putting the polish" on written communication skills, in a manner that is acceptable to industry standards and professional journals. By developing a large segment in written communication skills in an Advanced MIS course, students are able to tie together the various knowledge bases they have acquired in programming languages, computer technology, and business courses into a workable whole that enables the student to transcend from classroom to profession. If an Advanced MIS course is to be truly considered a capstone, a tieing together of the technology and ability to express oneself must occur. Only then are we truly sending our students out "armed for the battle."

PROBLEM AREA

Computer technologies have for some time been viewed by other professionals in industry as being rather poor communicators--both in oral and written inter-change. Technical professions lend themselves to an overabundance of acronyms, jargon, and industry-specific (computerese) language. In all too many cases, computer professionals are likely to find themselves addressing an audience that is <u>not</u> computer literate, thus compounding an already existing language barrier. Many colleges offer courses in business communication, which focus on correspondence and reports, or technical writing, which tends to focus on engineering-related topics or documentation/specifications writing. Due to the demands many students face in fitting all their required courses into a plan of study, many of them are not able to enroll in either of these much-needed courses.

GETTING STARTED

<u>Audience Definition</u>

Most students had never considered defining an audience as they are accustomed to presenting oral reports in front of a peer group or writing papers for instructors. When defining an audience, the writer must:

- define who the readers are
- define what the readers need to know
- define what the readers will

do with the information provided.

I also encouraged students to look at the cover pages of several journals as some information about the audience is given (i.e. practitioners, educators, business people, members of a professional organization, etc.). Many of the students spent many long hours going through journals in my office or in the library to "get a fix" on the audience.

Defining or Narrowing Your Topic

Being novice journal writers, most students tend to tackle a topic much too broad to adequately cover in a 10-15 page paper. One of the requirements, early in the semester, was for the student to submit in writing or to discuss with me, their narrowed topic. Invariably, I found the topic they had selected to be too broad and one that could be adequately covered only in an entire book.

The purpose of the paper, once established, would aid in this task by defining:

- the problem, or the central issue of the document
- the technical issues, or the specific major points to be made in the document
- the rhetorical issues, or what the document will do for the readers

Avoiding Stilted Writing

Most computer writing utilizes some jargon, as in all technically-dependent fields. What sets the computer field apart is that unlike medicine, law, astronomy, engineering, or other scientific fields is that the computer field is still in its infancy. Therefore, many of the terms or jargon that we use (and accept as common knowledge) are virtually unknown to others. Altering the terminology or oversimplifying tends to destroy the topic's integrity, and yet, readers must have a clear idea of the topics being presented in order to act or respond to the subject. Consequently, the issue of jargon becomes audience-dependent. Students are encouraged to use a language level that the audience will understand. Again, spending some time studying the various journals will aid in establishing the appropriate terminology level.

WRITER'S CHECKLIST

Students are asked to go through the following checklist prior to "wrapping up" their paper.

Some writers feel that, like airplane pilots, they need a checklist; that is, a list that provides a quick check of all the vital steps that need reviewing. Used conscientiously in revision, this checklist, will help prevent many embarrassing oversights.

(1) Do you have a good title? Is it short but informative?

(2) Have you stated clearly and specifically the intent of

the investigation or report?

(3) Have you stated clearly and fully the outcome of the investigation--what was actually accomplished?

(4) Have you clearly and fully described the methods, materials, and equipment you used in conducting the investigation?

(5) Have you put everything required into the report?

(6) Does the report still contain anything that you would do well to cut out?

(7) Does the report present the substance of your investigation in a way that will be readily understandable to your intended readers?

(8) Are your paragraphs clear, well-organized, and of reasonable length?

(9) Is your prose style (diction) clear and readable?

(10) Have you included all the mechanical and prose elements that your report needs?

(11) Are your headings and titles clear, properly worded, and parallel?

(12) Have you inserted the documentary reference numbers in your text?

(13) Have you keyed the tables and figures into your text and have you sufficiently discussed them?

(14) Are all parts and pages of the manuscript in the correct order?

(15) Will the format of the typed or printed report be functional, clear and attractive?

(16) Does your manuscript satisfy stylebook specifications governing it?

(17) Have you included required notices, distribution lists, and identifying code numbers?

(18) Do you have written permission to reproduce extended quotations or other matter under copyright?

(19) Have you proofread your manuscript, for matters both large and small?

(20) While you were composing the manuscript, did you have any doubts or misgivings that you should now check out?

CONCLUSION

Even though there is a general feeling that computer people are not good communicators, these students have proven otherwise. Many of the students have had papers accepted by referred journals, professional and trade publications, and for conferences. In many cases, these

undergraduate students were competing with graduate students and practicing professionals. Our acceptance rate has been monumental. In all my years of teaching, I felt the greatest sense of accomplishment in knowing that I somehow played a small part in "opening the door" and showing the student "how" to develop to their fullest potential. The look on a student's face as they burst into my office, acceptance letter in hand, is worth more than any other accolade a professor can be given. I was recently honored at our College of Business Awards Banquet for Distinguished Student Achievement and Scholarship. The award, however, is really theirs, not mine. What a wonderful experience this has been for all of us.

A Research Case Study for CIS I Students

by Alka R. Harriger and Helene P. Baouendi
Computer Technology Department
Purdue University
West Lafayette, IN 47907

ABSTRACT: A student must learn so many new terms and concepts in CIS-I that the task may seem too overwhelming. Ideally, this course should teach the student how to apply the newly learned computing concepts in the student's chosen field of study. Unfortunately, most students tend to memorize the multitude of definitions for upcoming examinations. This study method allows the student to easily forget what has been learned. Our CIS-I course requires the student to complete a case study research project in addition to covering the necessary computing concepts and terminology. The semester-long research case study gives the student immediate applications of learned concepts, thereby improving the retention of knowledge. In addition the student learns to communicate these computing concepts clearly and acquires technical writing skills that will be extremely important in his/her professional career. This paper will discuss all phases of the evolution of the research case study, the milestones, and how the case study has been implemented with different class sizes and different major fields.

INTRODUCTION

The Computer Technology Department at Purdue University closely follows the guidelines of DPMA's model curriculum for undergraduate computer information systems (CIS) education. [1] The first course in both the two and four year programs, CPT 154: Introduction to Computer Based Systems, has been offered every semester since the inception of our programs in 1978. Two years ago, we also began offering a similar course for non-computer majors in the School of Technology, CPT 110: Computer Literacy. This paper will describe the major research case study which must be completed by both the CPT 154 and CPT 110 students.

Approximately 100 students, all majoring in CIS, take CPT 154 each semester. The class meets twice a week as a large lecture, and once a week in three separate recitations numbering 30-35 students each. CPT 110 has approximately 35 students each semester and meets three times per week as a recitation/lecture.

Both groups of students are extremely anxious to begin using the computer. Most students in CPT 110 do not take another formal computing course. Therefore, instead of discussing just the terms and concepts, the students need immediate real-life applications to help learn the terminology. Students in CPT 154, on the other hand, will take many more computing courses. In their case, CPT 154 serves as a foundation for the knowledge they will acquire in other CPT courses. Therefore, they must retain the knowledge from this critical course.

The primary goal for both classes was to design an assignment which would help retain the vast computing material covered in a semester, avoiding what Naomi Karten calls "the attend and forget syndrome." [2] At the same time, this assignment should help the students to communicate clearly the new concepts learned and encourage them to get acquainted with the computing literature. Like Taffe, we believe that "skimming several articles before a final choice exposes the student to a broad spectrum of new concepts

and practices in computing." [3]

To be successful, the assignment needed to have a flavor of the real world, as well as provide immediate applications of the computer. Our goal was accomplished by creating a familiar problem that could possibly exist and may be solved by the use of a computer. As stressed by Susan Brender et.al., "these projects help keep student interest high and make the learning process highly relevant for future careers." [4] By the end of the semester, each student had a professional report of the research on the problem and the recommended solution.

BACKGROUND

Before discussing the actual case study, a history of the five major changes which took place in the assignment are provided. For each one, we will review the reasons for the format, the problems which arose, and the resulting solutions.

Originally, the student was instructed to write a report on a topic selected from a list given by the instructor. A handout that described the layout and objectives and provided some pointers was also given at the beginning of the semester. Although this format gave the students flexibility in the choice of a topic, they lacked the necessary computer background to sufficiently narrow the scope. Without the real world context, the assignment seemed like "just another composition" to most students; this resulted in less interest in completing the assignment. Finally, the lack of set milestones allowed many students to wait until the last minute only to submit a poorly written and poorly researched paper.

The next form of the research assignment corrected only two of the major problems. Several milestones were created to make sure the students worked progressively on the paper instead of making a final effort at the end. Students were asked to write a problem definition which described a possible real world problem requiring a computer for its solution. The possible "assignments" would be assigned to a classmate to solve. This format added some interest since the students solved their own problems. Unfortunately, many of the assignments created by the students were too broad and too complex. This could be attributed to the level of the students (entering freshman) and to their lack of computing knowledge.

The next phase of the assignment consisted of three scenarios authored by the instructor. All students in the same recitation were given the same scenario, but the material to be researched could be on computer terminals, printers or microcomputers. The students were also given many handouts to help them with each milestone. Because the text emphasized larger machines and a separate course covered microcomputer level material, the assignment was mainly mainframe oriented.

The students who had to research the microcomputer part of the project had few problems. Unfortunately, the amount of material easily available to students researching monitors or printers at the mainframe level was low. The freshmen also had difficulty locating references since they did not have the opportunity to get acquainted with the Purdue library system. Please see Harriger and Ho for more details on this assignment. [5] Although this format was fairly successful, the problem with locating references created unnecessary frustration on the part of the students.

The topic of the fourth form of the research case study was still authored by the instructor and consisted of an single scenario given to the entire class. To make each student's assignment more unique, the instructor assigned a unique microcomputer to each student. All students had to find another microcomputer system which could possibly be a solution to the problem outlined in the scenario. To reduce the difficulties in locating references, the instructor maintained at least one reference on each assigned microcomputer. The students were permitted to "check out" these references on a loan basis.

This format was much more successful than its predecessor, since all students had an equal opportunity with locating references. Unfortunately, the assignment of the

microcomputer with the scenario gave many students the incorrect assumption that the hardware selection drives everything else. Although lectures emphasized the importance of requirements and software, the format of the research case study unnecessarily contradicted these important concepts.

The fifth and current format of the research case study is very similar to its predecessor. In addition to the scenario, specific software packages were assigned to emphasize the software. The student had to select two microcomputers which could operate this software. To emphasize the requirements, the specific hardware/software were unnamed until the class agreed on all requirements. The next section will discuss how this was accomplished.

THE CASE STUDY

This section will describe each phase of the research case study as it progresses through the semester. Some are actual milestones which must be completed by the students for a grade. Others are class activities which help the student with anticipated problems. Throughout the semester, the students are given various handouts to help with upcoming milestones.

During the first week of classes, the student is given a scenario which describes problems being experienced by a contrived company. A background of the company, descriptions of key personnel, details of the problem and expectations of the solution are included. The student is instructed to complete this assignment as if he/she played the role of a specific company employee. All students are required to analyze the scenario to find any hidden requirements.

The next phase involves a class analysis of the requirements. When given the assignment, students are told they will be part of an analysis team to document the user's requirements. Each student brings in a separate list containing what he/she believes the requirements should be. The class discusses all requirements and comes to some agreement. The analysis team leader (grader, teaching assistant, instructor, or student) takes notes on the discussion and prepares a user requirement report for the next meeting. This list will be used throughout the semester to introduce some computing concepts, illustrate some of the lectures, and answer any questions regarding the case study.

To avoid the natural tendency of the students to research the hardware or software before defining the problem completely, the assignment of the specific microcomputer system (or software package) should come only after the requirements have been decided upon. By giving each student a unique system (microcomputer or software), we not only limit copying of work, but also add interest to the project. For the past two years, students have been required to write about two microcomputer systems which could be possible solutions to problems outlined in the scenario.

The CPT 154 students generally have not completed a college composition class yet. Therefore, most are unfamiliar with the department-oriented library system. To reduce the frustration in locating reference materials, a guest presentation by a Purdue librarian provides details regarding library usage for a project such as ours. In the case of CPT 110, most students are familiar with the libraries. They are just given a list of suggested references with associated library locations.

The next major phase of the research project is called the "six definitions." Here, the students identify the subject, purpose, scope, readers, and sources of information, and create a report schedule. The primary goal of this assignment is to make sure all students are completing this case study as "the person in the real world" identified in the scenario, and not as a student completing the project for the instructor.

About five weeks into the semester, the student must turn in an outline and list of references. At this stage, the actual machine details are not expected in the outline, but it should clearly show the type of information the final report would contain. The reference list must be complete, although some changes may be made at a later date. The

primary goals of this milestone are to insure that the student 1) includes all necessary details in the final report, and 2) works progressively instead of waiting until the last minute.

The third milestone is essentially the final draft in rough form. Since CPT 110 is a small class, the instructor (or teaching assistant) meets individually with each student. Together, they verify that all the requirements are included. This conference is extremely beneficial for student and instructor alike. It gives the student the opportunity to ask questions and to clarify points. It gives the instructor occasion not only to make suggestions but also to know the student better. It also helps in the final grading process -- the teacher already knows the content of the paper and recommended changes, the student knows exactly what must be done to get a good grade. There is no after-grading bitterness.

Professionalism is expected from the final paper. In addition to the body of the report, a cover page, table of contents, appendices, a keyword glossary, endnotes and a bibliography must be included. Students are not only graded for the informational contents, but also for the quality of the work.

In CPT 110, the last week of the semester is reserved to oral presentations. Each student, makes an eight minute presentation to the class. The student plays the role of the employee, while the class represents the company. The student has to summarize the requirements, review and compare the two microcomputers and give recommendations. The talk should have a persuasive tone and should not be too technical. The student has to keep in mind that the members of the company have little computer background. Presentations are videotaped, and the tape is available for review. The students in the audience receive extra points for asking pertinent questions. Each student is required to ask at least one question per session. This forces the whole class to participate in the presentations, encouraging the speakers, and avoiding the too easy "attend and forget syndrome." [2] Since each student has a different set of microcomputers to review, the presentations complement each other and the level of interest remains high.

CONCLUSION

The research case study not only reinforces the learning experience, but also encourages students to seek answers to questions that arise, forces them to get acquainted with professional journals and creates "an awareness of the profession and a sense of identification with the computer science community." [3] If some students show some reticence at the beginning of the semester in writing or making an oral presentation in a non-English or non-Communications course, most of them are easily convinced that this case study is a great learning experience and that in the business world people need to be able to communicate. To be able to communicate in their technical field, they need good writing and oral skills. Most of the students have shown great enthusiasm with this project and are very proud of their final professional report. It gives them a feeling of achievement. They realize that they not only have learned a vast amount of computing concepts but also were able to use this knowledge to solve a real world problem.

REFERENCES

[1] Data Processing Management Association. "CIS Curriculum '86: Working Document -- Second Draft." A revised working Document for the CIS Curriculum '86 list of courses. April 12, 1985.

[2] Karten, Naomi "End User Demand Requires New Approach to Training." Vol. 24, No. 5, Data Management, May 1986, pp.10-12, 19.

[3] Taffe, William J. "Teaching Computer Science Through Writing." Vol. 18, No. 2, SIGCSE Bulletin, June 1986, pp.82-83.

[4] Brender, Susan et.al. "Documentation writing: Practice makes perfect." Vol 6, No. 3, Data Management, February 1987, pp.16-17.

[5] Harriger, Alka Rani and Tom I. M. Ho. "A Data Processing Communication Skills Course." Vol. 17, No. 1, SIGCSE Bulletin, February 1986, pp. 97-101.

SHADOW ENTITIES:
LESSONS LEARNED FROM A DATABASE DESIGN PROJECT

Judith D. Wilson
University of Cincinnati

ABSTRACT

Open-ended database design projects can be used to introduce information systems students to the types of problems they are likely to encounter as designers and builders of database systems. One of these projects resulted in more difficulties than the instructor could foresee. The paper examines how this happened and suggests an approach to handling such situations.

INTRODUCTION

Undergraduate students in computer science and engineering and computer information systems programs will be the primary designers and builders of computer-based information systems. Consequently, academic computing programs that train such students must prepare them to effectively handle realistic problems. This requires mastery of techniques and tools for the analysis, design and building of such systems [4], but it also requires exposure to open-ended problems that have not been used before [1].

A project-focused database course has been designed and taught in the undergraduate information systems program of a large state university with these very general educational and training objectives in mind. Although this may be justifiable from the point of view of training designers of information systems intended for complex and evolving environments, the use of open-ended projects can have pedagogical difficulties. This paper describes a project that involved unforeseen conceptual data modeling difficulties and attempts to establish a reasonably good understanding of how and why these difficulties occurred.

GENERAL PROJECT CHARACTERISTICS

The course teaches database design to students who are already familiar with the principles of database management. One of the course objectives is to give students their first experience with an open-ended problem, the kind of problem they are likely to encounter in the analysis and design of large database systems. The pedagogical strategy is to introduce design tools and practices in a straightforward way with simple examples and to have students apply them to a design problem which is not so simple or straightforward. The design project is the central activity that unifies the course content and provides motivation to understand and master the use of design tools and practices.

Teams of three or four students use the entity-relationship diagramming technique to develop a conceptual model of the project enterprise. This technique is used because it is easy to teach and encourages a top-down design approach [3, 5, 6, 7, 9]. Conceptual models are later mapped into relational or network DBMS-compatible implementation models, and an internal model is sketched.

A PROBLEM-RIDDEN PROJECT

One particular project challenges students to design a database system for

part of an enterprise that will eventually be fully automated. The enterprise is a daily newspaper in a large midwestern city, and the focus of the project is the newspaper library which maintains historical files of graphics, photographs and clippings for ongoing reference (see Appendix).

The course project describes the library system as part of a larger project to automate all newspaper operations. Inputs to the library system are sometimes described in terms of this larger environment (For example, "total advertising inches ... for each page of the previous day's paper will be made available to the library once operations have been computerized."). In addition, discussions of the design project often refer to the information needs and activities of other aspects of the enterprise. Yet the project does not require students to integrate the library "view" into a larger model of the newspaper enterprise. As it turns out, this situation introduces unexpected difficulties into the process of conceptualizing and modeling entities and relationships.

A SHADOW ENTITY

An objective of the library database subsystem is to restrict access to library items or information based on whether the user is an editorial or non-editorial staff member, or neither (some library information is available to the general public). Complete information about staff members will be maintained elsewhere in the newspaper enterprise, but the library subsystem only needs that information about users necessary to determine access privileges to library information.

The user is referenced in the library requirements analysis document (the problem description), and would be expected to play a prominent role in a librarian's descriptions of library operations (Ex., "Requests for statistical reports ... can be made only by ... the editorial staff", "... staff members have access to all library items."). In this case, however, the only information that is required about the user is a type attribute or password to support access privilege decisions. If the user is modeled as an entity, it probably will not be mapped into a logical record type, and the relationships in which it participates, for example "accesses" or "requests", will map neither to access paths in the implementation model, nor to linkage paths in the internal model. Thus, from the point of view of the library subsystem, the user is at most a "shadow" entity.

The three student design teams that worked on this project had modeling difficulties expected for projects of this type [8, 9]. In addition, there were difficulties which can be associated with the problematic user entity. Several of these difficulties are illustrated by the diagram shown in Figure 1.

In all cases, the design teams introduced a general (public) user subtype into their models. The general user subtype is fully extraneous to the library enterprise since it has no attributes (with the possible exception of a system-supplied key). Moreover, inclusion of the general user entity encourages the designers to model retrieval phenomena, such as "lists", "displays" and "prints", as relationships between the general user and the library items. Finally, as the number of apparent information links (or relationships) increases between user subtypes and library items, it becomes more difficult to notice that real relationships, such as the association between "article" and "graphic" entities, have been omitted.

In an actual database environment, the user would probably be virtual in a set of applications (or external views). At most, this would require a table lookup for password or ID verification. The only candidate for a real relationship between user and library item entities is "loan request". If the library system is to keep track of loans then "borrows" can be modeled as a relationship between "user" and "visual" entities, or as a derived (or "composite") entity if loan information will include dates or other information specific to loans [2]. User information needed to keep track of such loans could be obtained from staff files kept and managed elsewhere in the enterprise.

How should the user be conceptualized within the library subsystem? Is it an eventual "plug-in" point for integration of views across the newspaper organization? If so, how should the associations between user and library items be modeled?

The general problems seem to be the following. If the user entity, along with its subtypes, is omitted, the information structure may be fragmented. In Figure 2, which presents a possible model of the library subsystem, user subtypes integrate "visual", "article" and "page summary" entities. If user entities are included to avoid fragmentation then it is tempting to model a complete user entity (with a complete set of user attributes) and this encourages modeling processes such as "lists", "requests" and "accesses" as relationships between user and library item entities as shown in Figure 2.

The actual library subsystem information structure seems to be captured in Figure 3, where "Staff" is a plug-in point for later view integration. This fact is indicated on the graph by shading the "Staff" entity. The diagram also contains two entity-relationship graphs since the entity "Page Summary" is not associated with any of the other library item entities.

WHAT WENT WRONG?

As remarked earlier, modeling a subsystem without attempting to integrate it into a larger enterprise model invites difficulties. The most significant problem, however, may be the seductive effect that the subsystem view can have on the construction of the project problem description. In this case, the instructor also had fallen victim to a change in perspective, the shift from newspaper enterprise to newspaper library subsystem. Thus the problem description includes as update requirements, "Add a new staff member" and "Delete a staff member", which are not relevant to the library subsystem insofar as they imply that attributes other than ID must be included.

Scanning initial versions of entity-relationship diagrams for problems is also more difficult than usual. A quick scan is not sufficient although many types of problems can be anticipated based on experiences with previous projects. In this case, it was difficult to notice that the user entity subtypes were minimal, or "shadow", entities, and that inclusion of these entities introduced processes into the model which seemed necessary to avoid fragmenting the diagram.

The problem was sensed by a puzzled student who wanted to see the user as an application phenomenon. A closer look at entity attributes which were not fully defined in initial versions of student models confirmed suspicions raised by the student's questions.

LESSONS LEARNED

The lessons learned from this exercise in open-ended project management are two. First, database design problems that implicitly involve the integration of user views introduce difficulties that cannot be handled by a simple manipulation of available design tools and past practices. In this case, the entity-relationship approach does not provide a way to explicitly model minimal entities or plug-in points for later view integration. The problem here suggests the kinds of difficulties one might have describing a phenomenon using a language that lacks requisite structures and vocabulary. The description fails unless the language can be extended in an intelligible way.

The second lesson is a pragmatic one. The instructor may not fully understand the scope and nature of an open-ended design problem at the onset of the project. Thus, it is mandatory to maintain a flexible attitude, to approach the design project as though part of the design team, to take part in the discovery process, and to be willing to admit previous mistakes. If student's see that progress can be made when analyzing situations that are not well-understood, and that rebounding from erroneous assumptions is essential to this progress, a very valuable lesson will be learned about how to design systems for real-world applications.

REFERENCES

[1] Booth, T., et al., "Design Education in Computer Science and Engineering", COMPUTER, Vol. 19(6), pp. 20–27 (June 1986).

[2] Chen, P.P. "Database Design Based on Entity and Relationship". PRINCIPLES OF DATABASE DESIGN, VOLUME I: LOGICAL ORGANIZATIONS (Yao, S.B. Ed.). Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1985).

[3] Chrisman, C. & B. Beccue. Entity Relationship Models as a Tool for Data Analysis and Design. ACM SIGCSE BULLETIN, Vol.18, pp. 8–14 (February 1986).

[4] Henderson, P.B., "Anatomy of an Introductory Computer Science Course", ACM SIGCSE BULLETIN, Vol. 18(1), pp.257 –262 (February 1986).

[5] Howe, D.R., DATA ANALYSIS FOR DATA BASE DESIGN, Edward Arnold, Baltimore, Maryland (1983).

[6] Jarvenpaa, S.L. & J.J. Machesky, "End User Learning Behavior in Data Analysis and Data Modeling Tools", PROCEEDINGS OF THE SEVENTH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS, pp. 152–167 (December 1986).

[7] Juhn, S.H., & J.D. Naumann, "The Effectiveness of Data Representation Characteristics on User Validation", PROCEEDINGS OF THE SIXTH INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS, (December 1985).

[8] Wilson, J.D., & W.W. Woolfolk, "Conceptual Data Modeling with the Entity-Relationship Approach: Problems Novices Have and How to Avoid Them", (1986), under review.

[9] Wilson, J.D., "Entity-Relationship Diagrams and English", ACM SIGCSE BULLETIN, Vol. 19(1), pp. 26–35 (February 1987).

APPENDIX

NEWSPAPER LIBRARY DATABASE DESIGN PROJECT

A large urban newspaper has a library department which is used as a reference by newspaper staff and other interested parties. Until now, the library staff has manually cataloged story clippings, photographs and graphics each day. The newspaper is in the process of automating its entire operation. Your task is to design a database system for the library. You may assume that the following facts about the library and its holdings have been discovered by a requirements analysis:

1. Library items include opinion columns, stories, photographs, charts, graphs and maps.

2. Approximately 60% of the stories are news stories. The remaining stories are features.

3. Approximately 90% of all photographs appear with stories. The remaining photographs were intended for stories which were not run.

4. Unless otherwise indicated, newspaper staff members have access to all library items.

5. Photographs dated within the previous twelve months can be released on 24 hour loan to newspaper personnel. Other photographs can be borrowed for 5 days.

6. UPDATE REQUIREMENTS

a. Add a story (50/day). Each story has a title, subject, author (if relevant) or source (ex, Associated Press), date, section and page.
b. Add a photograph (25/day). Each photograph is identified by date, subject and photographer. Photographs that appear with stories are also associated with the titles and dates of the stories.
c. Add a new staff member (10/month). Staff members are currently identified by name and department.
d. Delete a staff member (9/month).
e. Archive a story.
f. When operations have been computerized, total advertising and nonadvertizing inches for each page will be made available to the library.

7. INFORMATION RETRIEVAL REQUIREMENTS

a. Bi-monthly statistics on stories.
b. Weekly statistics on the percentage of overall inches devoted to advertising within each section.
c. Semi-annual report of the weekly fluctuation of advertising activity within each section and across sections.

NOTE: Requests for statistical reports on stories and advertising can be made only by members of the editorial staff.

d. An online listing of stories that are identified by title, date (day, month, and/or year), subject keywords, author or source, department, or any combination of these. (200/day)
d. Online display of any non-archived story.
f. A printout of any non-archived story.
g. Access to any archived story (5/week).
h. Online listings of photographs, charts, graphs OR maps identified using the same techniques as used to identify stories. (20/day)
i. Online requests for loans of photographs, charts, graphs or maps. Loans are made to newspaper personnel only. (10/day)

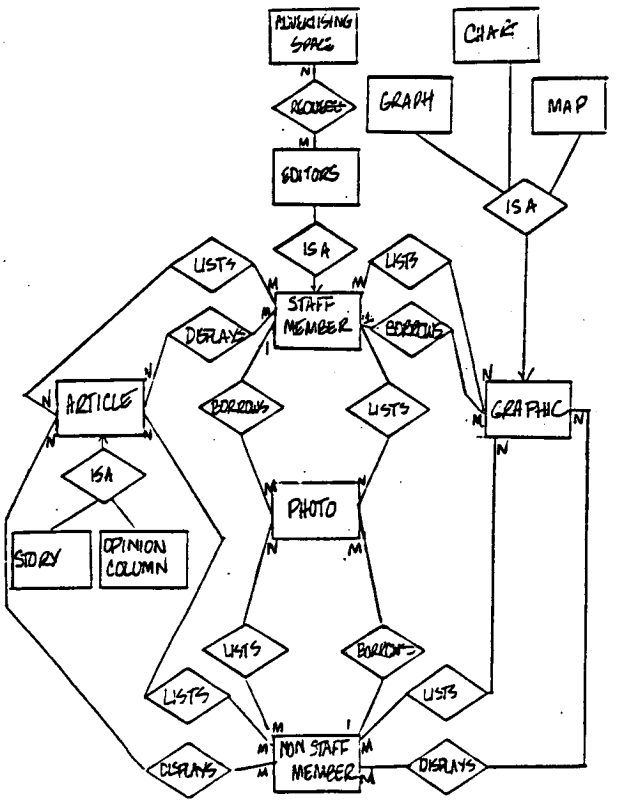


Figure 1. Entity-relationship diagram developed by a student design team for the newspaper library project.

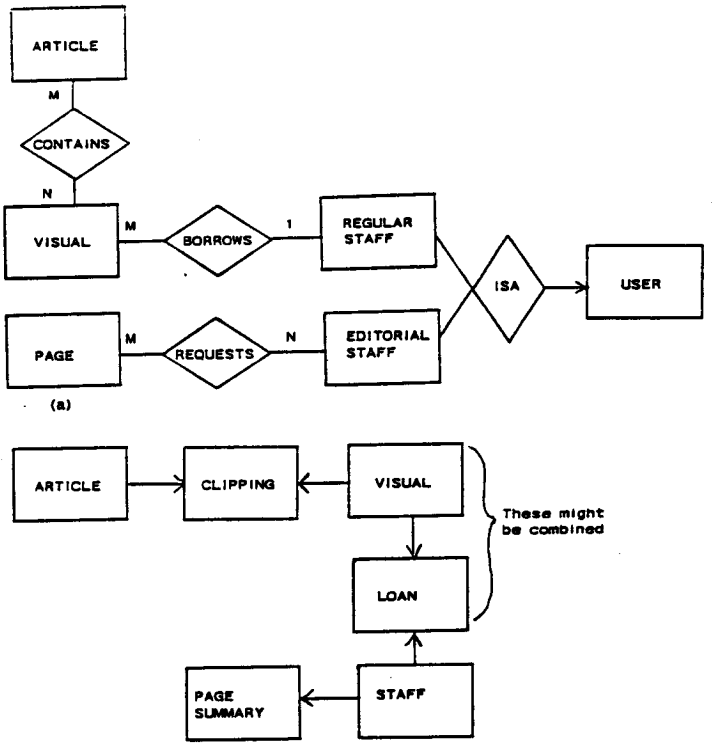


Figure 2. A solution to the project problem that combines sections of the model with a minimal user entity. Part (a) shows an entity-relationship diagram of the library. Part (b) shows a logical record map that might translate the diagram of part (a).

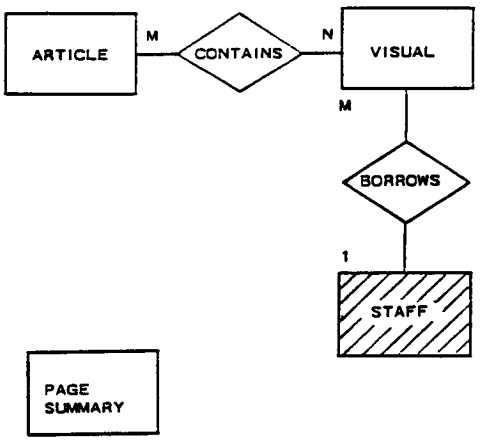


Figure 3. An alternative solution to the project problem. This diagram contains two entity-relationship graphs. The staff entity is shaded to indicate its role as a plug-in entity for later integration of the library "view" into a larger enterprise model.

DATABASE COURSES: ARE THEY HITTING THE MARK?

J. K. Pierson, Associate Professor
Karen A. Forcht, Assistant Professor
James Madison University
Jack D. Shorter, Assistant Professor
James Madison University

ABSTRACT

The second of a series of surveys in a continuing, longitudinal study provides information on the trends in importance placed by academicians' on topics in introductory database courses. Little change has occurred in academicians' opinions on topics over a two-year period. The study also reveals that the academicians are no closer to practitioners in their opinions of the importance of topics in introductory database courses than they were two years ago.

INTRODUCTION

The basis of every information system, whether a management information system, a decision support system, an executive support system, or an expert system, is a database. The importance of providing information systems professionals with backgrounds in database theory and practice has been acknowledged by practitioners and academicians alike. The Data Processing Management Association (DPMA) and the Association for Computing Machinery (ACM) have included a course on data files and databases in their model curricula intended as a guide for educators in planning degree programs in the computer information systems field [3] [6].

Periodic reviews of business courses are necessary if schools are to keep attuned to the needs of business and industry and are particularly important in dynamic fields such as computer information systems. Recently concern has been expressed both by academicians and practitioners that in some cases information systems degree programs do not include the appropriate computer experience necessary for graduates to be well prepared to embark on careers as professionals [2] [4] [10].

A LONGITUDINAL STUDY OF DATABASE COURSE CONTENT

The second of a series of surveys in a continuing longitudinal study has recently been completed that provides information on the importance placed by instructors on database course topics in schools accredited by the American Assembly of Collegiate Schools of Business (AACSB).

The study reported in this paper has two purposes: (1) to provide information on trends in coverage of topics in database courses and (2) to provide a basis for comparison of the importance assigned to topics by practitioners and educators. It is hoped that the findings provide the basis for discussion among academicians and practitioners as they continue to work together to plan adequate preparation for students studying for information systems careers.

RESEARCH METHODS

The first survey in the study of database academicians was undertaken in the spring of 1984. A questionnaire was sent to each of the deans of 236 schools accredited by the AACSB. A letter accompanying the questionnaire requested the dean to forward the material to the appropriate faculty member for a response. The survey instrument included 27 topics recommended for inclusion in introductory database courses in the model curriculum of either the DPMA or the ACM. Some of the topics overlap—a result of using topics recommended by two sources. The respondents were asked to rate each topic on a five-point scale as very important, important, of average importance, unimportant, or very unimportant. Rankings were obtained by assigning a score to each topic figured on the basis of five points for every

response of "very important," one point for "very unimportant," and two, three, or four points for the intervening responses. The response was excellent, with 159 questionnaires being returned, a response rate of 67.4 percent.

A second survey of educators was conducted in the spring of 1986. The survey instrument was again sent to deans of AACSB-accredited schools with the request that it be forwarded to an appropriate faculty member for a response. Of the 241 institutions surveyed in 1986, 119 responded, a 49.6 percent response.

The results of the 1986 survey of academicians were compared with information from a survey of database practitioners conducted in 1984. Database managers in 467 businesses were asked to rate 27 database topics as to their importance. Responses were received from 139, a 30.0 percent response rate [8].

FINDINGS

The findings are presented in three parts: (1) 1986 rankings of importance placed on introductory database course topics by academicians, (2) comparison of importance rankings of topics by educators in 1984 and 1986, and (3) a comparison of the importance of topics rated by educators and practitioners. It should be noted that the responses relate only to the topics' inclusion in introductory database courses. The findings cannot be generalized to other courses.

ACADEMICIANS' IMPORTANCE RATINGS OF DATABASE COURSE TOPICS—1986

Responses on topics are given in Table I and provide a profile of the importance placed on database course subjects by instructors of those courses. The topics are listed in rank order from the topic with the highest importance score to the lowest. For each topic, the rank and importance score are also shown.

The two topics with the highest rank, with scores of 405 and 382, deal with relational databases. The third-ranked topic is Network Data Model with a score of 333, a 49 point drop. The magnitude of the difference in scores between the second and third items clearly indicates the high relative importance placed on the relational topics ranked first and second.

The second largest difference in importance scores sets off the last ranked topic from the topics that rank higher. Character Codes is the topic ranked 27th; the importance score for that topic is 45 points lower than the 26th ranked topic, Basic Machine Architecture. Again, the size of the difference between the 26th and 27th ranked topics is a clear indication of instructors' views that character codes is the least important topic listed.

COMPARISON OF ACADEMICIANS' RATINGS OF DATABASE COURSE TOPICS: 1984-86

The 1986 findings are compared to those from the 1984 survey to provide an indication of trends in the importance placed on topics in introductory database courses. Table II lists the topics in the 1986 rank order of preference and shows the ranking and importance score of each item obtained in 1984 and 1986. The last column in the table provides a description of the change in rank: "none" for no change in rank, "+" meaning a higher rank in 1986 than in 1984, and "-" meaning a lower rank in 1986 than in 1984. Note in Table II that the three topics ranked highest and the two topics ranked lowest in 1984 retained their rankings in 1986.
An analysis of the rankings of 1984 and 1986 reveals little significant change. Only two topics that decreased more than two rank placings between 1984 and 1986: (1) Use and Management of Databases—3 places lower, and (2) Use of High-level, User-Oriented Languages—7 places lower. Two topics also increased more than two rank placings between 1984 and 1986: (1) Data Environment, Managing and Defining Data—4 places higher, and (2) Applied Data Structures—6 places higher.

COMPARISON OF TOPIC RATINGS BY EDUCATORS AND PRACTITIONERS

Table III reflects rankings by practitioners of the same topics ranked by educators in 1986. Academicians ranked Relational Data Model and Relational Systems as the most important of the topics to be included in introductory database courses. The ratings by practitioners placed

those same two topics in tenth and eighteenth places. The top two rankings by practitioners went to Use and Management of Databases and An Overview, topics that were ranked eighth and seventh in importance by educators. Also of interest is the fact that practitioners ranked all data models (relational, hierarchical, and network) as less important than did educators. The differences in ratings between practitioners and educators indicates a substantial difference in viewpoints between the two groups.

DISCUSSION

The following conclusions were reached after analyzing the results of the study of the importance placed on introductory database course topics by educators:

(1) Relational database topics are considered the most important by instructors of introductory database courses of the topics suggested by DPMA and ACM. One possible explanation for the emphasis on relational topics may be that in many schools relational DBMS are used for student assignments. Another reason may be the fact that relational database technology is newer than network or hierarchical technology and that much is currently being written on the subject.

(2) Instructors of introductory database courses place little importance on character codes as a topic to be covered in introductory database courses. This lack of perceived importance, it should be remembered, relates only to the database course. This study does not indicate whether instructors place importance on the topic being covered in other courses in the computer information systems area.

(3) There has been little change in the importance placed on introductory database course topics in the last two years. Most topics either retained their rank or changed only one or two rank placings.

(4) There appears to be a divergence of opinion between practitioners and educators on the relative importance that should be placed on topics included in introductory database courses. Relational topics are rated as much more important by educators than by practitioners. Data models also are rated higher by educators.

REFERENCES

[1] Adams, D. R., and Thomas H. Athey (eds), "DPMA Model Curriculum for Undergraduate Computer Information Systems Education," Park Ridge, Illinois: Data Processing Management Association, 1981.

[2] Chen, Jim, and John A. Willhardt, "A survey of Computer Systems Utilized by Business Schools," The Journal of Computer Information Systems, Volume XXVI, No. 3, Spring, 1986.

[3] "DPMA Model Curriculum for Undergraduate Computer Information Systems Education, 1986" Park Ridge, Illinois: Data Processing Association, 1986.

[4] Greenwood, Frank, Roger J. Deveau, and Mary Greenwood, "Educating Systems Professionals," Journal of Systems Management, Vol. 37, No. 1, January, 1986.

[5] Malmrose, Kirk L., and Robert P. Burton, "Data Files and Databases," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, Georgia, 1986.

[6] Nunamaker, Jay F., Jr., J. Daniel Couger, Gordon B. Davis, "Information Systems Curriculum Recommendations for the 80s; Undergraduate and Graduate Programs, Communications of the ACM, Vol. 25, No. 11, November, 1982.

[7] Pierson, J. K., Jack D. Shorter, William Moates, "Practitioners vs. Academicians' Views on Database Course Content," Proceedings of the Society for Data Educators Conference, 1987.

[8] Pierson, J. K., Jack Shorter, Jeretta A. Horn, G. Daryl Nord, "Profile of Introductory Database Courses Offered at AACSB-Accredited Institutions," Proceedings of the Fourth Annual Information Systems Education Conference, Houston, TX, 1985.

[9] Shorter, Jack D., "Database Management: Current Database Configuration: Data Managers' Opinions of Topics in Database Courses, The Journal of Computer Information Systems, Vol. XXVI, No. 3, Spring, 1986.

[10] Walton, LeRoy, "Computer Science Majors Cannot Live by Micros Alone," Data Management, Vol. 23, No. 2, February, 1985.

Table I
Academicians' 1986 Importance Ratings
of Topics in Introductory Database Courses

Importance Rank	Topic	Importance Score
1	Relational Data Model	405
2	Relational Systems, Relational Databases	382
3	Network Data Model	333
4	Database Administration	332
5	Database Management Systems	326
6	Data Environment, Managing and Defining Data	324
7	Overview	323
8	Use and Management of Databases	322
9	Data Model Overview, DDL, DML	317
10	Applied Data Structures	313
11	Hierarchical Data Model	312
12	Role of Database Information Systems	306
13	Integrated Databases	305
14	DBMS Evaluation	304
15	Use of High Level, User-Oriented Languages	303
16	Distributed Databases	301
17	Basic Technical Concepts for Data	276
18	Direct File Organization	267
19	Indexed Organized Files	265
20	System Resources for Data	235
21	Storage Device Characteristics & Physical I/O	230
22	Memory Management	218
23	Searching and Sorting Techniques	209
24	Dynamic Storage Management	200
25	Operating Systems Topics	197
26	Basic Machine Architecture	192
27	Character Codes	147

Table II
Comparison of Academicians' 1984 and 1986 Ratings
of Introductory Database Course Topics

—1986—		—1984—			Rank Change
Rank	Score	Rank	Score	Topic	1984-86
1	405	1	435	Relational Data Model	None
2	382	2	400	Relational Systems, Relational Databases	None
3	333	3	398	Network Data Model	None
4	332	6	375	Database Administration	+2
5	326	4	389	Database Management Systems	-1
6	324	10	362	Data Environment, Managing and Defining Data	+4
7	323	7	373	Overview	None
8	322	5	379	Use and Mgmt of Databases	-3
9	317	9	369	Data Model Overview,DDL,DML	None
10	313	16	327	Applied Data Structures	+6
11	312	12	348	Hierarchical Data Model	+1
12	306	11	357	Role of Database Information Systems	-1
13	305	14	340	Integrated Databases	+1
14	304	13	346	DBMS Evaluation	-1
15	303	8	370	Use of High Level, User-Oriented Languages	-7
16	301	15	335	Distributed Databases	-1
17	276	19	295	Basic Tech Concepts for Data	+2
18	267	17	310	Direct File Organization	-1
19	265	18	303	Indexed Organized Files	-1
20	235	21	262	System Resources for Data	+1
21	230	20	265	Storage Device Characteristics & Physical I/O	-1
22	218	24	229	Memory Management	+2
23	209	22	251	Searching and Sorting Tech.	-1
24	200	25	224	Dynamic Storage Management	+1
25	197	23	229	Operating Systems Topics	-2
26	192	26	214	Basic Machine Architecture	None
27	142	27	184	Character Codes	None

Table III
Comparison of Practitioners' and Academicians'
Rankings of Importance of
Introductory Database Course Topics

Practitioners Rank	Topic	Educators Rank
1	Use and Management of Databases	15*
2	Overview	7
3	Database administration	4
4	Data environment, managing and defining data	6
5	Database management systems	5
6	Integrated databases	13*
7	Role of database information systems	12
8	DBMS evaluation	14*
9	Indexed organized files	19*
10	Relational systems, relational databases	2*
11	Applied data structures	10
12	Use of high level, user oriented data lang	15
13	Distributed databases	16
14	Basic technical concepts for data	26*
15	Direct file organization	18
16	Data model overview, DDL, DML	9*
17	Storage device characteristics & physical I/O	21
18	Relational data model	1*
19	Searching and sorting techniques	23
20	Systems resources for data	20
21	Hierarchical data model	11*
22	Network data model	3*
23	Dynamic storage management	24
24	Operating system topics	25
25	Character codes	27
26	Memory management	22
27	Basic machine architecture	26

* Differences in rank of 5 or more places

USE OF PRESENTATION GRAPHICS IN CIS/86-6
DATA FILES AND DATABASES

Dr. John C. Shepherd
Dr. Thomas A. Pollack

Duquesne University
School of Business and Administration
Pittsburgh, PA 15282

ABSTRACT

Use of presentation graphics within organizations is increasing. By making effective use of graphics, color, sound, and other special effects, this tool can add impact and interest to presentations. Specifically, this paper looks at presentation graphics and at how to use it in CIS-86-6.

OBJECTIVES

We will attempt to accomplish the following objectives:

(1) Define presentation graphics;

(2) Describe the functions of presentation graphics;

(3) Show how presentations graphics can be used in CIS-86-6, Data Files and Databases;

(4) Compare four presentation graphics packages; and

(5) Demonstrate presentation graphics by using it to deliver this paper.

WHAT IS PRESENTATION GRAPHICS?

Presentation graphics is a collection of computer-based hardware and software tools for making effective audio-visual presentations. While available on a wide range of hardware, we will only discuss such packages that run on IBM and IBM-clone microcomputers.

To use such a tool, one must develop several graphics screens by drawing images using the cursor keys, a mouse, and a clip-art library of pre-drawn symbols and images. Alternately, one can use a scanner or graphics tablet to digitize existing images from photographs, documents and so on. Yet another approach is to "capture" screen images from other programs and save them as disk files.

Using a cut-and-paste technique, one image (slide) can be divided into portions, and a portion at a time can be shown. Text can be inserted in any size using multiple fonts and colors anywhere on the screen.

After the images are named and saved on a disk, their presentation sequence must be decided. In addition, the method of transition from "slide" to "slide" must be chosen. Typically one can choose from diagonal, horizontal and vertical wipes, and explosions and pushes, where one image "pushes" the previous one off the screen. The time it takes to perform any slide-to-slide transition can also be varied, so this must be decided also.

Next, the viewing duration of each of the slides is chosen. Optionally, the user can be instructed to press a key to continue.

HARDWARE REQUIREMENTS

The software packages require a CGA or EGA monitor and an appropriate controller because they output in graphics mode. Many require a mouse or at least allow a mouse for drawing. The CGA option provides for composite (TV) output, permitting the video taping of the results or the broadcasting over conventional televisions.

Typical total hardware cost: $450-$2,000

SOFTWARE CAPABILITIES

(1) Screen capture

(2) User-defined slide shows

(3) Branching based on user-input

(4) Multiple fonts

(5) Clip art library

(6) Edit graphics produced from other programs

(7) Sound

(8) Animation

SCREEN CAPTURE

This is the ability to save screens from any program (e.g., you are running Lotus and want to save a screen, both text and graphics, for inclusion with your slide-show).

After saving a screen, one can use the presentation graphics package to modify it, adding text, enlarging portions of it, altering colors; changing resolution and so on.

USER-DEFINED SLIDE SHOWS

This is the primary goal of presentation graphics: One can create multiple screens-full of graphic information. Each screen is called a slide. A slide show is the ability to show your screens (slides) (1) in any sequence, (2) for any length of time, (3) with animation, (4) with dissolves, wipes, explosions, etc. from one slide to the next.

BRANCHING BASED ON USER INPUT

Permits users to branch to different locations within a slide show. Based on a menu selection, the presentation graphics system displays a corresponding sub-slide show, then returns for another menu choice. This is very useful as a learning device, because a student can view lessons as needed.

MULTIPLE FONTS

All the packages support fonts such as: computer, Roman, thin, bold, Old English, etc.

CLIP ART

This is a library of graphics already available without your drawing them. These libraries vary in complexity, usefulness and size. Most useful for our needs are computer-related symbols.

EDIT GRAPHICS

The ability to import graphics output of other programs (e.g., saved Lotus graph) and then change the graph.

SOUND

The ability to change pitch, frequency, etc. Some packages even permit digitized voice, music, and special effects.

HOW TO USE PRESENTATION GRAPHICS WITH CIS-86-6

Presentation graphics is particularly useful for topics requiring animation - at ISECON '87 we presented slide shows for:

(1) Reviewing the program compilation process;

(2) Showing data moving from a disk drive to a controller, to a channel to the buffer, then to the FD in the DATA DIVISION of a COBOL program;

(3) Showing relational joins. The presenters showed two tables being searched and matched, and the resulting table; and

(4) Demonstrating how a general database lecture can be enhanced using presentation graphics.

COMPARISON OF PRESENTATION GRAPHICS PACKAGES
(As of March 1987)

IBM PC STORYBOARD

(1) Small clip art library (320 images)

(2) Limited hard copy (e.g., the only color printer choice is the IBM color printer)

(3) 4 Chart types (pie, line, etc.)

(4) 16 colors

(5) 4 fonts

(6) Can capture only medium resolution CGA graphics from other programs

(7) No mouse option

(8) No royalty for run-time

(9) $275

SHOW PARTNER
Bright Bill - Roberts
120 E. Washington Street
Syracuse, NY 13202
(315) 474-3400

(1) Can capture text and graphics

(2) Mouse support

(3) No graph generator

(4) Easy to use

(5) 320K RAM required

(6) Interactive

(7) 3 sound effects plus interface to digitally record and playback voices, music, and sound effects from a disk

(8) Run-time can be used without royalty.

(9) Small clip art library (200)

(10) EGA, CGA, Hercules

(11) Hard copy - 60 printers, Polaroid palette

(12) 16 colors

(13) 20 fonts

(14) $79

EXECUTIVE PICTURE SHOW
PC Software of San Diego
11627 Calamor Court
San Diego, CA 92124
(619) 571-0981

(1) 7 graph types (cannot import data)

(2) Difficult to use - must pre-allocate buffers

(3) $35 option for clip art (450 images)

(4) CGA only

(5) 4 colors

(6) 10 fonts

(7) $245

EXECUVISION,
Visual Communications Network
230 Main Street
Cambridge, MA 02162
(617) 497-4000

(1) An older program without many features

(2) Really just a paint program

(3) Difficult to use

(4) 256K RAM

(5) Optional huge library (16 disks, 2500 images)

(6) 16 colors

(7) 11 fonts

(8) $395

CONCORD - (Same company)

(1) Large clip art library (2500 images)

(2) High Res graphics

(3) EGA, CGA

(4) 64 colors

(5) 17 fonts

(6) 384K RAM required

(7) Charts

(8) Image capture - good - anything

(9) Paint - limited - move, erase, "spray"

(10) Slides

(11) Reads 1-2-3 spreadsheets

(12) 15 fonts including color, size, orientation

(13) 2 MB or disk storage

(14) Need 640K RAM for high res graphics

(15) Can add tunes

(16) $695

SUMMARY

Using a presentation graphics package requires a commitment in hardware, software and time. At a minimum, a CGA-based PC, together with the appropriate software must be acquired. Next, the learning curve must be climbed; allow 40-50 hours to learn and feel comfortable with the presentation graphics software. Finally, the slide show must be constructed one slide at a time. Allow about 1 hour per slide, but from one slide, several can be derived. Finally a program must be written to sequence the slides, provide the transition method between slides, the length of time to show each slide, etc.

Despite the time commitment, the rewards are worth the investment. Concepts requiring animation can be illustrated as in no other way. Color, graphics, flashing of portions of the screen, and various ways of showing portions of a screen add punch to your lecture, enhance student learning, and result in immense personal satisfaction.

REFERENCES

(1) PC Week, "Advanced Techniques Add Life to the Boardroom Sales Pitch," December 9, 1986.

(2) P.C. Magazine, "Wipes, Pans, and Fade-outs: Animating Your Business Graphics, " Vol. 6, No. 5, March 10, 1987.

(3) InfoWorld , "Presentation Graphics Software," Vol. 8, Issue 38, September 22, 1986.

THE USE OF COMPUTER-AIDED SOFTWARE ENGINEERING TOOLS IN AN MIS CURRICULUM

Donald L. Burkhard, PhD
University of Virginia

Computer-Aided Software Engineering has recently become a hot topic for systems development professionals. Promises of tremendous increases in productivity and software quality are enticing. This paper describes the basic capabilities of CASE technology described and the experiences of implementing CASE into the MIS curriculum at one University.

INTRODUCTION

Over the past several years, interest has grown in the use of Computer-Aided Software Engineering (CASE) both in industry and in academia. Although some CASE tools have been available for almost a decade, only since the early 1980's has the hardware technology existed to really support the use of CASE. This paper discusses the state of the art in CASE tool capabilities and describes how CASE tools can be applied in an academic setting. The paper concludes with a description of the implementation and use of CASE tools in our curriculum.

CASE EVOLUTION -- REVOLUTION

The early case tools were very limited in their capabilities and would not necessarily be considered CASE tools today. Some of the initial tools were merely computer-aided drawing systems that allowed for easy creation and maintenance of the graphical techniques used in many systems development methodologies. CASE has evolved to the point where some products support a whole range of systems development tasks (for example, graphics, data definition, and code generation). As defined by some, the term "CASE" also includes project management support.(1) Of course, various CASE products differ in the actual degree of support they provide for the systems professional, and there is no single accepted definition of CASE.

CASE technology offers a wide range of support for the systems development life cycle (SDLC). A representative list of phases and associated activities of the life cycle are listed below [adapted from (5)]. Many of these activities are prime candidates for automated support:

- Requirement Analysis (define users requirements)
- Logical Design (prepare general design specifications)
- Physical Design (prepare detail design specifications, define subsystems, define database structure)
- Program Design (design and code programs, unit test programs, document programs)
- System Implementation (perform subsystem and system testing, train user personnel, perform system conversion)
- System Operation (operation, maintenance, and evaluation).

CASE technology includes the graphics capabilities which have been traditionally used in the SDLC. These graphics include data flow diagrams, structure charts, and data modeling diagrams. CASE also involves the definition of data and procedures. These definitions are then stored in a centralized repository which can then be referenced from a variety of places providing one common definition. A change in the definition is automatically updated for all related references through the central repository. CASE technology also supports code generation and testing.

The phase of analysis in which requirements and data are defined is referred to as the "front end" of the life cycle. The use of these definitions and the actual creation of code, or programming, is referred to as the "back end" of the life cycle. (3) CASE is capable of supporting all phases of the SDLC. However, existing CASE products generally support either front end or back end activities, and currently there are only a very few vendor offerings that purport to support the entire life cycle.

CASE products can be categorized by the phases of the SDLC that they support. They can also be categorized by the kind of hardware they require: microcumputer, minicomputer, or mainframe. Generally, the microcomputer-based CASE tools are less expensive per workstation than minicomputer and mainframe-based tools. However, the mainframe-based tools offer a more sophisticated method of sharing project and system information among the software development professionals. In addition, most of the microcomputer-based CASE products support the front end functions and do not address code generation, testing, and other backend activities.

The exalted benefits of CASE are profound. For example, one obvious benefit is that diagrams are easily changed and do not require redrawing by hand, a tedious, time-consuming process. Some users have claimed as much as a ten-fold increase in productivity after a very short period of use. Other potential benefits include better software quality, generation of reusable components, better project control and the enforcement of standards and adherence to a structured methodology.

ACADEMIA -- LAGGING OR ON PAR

The use of CASE technology is just beginning to grow. As recently as one or two years ago, little was said about CASE. Today it is perhaps one of the hottest products providing support for the systems professional. It has been said that systems development professionals are slow to adopt the same time-saving technologies that they provide for others in the organization. This appears to apply to CASE, as one author has estimated that only one-half of one percent of potential CASE users are currently using it. (2)

Just as the use of CASE in industry is in its infancy, so is its use in academic programs. One estimate puts the cost of a single CASE workstation, including both hardware and software, at around $30,000. (3) Software prices range from $1,000 to $9,000 per copy. At such a cost, implementing CASE on a one-to-one basis represents a large investment for an organization, even more so for an academic institution. Fortunately, in an academic setting, a one-to-one ratio of workstations to users is not necessary. In addition, many of the CASE vendors offer substantial discounts for educational purposes. Even so, establishing an acceptable workstation/user ratio can be expensive, and it may prove difficult to convince some people of the need for using CASE in the curriculum. As a result, implementing CASE may be infeasible for schools that have large undergraduate programs in MIS.

Another problem that an academic program encounters is how to implement the use of CASE in the curriculum. It must be

decided what role CASE is to play -- whether it is to be a major component which students must use for project development or whether students are to receive only a brief exposure to the technology. In either case, it may be necessary to restructure course content and to develop support material.

Despite the potential barriers to implementing CASE in an MIS curriculum, the author knows of many schools currently using such tools. It is not known how wide-spread the use of CASE is in academic programs, but it is clear that many schools have looked into the use of the tools. The following describes our experiences in implementing CASE into the MIS curriculum.

OUR EXPERIENCE

We are a relatively small undergraduate business school with approximately 630 students and 240 MIS majors (120 per class). In the spring of 1986, we became interested in implementing the use of a CASE tool in our curriculum. Our interest was to include the use of the tool in the students' final year in the program. It was difficult at the time to find information on available CASE products. We contacted the vendors that we could identify regarding educational/academic programs. Several of the vendors had existing programs, while others had not yet considered such programs. Programs varied from a limited number of free copies to discounts on a per-copy basis. Discounts ranged from about 20% to better than 50% of the retail price.

The constraints on our selection were that the product had to run on ATT 6300+ machines. We had just installed a third microcomputer laboratory specifically for advanced MIS work. The lab had nine networked 6300+'s, and we hoped to be able to have CASE tools available on each machine. However, even though resources are generally available, large expenditures could not be justified.

Our selection of a CASE tool actually occurred in two phases. First, we applied to Index Technologies (InTech) for support from their academic program, which consists of providing two copies of their product, Excelerator, to schools which meet specific requirements. Then, after examining the product, we contacted InTech and negotiated for more copies, bringing the total number procured to nine.

During the summer of 1986, we developed a tutorial which leads students through a large portion of Excelerator's features by using an example project. In addition, we restructured the Systems Analysis and Design course so that the tool's features were used in parallel to corresponding course topics.

The CASE tool was introduced in the fall of 1986. The tutorial involved a series of four deliverables. The first set of requirements involved the creation of several levels data flow diagrams. Defining the data flows, stores, and elements comprised the second set of requirements, which was followed by analyzing the contents of the dictionary. Analysis included level checking and consistency of data flows and completeness of data definitions. Finally, a whole set of documentation was required for the described system. The total time necessary to complete all of the requirement was typically 15-20 hours.

After initially learning how to use the tool, students were encouraged to use it for the other projects in the rest of the course and in other courses, particularly the Systems Development course. Reaction from students has been favorable -- they consider the CASE tool fun and easy to use. In addition, the tool encourages adherence to a structured methodology. The projects that are turned in are of better quality and are more complete.

CONCLUSION

In conclusion, CASE tools are still relatively new. Both industry and academia are rushing to implement the use of these tools. There are some barriers to acquisition of the tools, but benefits can be tremendous. Our experience of implementing a CASE tool into an academic setting was received well by students. In addition to being perceived as a fun and useful tool, academically, the tool helped emphasized and enforced the use of structured methodology. Similar outcomes can be obtained in practice as well.

REFERENCES

(1) Case, A.F. Jr.; Information Systems Development: Principles of Computer-aided Software Engineering; Prentice-Hall, Englewood Cliffs, NJ; 1986.

(2) Leavitt, D.; "Design Tools: The Real Starting Point;" Software News; February 1986.

(3) Martin, J. and McClure, C.; "The Latest Look in Programmer Productivity Tools;" Business Software Review; May 1986.

(4) McClure, C.; "CASE;" Proceedings from the First CASE Symposium; February 1987.

(5) Sprague, R.H. Jr. and McNurlin, B.C.; Information Systems Management in Practice; Prentice-Hall; Englewood Cliffs, NJ; 1986.

SOFTWARE DEVELOPMENT TOOLS FOR THE PROGRAM DEVELOPMENT LIFE CYCLE

Laurena A. Burk
Department of Systems Analysis
Miami University
Hamilton, Ohio 45011

Lloyd R. Weaver
Department of Computer Technology
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

This paper will acquaint instructors of information systems with the tools available for the analysis and logic design phases of the program development life cycle and will recommend where these different tools would be most appropriate in the CIS curriculum. To this end, the elements of the program development life cycle will be reviewed with emphasis on the **analysis** of the design specifications and detailed **logic design** phases. The tools will be divided into two groups. The first group of techniques presented support the high level nature of the **analysis** phase. The second set of tools supports the more detailed requirements of the **logic design** phase.

INTRODUCTION

The Program Development Life Cycle ensures software quality, enhances developer productivity, and promotes consistent structured design. The cycle, depicted in Figure 1, consists of the following phases:

(1) **Analysis** of design specifications
(2) **Design** of the algorithm **logic**
(3) Generation of test data
(4) Walkthrough and trace the logic
(5) Implement the design
(6) Walkthrough and trace the code
(7) Test and Debug the program
(8) Documentation (an ongoing product of previous phases)

Architects, as opposed to carpenters, are responsible for the creative portion of developing a structure. The first four phases of the program development life cycle represent the

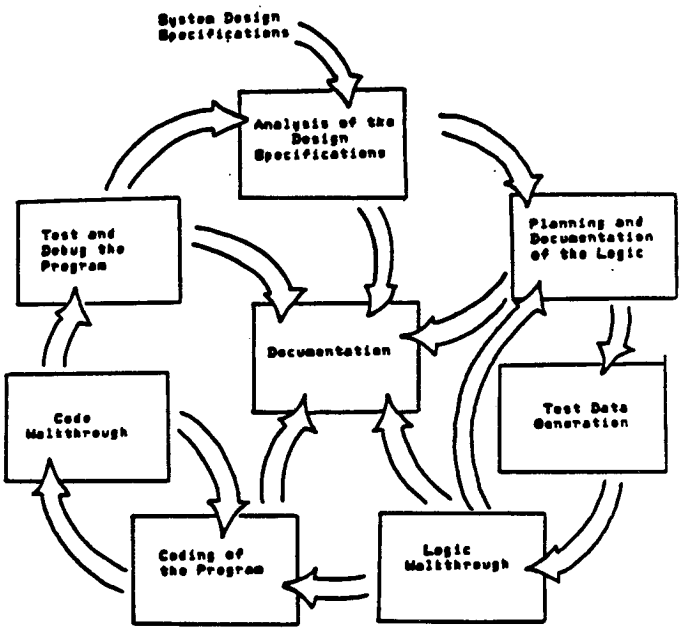


Figure 1.
The Program development Life Cycle

architect's blueprint of an algorithm. The next three phases are the carpentry. Just as architects use the blueprint and scale models to communicate their designs, programmer/analysts require tools to communicate algorithmic designs.

Eight tools are introduced, split into two divisions: those supporting the high level **analysis** phase and those supporting the more detailed **logic design** phase. Figure 2 illustrates this division.

This paper introduces and provides a mini-tutorial for the eight tools and their applications and recommends where the different tools would be most appropriate in the Information Systems curriculum.

ANALYSIS PHASE

HIERARCHY CHART
DATA FLOW DIAGRAM
SYSTEM FLOWCHART
HIPO with STRUCTURED ENGLISH

LOGIC DESIGN PHASE

FLOWCHART
D-CHART
NASSI-SCHNEIDERMAN DIAGRAM
PSEUDOCODE

Figure 2.
Program Development Life Cycle Tools

ANALYSIS PHASE TOOLS

The **analysis** phase seeks to shape a software solution by simplifying the system design specifications and producing a graphic which conquers complexity. The graphic tools used in the **analysis** phase both partition the system design specifications and organize them hierarchically.

THE HIERARCHY CHART

A hierarchy chart is a graphic tool which shows the organization of modules, the relationship between modules, and the function of each module. Figure 3 is a simple hierarchy chart. The rectangular boxes represent

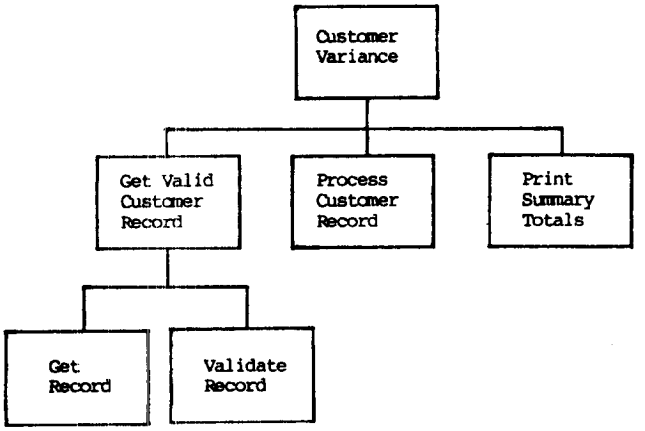


Figure 3.
A Simple Hierarchy Chart

modules. The name of each module is in its box. The name indicates module function. The main control module is at the top. Its immediate subordinates are Get Valid Customer Record, Process Customer Record, and Print Summary Totals. These three modules are invoked by the main control module. Get Valid Customer Record, Process Customer Record, and Print Summary Totals may later be found to be too complex and therefore be decomposed. For example, Get Valid Customer Record may call two modules, Get Record and Validate Record. The most important point to remember in decomposing modules is that siblings, the lower level modules, must contribute to the function of the parent, calling module, or else they are in the wrong branch of the family and should be rearranged. Thus, the relationship and organization of the modules are depicted in the hierarchy chart.

A hierarchy chart is a high level analysis tool because a.) it represents top down structure by partitioning the system design specifications into modules, b.) it depicts module function without detailed inner algorithmic logic, and c.) it depicts the

hierarchical relationships between modules.[8] In the traditional context, **module** implies a one-to-one relationship to a subprogram. Initially, however, the **modules** are logical tasks which emerge during the analysis of the design specifications. Thus, a logical hierarchy chart can be formulated for the simplest specification, even in an introductory course. A typical approach to analyzing a specification might be:

(1) Form "things-to-do" list
(2) Determine inputs given
(3) Determine outputs required
(4) State the processing tasks
(5) Develop a hierarchy chart

The logical hierarchy chart would start out with only four or five logical modules, as shown in Figure 3. The dotted line in Figure 3 separates the first cut hierarchy chart, which parallels the things-to-do list, from a second cut hierarchy chart in which the task Get Valid Customer Record requires two subtasks, Get Record and Validate Record. Although the beginning programmer may not design the algorithmic logic for four separate modules in a first assignment, the later transition to physical modules to support the Input, Process, and Output functions is made smoother by the hierarchy chart as a graphic high level tool for analyzing specifications. As the hierarchy chart is developed, the seeds of structured analysis and design are sown. The name of each logical module denotes a single independent task. As specifications become more complex, the modules are decomposed into subordinate modules which support the task of the parent module. The use of the hierarchy chart in the analysis phase encourages development of independent, logical modules which can evolve into codable, transferable physical modules. The hierarchy chart is an input to the design of the algorithmic logic step in the program development life cycle. The final cut hierarchy chart for the Customer Variance problem which is described in the logic design phase section of this paper is in Figure 4. Notice that the

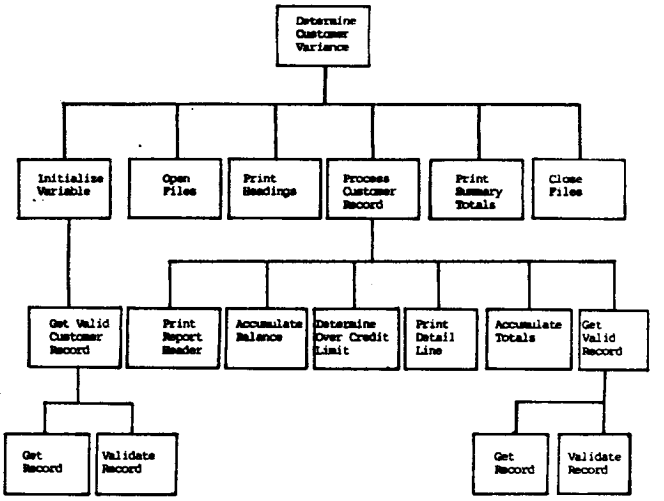


Figure 4.
The Final Hierarchy Chart for the Customer Variance Problem

child modules support the task of the parent. The details, such as initializing, opening and closing files have also been added. The stepwise decomposition from Figure 3 to Figure 4 results in a high level, concise graphic of the Customer Variance problem, needed to begin logic design.

The hierarchy chart has traditionally been used to represent physical modules rather than to analyze program specifications. As an analysis tool, the hierarchy chart can parallel the concepts of input, process, and output in the first weeks of a programming course. Koffman's introductory Pascal text[5] presents the hierarchy chart as an analysis tool in the second chapter. As a bridge to algorithmic design, the hierarchy chart is a graphic which parallels the topic of subprograms in an introductory course. Later in the curriculum, probably in the third or fourth course, the student of program design will be better prepared to accept the analyst's data flow diagrams and the processes for converting a data flow diagram to a hierarchy chart because of the early exposure to the

hierarchy chart as an analysis tool. Thus, the hierarchy chart is invaluable to the transition from analysis to logic design.

THE DATA FLOW DIAGRAM

The data flow diagram (DFD) is a graphic representation of the user requirements. If properly developed, the data flow diagram will insure a programming project's success. The data flow diagram is part of the structured design specification created by the systems analyst. The data flow diagram is the most logical tool of the analysis phase of the program development life cycle. It is the logical model of the user requirements which is transformed into hierarchy charts, system flowcharts and HIPO charts. It is generally the system designer's job to transform the logical data flow diagram into the hierarchy chart and the algorithm descriptions.

While working within the program development life cycle analysis phase, it is important to understand the DFD rather than to create a DFD. The data flow diagram is a network representation of a system. It is top-down partitioned, logical, concise and graphic.[8] It represents the movement of data through processes, where it is transformed by reformatting, validating, or changing in some way. Each process is a "value added" step toward producing meaningful information from isolated data. The components of a data flow diagram are: processes, data flows, data stores, and source/sinks. Figure 5 depicts a high level data flow diagram. The Cashier and Billing are sources, Accounting is a sink, and Determine Customer Variance, Capture Transactions, and Update Balances are processes. The data flows are Charge Slips, Return Slips, Payment Slips, Payments, Charges, Returns, and Customer Variance Report. The Cashier is the origin of charge and return slips. Payment slips come from the Billing department. Charge, Return,

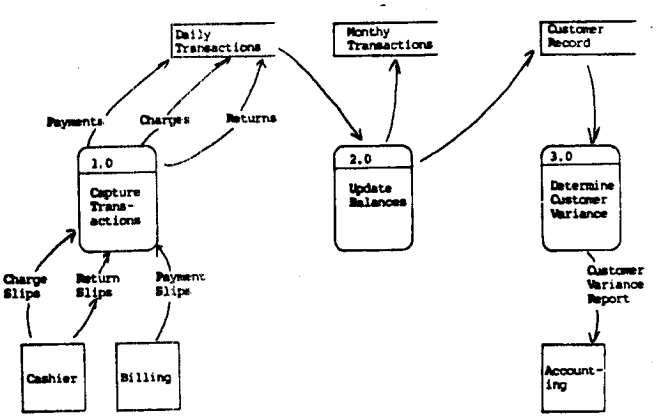


Figure 5.
A High Level Data Flow Diagram

and Payment slips are inputs to the Capture Transactions process. Daily Transactions, Monthly Transactions and Customer Records are data stores. The Daily Transaction is a time repository of data,[8] which is input to the Update Balances Process. Both Update Balances and Determine Customer Variance access the Customer Record data store. The Customer Variance Report is an output of the Determine Customer Variance process and is sent to Accounting, a sink. The data dictionary which accompanies the data flow diagram contains the description of the data flows and a description of the processes in structured English or pseudocode. Implementation details are omitted from the sample structured English for the Determine Customer Variance Process in Figure 6. The

```
3.0 For each Customer Record
        Get Customer Balances
        Add to Customer Total
        If Customer Total exceeds Credit Limit
            Add one to Over Limit Count
            Compute Amount Over Limit
            Add Amount Over Limit to Over Limit Total
            Write Variance Record

    Print Customer Variance Report
```

Figure 6.
The Structured English for the Determine Customer Variance Process

systems analyst leaves those details to the program designer in the logic design phase. DFDs are generally

introduced in the third or fourth course in the curriculum, after the student has had substantial programming experience. Creating the DFD is far more difficult than interpreting it and performing transform or transaction analysis to produce a hierarchy chart. But students can understand the DFD prior to developing the analytical skills needed to create a DFD. The Information Systems curriculum is "bottom up" rather than "top down". Logic design is taught before analysis unless the Program Development Life Cycle is emphasized in a first course and an analysis tool such as a hierarchy chart is introduced early in the curriculum.

Thus, the analysis phase of the Program Development Life Cycle is likely to have a data flow diagram as an input because the data flow diagram shows the essential features of the specification without regard to implementation. Algorithmic logic is left to the structured English and data dictionary; even there it is high level logic describing the tasks to be done at a level which will enable a hierarchy chart and more detailed Input-Process-Output (IPO) specifications to be developed. Although the DFD is input to the analysis phase of the Program Development Life Cycle, the DFD's associated structured English and data dictionary house important facts about the data flows which are needed in the logic design phase of the program development life cycle.

HIERARCHY PLUS INPUT/PROCESS/OUTPUT TECHNIQUE (HIPO)

The sequence DFD to hierarchy parallels the sequence structured English to Input-Process-Output (IPO) specifications. The designer maintains the graphic, high level from DFD to hierarchy chart while refining the algorithmic logic from structured English to IPO. The HIPO technique assists the designer in developing detailed, accurate logic while maintaining the concise, graphic hierarchy.

The hierarchy chart in Figure 4 is the high level logical portion of HIPO. The IPO chart of Figure 7 connects the hierarchy chart to the design phase via the pseudocode for the process. Also,

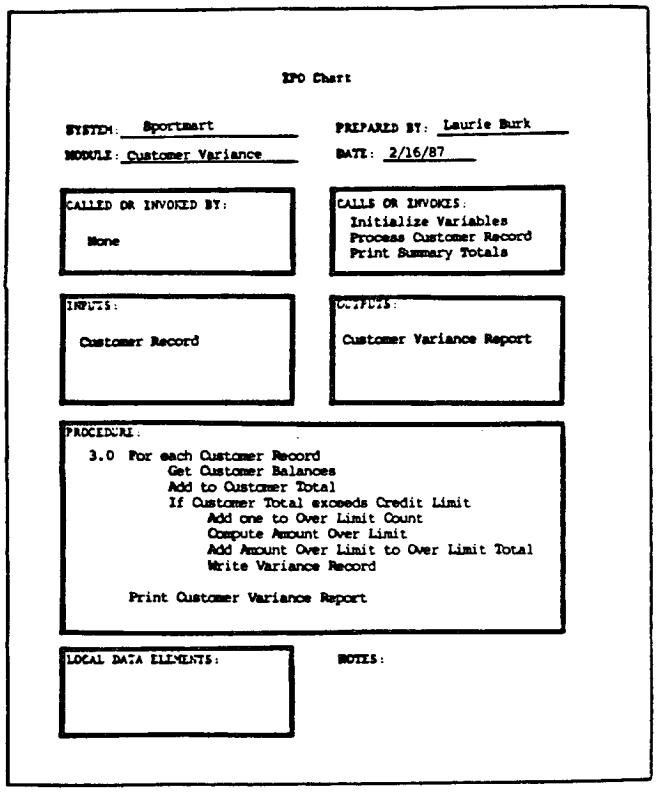


Figure 7.
IPO Chart for
Determine Customer Variance Module

the IPO chart textually describes the hierarchy by noting: which modules are invoked by the Determine Customer Variance module, which modules invoke it, what data are required by the module, and what information is exported by the module. Each module in the hierarchy chart will have a corresponding IPO chart. In Figure 7, the modules called by Determine Customer Variance are: Initialize Variables, Print Headings, Process Customer Record, and Print Summary Totals. Although Open Files and Close Files are on the hierarchy chart, they are not called by Determine Customer

Variance. These two tasks are done in the Determine Customer Variance module itself. The final hierarchy chart will differ from that in Figure 4. The Open Files and Close Files modules would be erased to reflect a one-to-one correspondence to the physical modules. The Process Customer Record module also looses some siblings in the logic design phase, as shown in Figure 4. Totals are computed in Process Customer Record rather than in separate modules. The hierarchy charts in Figure 5 went through several iterations of decomposition, each adding detail based on the specifications and requirements. The end product was Figure 4. As the logic of the modules is derived by the designer, some logical modules which resulted during the analysis phase of the Program Development Life Cycle may be absorbed back into parent modules. This is a far better situation than having too many tasks to do in one module. Complete decomposition provides a complete hierarchical list, thus ensuring that restricted control structures and transferability will be maintained in the logic design phase of the Program Development Life Cycle.

The HIPO technique could be introduced as early as the second course in the curriculum because it is both a high level graphic tool and a logic design tool. As soon as students analyze program specifications, and become proficient with a logic design tool such as pseudocode, the HIPO technique is within their capabilities. HIPO is an especially good tool for relating, transitioning and limiting high level graphic analysis tools with logic design.

THE SYSTEM FLOWCHART

The system flowchart indicates the form of the inputs, outputs and processes which were modeled in the data flow diagrams. The system components, such as files, programs, printed output, manual tasks, displays and manual input are specified in the system flowchart. Thus, the system flowchart is the physical representation, the "how to do", for the data flow diagram, which is the "what to do".

Figure 8 is the system flowchart for the Customer Variance program.

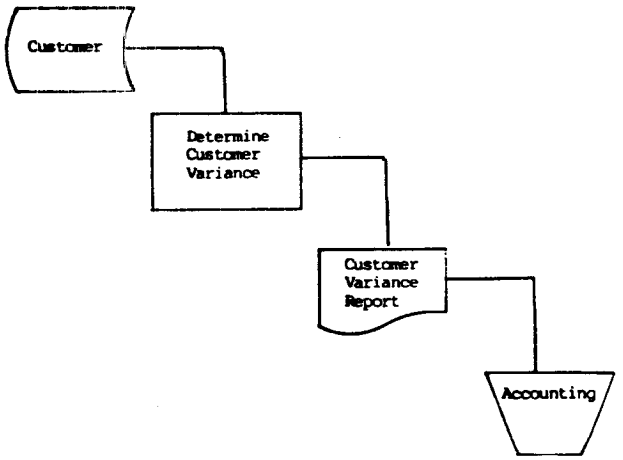


Figure 8.
Customer Variance System Flowchart

Customer File is on disk, and is input to the Customer Variance program, which produces the printed output, Customer Variance Report. The report is sent to the Credit Manager in Accounting.

The system flowchart, like the DFD, is an input to the analysis phase of the Program Development Life Cycle. The students need to understand it rather than create it in the first course. The students create the system flowchart from the DFD in the third or fourth course.

The analysis phase of the Program Development Life Cycle is supported by graphic tools which partition the program design specifications into workable sub-tasks. Hierarchy charts, and the HIPO technique together form a bridge from the analysis phase to the logic design phase. The DFD itself is mainly an input to the analysis phase, while its accompanying data dictionary is parallel to the IPO charts, and is closer to the logic design phase. The

system flowchart translates the logical flow in the DFD to the physical level needed for logic design.

LOGIC DESIGN PHASE TOOLS

Good programmers recognize the need for planning the logic of large programs before generating the source code. This process requires a planning tool to illustrate the logical steps needed to complete the algorithm. This section will provide a working definition of structured programming, illustrate how four logic design tools could be used to plan a structured program and compare these tools giving special emphasis to their appropriateness in a programming curriculum.

STRUCTURED PROGRAMMING

Many authors use the term "structured" in the titles of textbooks and then persist in providing examples that violate the premises of a structured program throughout their texts.

Since Bohm and Jacopimi's paper written in 1964 and Edsger Dijkstra's presentations and letters in 1965 and 1968, few programmers would argue against structured programming. What, exactly, is meant by structured programming? To understand the definition, one must be conversant with the three restricted control structures:

(1) Sequential control
(2) Conditional or Selective control
(3) Repetitive or Iterative control

Sequential control is the basic structure. Instructions are processed one following another in the order listed. Conditional control allows a binary choice based on the evaluation of a condition, one set of instructions are processed or a different set is processed. Traditional If-Then-Else processing is the most common form of this control structure. Repetitive control causes a set of instructions to be repeated for a predetermined number of times or, more frequently, as long as a specified condition exists. A working definition for a structured program follows.

A structured program consists only of one or more restricted control structures (sequential, conditional or repetitive) each of which has a single entry and single exit.

Good programmers also use elements of style such as proper modularization, meaningful variable names and statement indentation. The goal of structure and style is to enhance understanding of an algorithm and hence improve programmer productivity and lower costs.

DESIGN TOOLS

To illustrate the four most common design tools, a common algorithm will be used. The specifications for this algorithm follow:

Sportsmart, Inc. maintains an automated accounts receivable information system supporting a customer account file. Figure 9 provides a data element dictionary illustrating the field structure of each record in the file.

Field Name	Type	Size	Remarks
Customer Account #	A/N	6	Primary Key
Customer Name	A/N	30	
Address	A/N	30	
City	A/N	15	
State Code	A/N	2	
Zip Code	A/N	9	
Area Code & Telephone	A/N	10	
Balance Last Statement	N	4.2	####.##
Credit Limit	N	4.2	####.##
Balance Over 90 Days	N	4.2	####.##
Balance Over 60 Days	N	4.2	####.##
Balance Over 30 Days	N	4.2	####.##
Current Balance	N	4.2	####.##

Figure 9.
Data Element Dictionary : Customer
Account File Organization : Sequential

SPORTSMART, INC.
ACCOUNTS RECEIVABLE INFORMATION SYSTEM
SPECIFICATIONS: CUSTOMER VARIANCE

The Customer Variance Program provides the Accounting group of Customer Services with an exception report flagging all customers owing Sportsmart more than their credit limit allows.

Input. Inputs for the Customer Variance Program are taken from the Customer file. The report date is printed using the system date function.

Processing. Access and parse all records of the Customer Account file. For each record accessed, sum the balance past 90 days due, balance past 60 days due, balance past 30 days due and the current balance. This sum is the total amount owed by the customer. Compare this total with the credit limit. If the total due exceeds the credit limit, the customer record is to be printed to the report. With each record accessed, increment the total number of customers. With each record written to the report, increment the total number of customers over their credit limit, and add the balances and variances to their respective running totals.

Print the report title and column headers on each page of the report. Print no more than 26 customer record detail lines on a page. Double space the detail lines as shown on the report specification. On the last page, when all detail lines have been printed, print the total number of customers whose balance exceeded their credit limit and the column totals. Calculate and print the percentage of customers over their limit.

Use the following formulae to generate the report:

Total Due = Balance Past 90 +
Balance Past 60 +
Balance Past 30 +
Current Balance

Variance = Total Due - Credit Limit
(Note that this will be a positive value)

Percent Customers Over Limit =
$$\frac{\text{Number of Customers Over Limit}}{\text{Total Number Of Customers}} * 100$$

Output. The output for the customer variance program consists of the Customer Variance Report. The report is generated monthly and is sent to customer services. A printer spacing chart (not included in this paper) illustrates the detailed output specifications.

FLOWCHARTS

The traditional flowchart is still heavily used in industry. A recent study by Mary Sumner and Jerry Sitek from Southern Illinois University concluded that the flowchart was used in the logic design phase by nearly half of the projects surveyed.[9] Figure 10 illustrates the Customer Variance Report generation using this tool.

Note the care that must be taken to prevent violations of the single entry, single exit requirement of structured programming. It is very easy to slide into the unstructured trap illustrated in Figure 11.

Here the process customer record module would NOT contain the input at its end and double entry into the repetitive control structure results.

Many users of the flowchart also do little modularization, resulting in several on and off page connectors. This adds greatly to the confusion and possible misunderstanding of the algorithm, thus adding time and complexity to modification and reducing productivity.

Beginning students seem to require a geometric tool (symbols in addition to text) to sense the flow of logic during

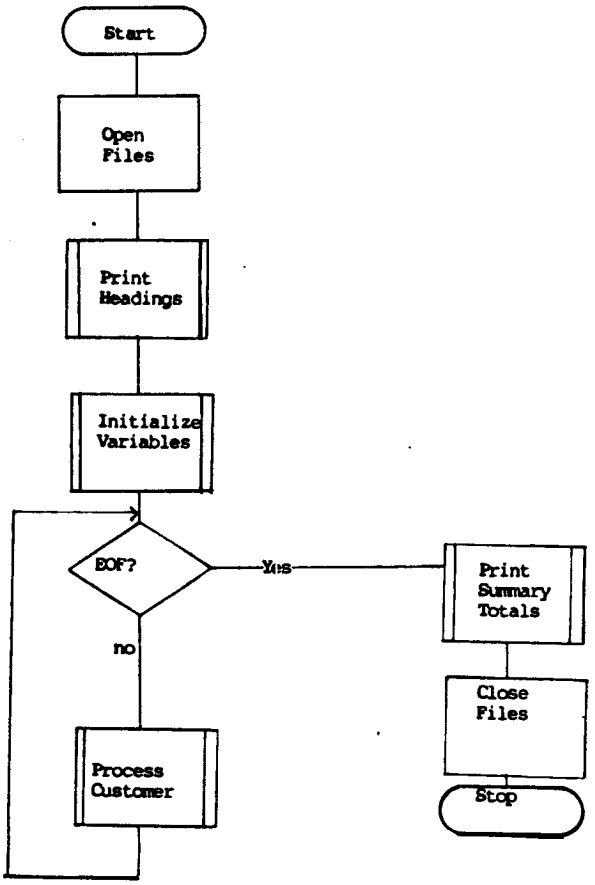


Figure 10.
A Single Entry, Single Exit Flowchart for the Customer Variance Main Program

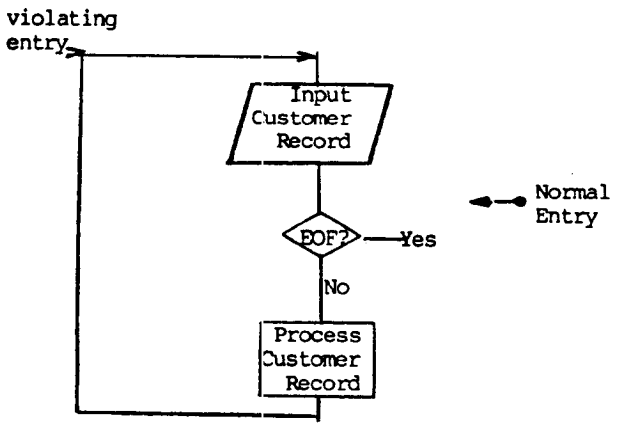


Figure 11.
Unstructured Flowchart Example

algorithm development. With care to ensure proper structure and prudent modularization, the flowchart is still a viable design tool.

D-CHARTS

To help prevent the violations of structure that are easily allowed by the traditional flowchart, a modified flowcharting technique is presented. As a tribute to Edsger Dijkstra, the technique is called the D-Chart. Figures 12-14 again illustrate the variance program design. The symbols

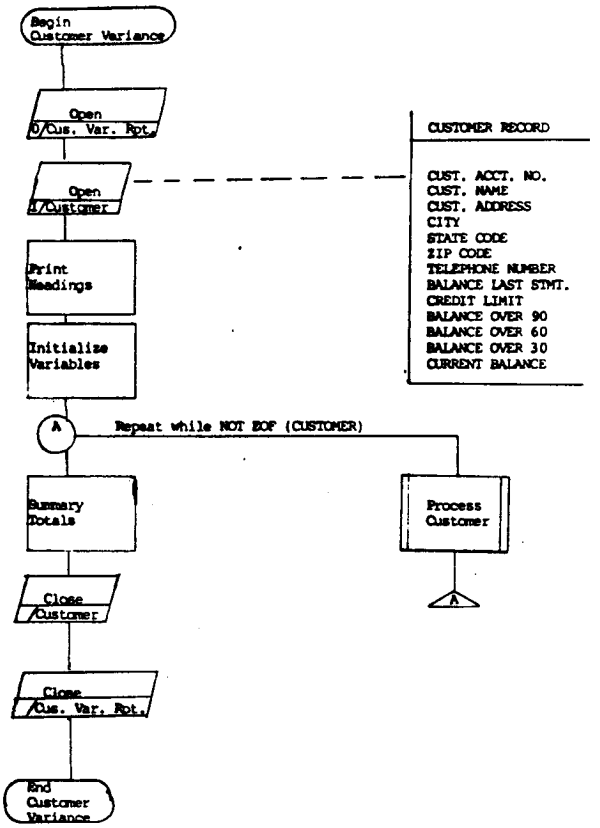
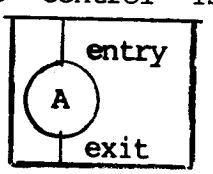


Figure 12.
Customer Variance Main Program D-Chart

are very much the same, but the connecting style to illustrate conditional and repetitive control structures is quite different. Note how the nesting of the two conditional control structures is more obvious using this tool. Since the entry and exit of the repetitive control is indicated by a common symbol, it is more difficult to violate structure requirements.



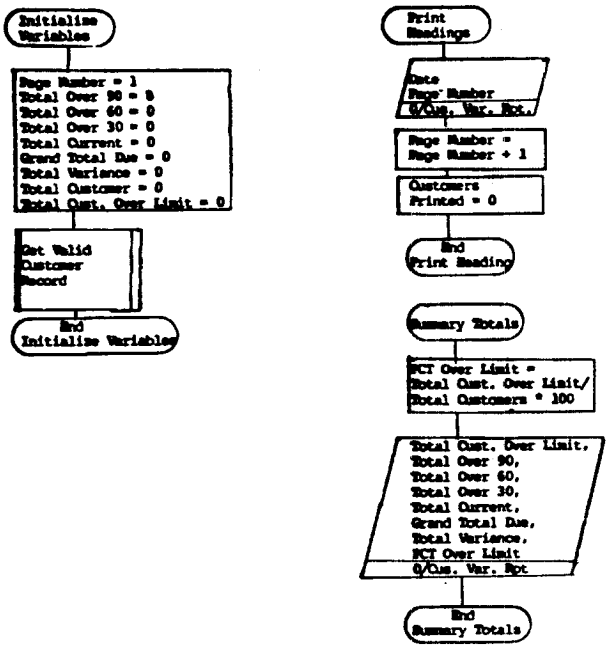


Figure 13.
D-Chart of Initialize Variables, Print Headings, and Print Summary Totals Modules

The tool is geometric and well suited for the beginning student.

PSEUDOCODE

Of the projects surveyed by Sumner and Sitek,[9] 38% used pseudocode as the logic design tool. Pseudocode is nongeometric, using no symbols to indicate the actions being designed. The tool is easier to modify, is better suited to word processing and requires less space. Students preferring pseudocode, however, seem to require maturity in their logic development skills, thus making this tool better suited to the more advanced programming courses. Figure 15 illustrates the Customer Variance algorithm design using pseudocode.

AND THE SURVEY SAYS?...

A survey of 97 business data processing students was conducted at the end of the first and second programming courses at Purdue University to determine a preference between the D-Chart and pseudocode as a tool for logic design. Of the 97 students in the first course, 33 had little or no previous experience with logic design. In this group, the D-Chart was preferred by a margin of over 8 to 1. The remaining 64 students with significant experience were more mixed with 39 opting for D-Charts. At the end of the second course, 83 of the original 97 students were available with whom to reconduct the survey. A dramatic shift of preference occurred with pseudocode now being a nearly 7 to 1 favorite. Figure 16 illustrates the survey results.

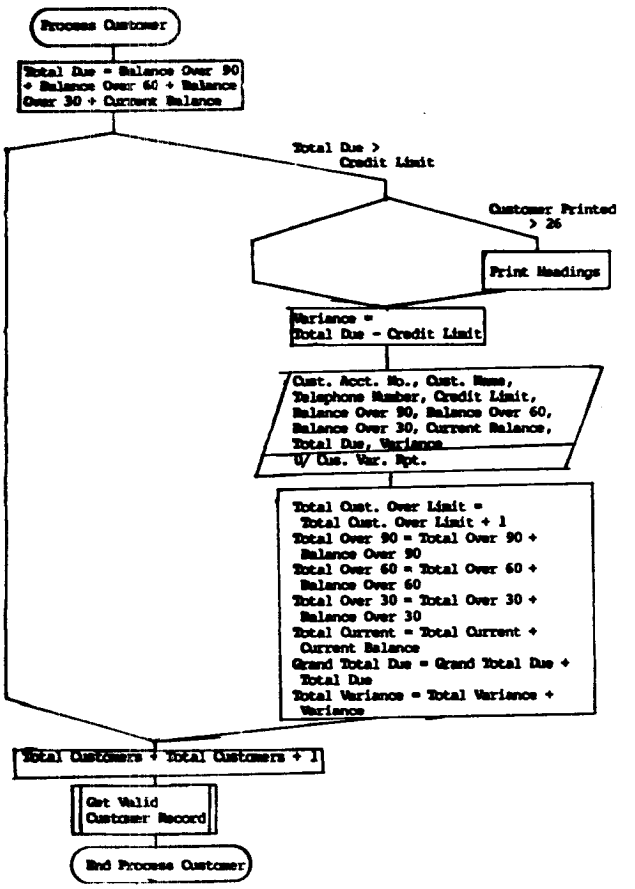


Figure 14.
D-Chart for the Process Customer Module

```
Begin CUSTOMER VARIANCE module
        Define CUSTOMER RECORD as
                        CUST ACCT NO
                        CUST NAME
                        CUST ADDRESS
                        CITY
                        STATE CODE
                        ZIPE CODE
                        TELEPHONE NUMBER
                        BALANCE LAST STMT
                        CREDIT LIMIT
                        BALANCE OVER 90
                        BALANCE OVER 60
                        BALANCE OVER 30
                        CURRENT BALANCE

    Open (input) CUSTOMER file
    Open (output) CUSTOMER VARIANCE REPORT file

    Process PRINT HEADINGS module
    Process INITIALIZE VARIABLES module

    Repeat while NOT END OF CUSTOMER FILE
        Process CUSTOMER RECORD module
    End report

    Process SUMMARY TOTALS module

    Close CUSTOMER file
    Close CUSTOMER VARIANCE REPORT file

End CUSTOMER VARIANCE module
```

Figure 15.
Pseudocode for Customer Variance Main Program

SEMESTER	LITTLE EXPERIENCE	SIGNIFICANT EXPERIENCE
1st	n=33 D-Charts: 29 Pseudocode: 4	n=64 D-Charts: 39 Pseudocode: 25
2nd		n=83 D-Charts: 11 Pseudocode: 72

Figure 16.
D-Chart/Pseudocode Preference Survey

NASSI-SCHNEIDERMAN DIAGRAM

The Nassi-Schneiderman diagram is a hybrid between pseudocode and the geometric tools. The logic flow is depicted graphically but without flow lines, arrows or separate symbols for design tasks. The diagram enhances the development of structured programs because the graphics provide only the means to illustrate the three restricted control structures. The disadvantages of the geometric tools (for example, modification difficulty, ill-suited to word processors) still are present however. In practice, the Nassi-Schneiderman will work well with more mature programmers but tends to break down when the algorithm is very large and complex. Figure 17 shows the variance report program designed using the Nassi-Schneiderman technique.

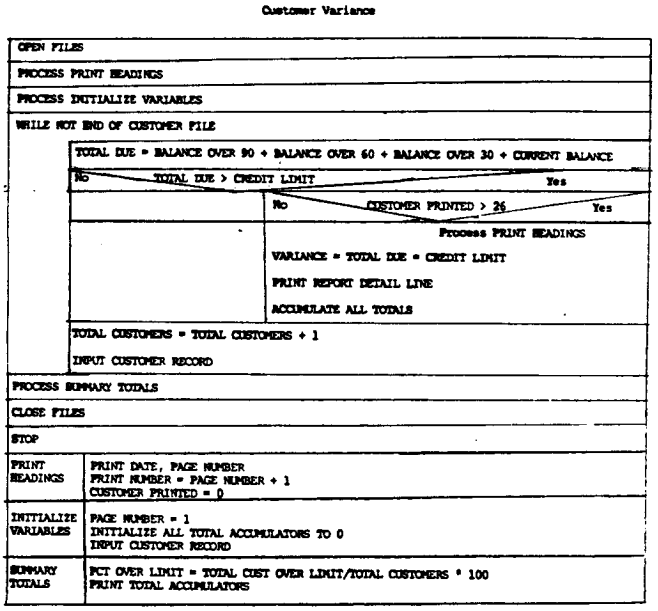


Figure 17.
A Nassi-Schneiderman Diagram for the Customer Variance Main Program

CONCLUSION

The tools supporting the Program Development Life Cycle should be introduced in the the Information Systems curriculum as early as possible. Even though the first assignments may be short and simple, both analysis and design tools should be required prior to the actual coding of the program to prepare students for the large and complex assignments to come. Analysis tools are needed from the

earliest exposure to programming to promote consistent structured design. Logic design tools are needed to promote structured programming. It appears that in the initial course, a geometric design tool such as the D-Chart is the most appropriate. Non-geometric tools such as pseudocode are better suited for the experienced student.

BIBLIOGRAPHY

1. Bohm, C. and Jacopini, G. "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Communications of the ACM, 9, 5 (May 1966), 366-371.

2. Davis, William S. (1983). Tools and Techniques For Structured Systems Analysis and Design. Reading, Massachusetts: Addison-Wesley.

3. Dolan, Kathleen (1984). Business Computer Systems Design. Santa Cruz, California: Mitchell Publishing.

4. Dijkstra, Edsger. "Go To Statement Considered Harmful." Communications of the ACM, 11, 3 (March 1968), 147-148.

5. Koffman, Elliot B. (1986). Problem Solving and Structured Programming in Pascal, 2nd edition. Reading, Massachusetts: Addison-Wesley.

6. McMenamin, Stephen and Palmer, John. (1984). Essential Systems Analysis. New York, New York. Yourdon Press. 1984.

7. Nickerson, Robert C. (1987). Fundamentals of Structured Cobol, 2nd edition. Boston, Mass. Little, Brown and Company.

8. Page-Jones, Meilir (1980). The Practical Guide to Structured Systems Design. New York: Yourdan Press.

9. Sumner, Mary and Sitek, Jerry. "Structured Methods and their Practical Use. Implications for Educations" in Proceedings of the Fifth Annual Information Systems Education Conference. Atlanta, Georgia. October 25-26, 1986.

10. Weaver, Lloyd R. "The Program Development Life Cycle" in Proceedings of the Fifth Annual Information Systems Education Conference. Atlanta, Ga. October 25-26, 1986.

THE SHOEMAKER'S CHILDREN -- ARE SYSTEMS ANALYSIS AND DESIGN COURSES TEACHING THE RIGHT TOOLS?

Jennifer L. Wagner, Ph.D.
Roosevelt University

Harry C. Benham, Ph.D.
University of Oklahoma

The fable tells of a busy shoemaker who neglected to make shoes for his own children. This paper examines the "shoemaker" aspects of systems development. Are there appropriate, automated, up-to-date, and usable software development tools? The content of systems development texts was examined to determine the depth of coverage of computer aided software engineering (CASE) tools and related strategies. The results indicate that, while CASE tools are available, most developers are not using them, nor do the texts available today fully discuss them.

INTRODUCTION

A well-known fable tells of a shoemaker who was so busy providing shoes for his customers that he neglected to make shoes for his own children, who had to go barefoot. There are modern business parallels; poorly managed management consulting firms and poorly staffed personnel agencies are commonly observed examples. This paper examines the "shoemaker" aspects of systems analysis and design. Has the software development profession, busy providing automated tools for others, failed to supply itself with the most appropriate, automated, up-to-date, and usable tools?

APPROPRIATE TOOLS FOR ANALYSIS AND DESIGN

Ed Yourdon, one of the early proponents of structured techniques, has recently pointed out that a new set of tools should be available to systems developers (9). He refers to them as computer aided software engineering (CASE) tools and suggests that they should include:

(1) the entity-relationship diagram as a replacement for the data structure diagram, in order to show relationships among higher order data groupings;

(2) state transition diagrams, which can model time-dependent data and represent conditions which cause the system to change from one state to another and which are particularly important in real-time systems;

(3) a new data flow diagram, which indicates signals, interrupts, and control processes, which coordinate the timing of other functions; and

(4) automatic cross-checking rules to guarantee consistency among the entity-relationship diagram, state transition diagrams, and data flow diagram.

Yourdon has further suggested the incorporation of two overall strategies into the analysis and design processes:

(5) a reduced emphasis on the "current system" model because it takes too long and produces too little new, interest-

ing, and usable information; and

(6) the development of an automated systems analysis database and automated graphics tools which will eliminate the tedium of drawing and redrawing data flow diagrams.

These CASE tools and strategies will serve as the basis for the present discussion, not because Yourdon's opinion is necessarily an accurate representation of system developers' needs, but because his view of the profession is the result of a long-standing involvement and concern.

Availability. According to Yourdon, whose firm provides workstation products, there are 6000 CASE workstations now in place in this country. This translates to workstations for about 1% of the programmer/analysts. Presumably, these workstations run software packages like Excelerator, a microcumputer-based systems development tool from Index Technologies Corporation. Excelerator is a completely integrated development workbench, which permits a variety of graphic system representations and includes an automated development database. This package and other automated development tools, such as Metafile and the MicroFocus Cobol Workbench, are readily available commercially.

Thus, it seems that appropriate, automated, and up-to-date tools are, in fact, available to systems developers. However, systems professionals must know of such tools in order to use them. This knowledge is probably best acquired within the university information systems curriculum.

Texts. Ideally, university information systems courses should provide students who plan to become systems developers with "hands-on" labs using appropriate and up-to-date CASE tools, several of which are available in "classroom", "student", or abridged versions. The study of text material, which corresponds to, explains, and elaborates on such "hands-on" experience, provides a complete and useful educational experience for the student. This type of experience is certainly available in some programs. However, informal surveys have shown that it is the exception, rather than the rule.

If "hands-on" experience is not available, the students should at least learn about automated development tools during their coursework. The textbook for the systems analysis, design, or development course should, thus, include coverage of such tools. The content of such texts should serve as an indicator of the expected familiarity of prospective systems developers with appropriate techniques.

METHOD

Seven systems analysis, design, and development texts were examined (1), (2), (3), (6), (7), (8), (10). The three texts which are over six years old are still in print, are used widely, and are considered classics. The development of the Powers et al. text was based on the DPMA curriculum guidelines. Similarly, the Couger et al. text is based on the ACM curriculum guidelines.

The content of each of the seven texts was analyzed to determine its depth of coverage of each of the CASE tools and strategies suggested by Yourdon.

RESULTS

The treatment of each tool suggested by Yourdon is discussed separately.

The Entity-Relationship Diagram. Only the Eliason text mentions the entity-relationship diagram; it is discussed thoroughly, as is the standard data structure diagram. The entity-relationship diagram is described as more "user-oriented", while the data structure diagram is represented as very physical and possibly

implementation-dependent.

Couger et al. mention the term "entity diagram" (p. 78), which is treated as another name for a data flow diagram. Page-Jones advocates the use of a data access diagram in place of a data structure diagram; this data access diagram looks quite similar to today's entity-relationship diagram. Powers et al., on the other hand, describe a data access diagram which is very physical, but their data structure diagram looks much like an entity-relationship diagram.

The Senn, DeMarco, and Yourdon and Constantine texts do not discuss entity-relationship diagrams in any form or under any pseudonym; their data structure diagrams are traditional.

State Transition Diagrams. Page-Jones points out the frequent demand for state memory, but advises avoiding it whenever possible; he has no device, graphic or otherwise, for dealing with it easily. The Yourdon and Constantine text delivers a similar treatment, with a somewhat greater emphasis on the need for state transitions, but again without an acceptable method for annotating them.

The texts by Couger et al., DeMarco, Eliason, Powers et al., and Senn do not mention state transitions or state memory in any identifiable form.

A New Data Flow Diagram. While all of the texts discuss the inclusion of control, none of the authors fully advocate it. The Couger et al. text states that data flow diagrams provide only weak support for control. Eliason and Page-Jones show control only with occasional flags in structure charts. Senn permits control indicators on lower level data flow diagrams only. The Yourdon and Constantine text is very design-oriented and includes only a minimum of analysis; it does, however, provide a fairly theoretical discussion of synchronizing modules. The DeMarco and Powers et al. texts both forbid the showing of controls in data flow diagrams.

Automatic Cross-Checking Rules. None of the texts specifically discuss automated cross-checking rules. The Couger et al. text mentions them as a part of general automated tools.

Less Emphasis On The Current System Model. Both the DeMarco and Powers et al. texts emphasize (with about twenty text pages) modeling the current system. The Eliason and Senn texts place less emphasis, indicating that data flow diagrams of the required (new) system should be prepared. The Couger et al., Page-Jones, and Yourdon and Constantine texts, which are design-oriented, do not treat this portion of systems analysis.

An Automated Database And Graphics. The bulk (roughly 400 pages) of the Couger et al. text discusses the automation of systems development. Chapter titles (and authors) include: "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems" (Daniel Teichrow and Ernest A. Hershey, III); "The PSL/PSA Approach to Computer Aided Analysis and Documentation" (Daniel Teichrow, Ernest A. Hershey, III and Y. Yamamoto); "An Extendable Approach to Computer-Aided Software Requirements Engineering" (Thomas E. Bell, David C. Bixler, Margaret E. Dyer); "Plexsys: A System Development System" (Benn R. Konsynski and Jay F. Nunamaker); "Automation of System Building" (Daniel Teichrow and Hasan Sayani). A variety of systems are described, providing the student with a thorough understanding of the need for and availability of automated development tools.

DeMarco lists the features desired in an automated systems analysis package and provides a critique of the automated data dictionary packages

which were available in the late 70's. His text emphasizes the need for drawing and redrawing throughout. Page-Jones also emphasizes graphics, but fails to mention automation. The Eliason, Powers et al., Senn, and Yourdon and Constantine texts also fail to mention any automated tools.

CONCLUSION

While CASE tools are available to systems developers, most are not using them, nor do the texts available today fully discuss them. Only the Couger et al. text emphasizes the automation of the systems development process or the provision of cross-checking rules for maintaining consistency among the elements of the automated process.

Of the other tools recommended by Yourdon, the entity-relationship diagram seems to be "catching on"; that is, the specific term is defined fully in the most recent text, while the general concept is treated briefly in the fairly recent texts, and completely ignored in the oldest texts. It also seems that the more recent texts are de-emphasizing the "current system" model.

State transition diagrams, or other similar methods for dealing with the system's change from one state to another, are treated rather unsuccessfully and only in the oldest texts; this may be due to the perceived infrequency of real-time business systems. The data flow diagram showing control processes still seems to be against most authors' principles.

If systems developers are to be able to utilize the "tools of their trade" fully and not be seen as "the shoemaker's children", automated development systems must play a greater role in the curriculum and textbooks.

REFERENCES

(1) Couger, J. Daniel, Mel A. Colter, and Robert W. Knapp, Advanced System Development/Feasibility Techniques, John Wiley, 1982.

(2) DeMarco, Tom, Structured Analysis and System Specification, Prentice-Hall, 1979.

(3) Eliason, Alan L., Systems Development, Little, Brown, 1987.

(4) Goldberg, Eddy, "Excelerator aids Stradis adoption," ComputerWorld, April 21, 1986, p. 43.

(5) A Guided Tour of Excelerator, Index Technologies Corporation, 1985.

(6) Page-Jones, Meilir, The Practical Guide to Structured Systems Design, Yourdon Press (Prentice-Hall), 1980.

(7) Powers, Michael J., David R. Adams, and Harlan D. Mills, Computer Information Systems Development: Analysis and Design, South-Western Publishing Company, 1984.

(8) Senn, James A., Analysis and Design of Information Systems, McGraw-Hill, 1984.

(9) Yourdon, Edward, "What Ever Happened to Structured Analysis?" Datamation, June 1, 1986, pp. 133-138.

(10) Yourdon, Edward, and Larry L. Constantine, Structured Design, Yourdon Press (Prentice-Hall), 1978.

LEARNING EFFECTS AND RESOURCE UTILIZATION WITH PEER DEBUGGING IN AN INTRODUCTORY PROGRAMMING COURSE: REPLICATION THROUGH A CONTROLLED EXPERIMENT.

By

George Jacobson and George Burstein
California State University, Los Angeles
5151 State University Drive
Los Angeles, CA 90032
(213)224-2961

ABSTRACT

Peer debugging is associated with higher levels of learning for teaching BASIC and with marked reduction of resource use when teaching Lotus 1-2-3. These findings persist over a variety of instructional modes, durations, and applications. Statistical significance was again established at the .05 level in a controlled experiment with students in a university introductory programming course (Burstein & Jacobson 1986). Superior levels of learning were found when students in control groups (working without peer debugging) were compared with students in experimental groups (using peer debugging). The different languages used in these studies (COBOL and BASIC) did not affect this result. The results were consistent whether the learning was of a ten-week or a six-week duration. Whether researchers used batch processing with a mainframe computer, used timesharing with a minicomputer, or with a microcomputer did not affect these findings. Students involved with peer debugging learned at the same higher level whether they programmed on microcomputers, minicomputers or mainframes. Cost savings approximating 50%, without reduction in learning levels, accompanied the assignment of two students to one microcomputer in the four week portion of the same introductory course of instruction which dealt with Lotus 1-2-3. Implications of this replication of findings for university faculty and training staff in business and industry are discussed.

INTRODUCTION

Burstein and Jacobson (1986) conducted a controlled experiment to study the impact of elements of experiential learning on the development of superior courses of instruction in the university Information Systems curriculum. This experiment was concluded in December 1986. Findings concerning the design of such experiments and the effects of peer debugging on learning and resource use are reported here. Replication of earlier studies in

the field of introductory programming instruction are evident. This replication has striking implications for both university faculty and administrators, for human resource managers, training directors, for organization behaviorists, and for administrators in those business and industrial firms. Such specialists who develop and conduct introductory programming training for employees may well be able to transfer these findings to their programming instruction.

The paper reports the findings concerning learning differences (as measured on the final examinations in BASIC and Lotus 1-2-3) in some of the control and experimental sections of an introductory programming course at a university. This course is a combination of CIS/86-1 AND CIS/86-2 in the model curriculum of the DPMA. This course is in addition an important component of the common business core as defined by the American Assembly of Collegiate Schools of Business (AACSB), the major accrediting body for Schools of Business. It exposes the student in School of Business specialization to general computer concepts and program languages. In addition, it serves as the first computer course for students majoring in Information Systems.

The field of education for information systems is exploding with new technologies, multidisciplinary applications, and a wide variety of students from different backgrounds. Weinberg (1971) and Dossett & Hulvershorn (1983) suggest that experiential learning, and especially peer debugging may spell the difference between ineffective and effective teaching/learning.

METHODOLOGY

The experiment was conducted as controlled. Literature review, adaptation and development of instruments, selection of instructors, and selection of students all proceeded in a manner which created equivalent environments for both control and experimental groups for all variables except those which related to peer versus individual learning modes. During periods of the course when BASIC was taught, the students in experimental groups were randomly paired with a co-student who debugged the homework assignment (a computer program) for the paired student. This debugging occurred prior to submission of these programs for faculty evaluation. Students in control groups submitted their work for faculty evaluation without peer review. The group debugging time spent in experimental groups came at the expense of additional lecture time provided to students in the control groups. During the portion of the course in which Lotus 1-2-3 was taught, students in the experimental groups were assigned in pairs to one microcomputer. A pre-test of Lotus 1-2-3 was administered in the first session when it was to be taught.

At the very first portion of the first class meeting students completed a 30 true-false objective question pre-test of knowledge of BASIC. Jacobson & Jackson used the identical test for BASIC in 1984 (Jacobson & Jackson 1986). The syllabus was also distributed during this first class. It contained the homework assignments (exercises in computer programming which were to be based on prior lectures) due in writing, at the beginning of each subsequent session for processing during that session. Students assigned to experimental groups exchanged their homework assignments with peers, who were randomly selected and changed in each session. The peers reviewed each other's homework, wrote a specific critique to facilitate the running of the program, signed and returned the paper they reviewed. Students then attended an hour's lecture which did not cover the specific material of their current homework assignment. Each student then entered his or her program as critiqued into the computer.

Students in the control groups processed their homework differently. They too attended the same type lecture, but it

was about ten minutes longer. This was to compensate for the time spent in peer review by students in the experimental groups. Following this lecture students in the control groups entered the program they had developed at home directly on the computer. Homework assignments for all students were graded with a standard key. Students could choose not to submit their homework, but were graded accordingly. The final examination in BASIC occurred at the end of the sixth week of the course while that for Lotus 1-2-3 was given at the end of the tenth week. These were objective standardized tests.

Only instructors with matched numbers of control and experimental sections in the course were included in the study. In this way the instructional skill, motivation, and other personal factors which might influence the results could better be kept constant. Another control used was the weekly assignment time for control and experimental sections. The instructors didn't know whether they taught the control or experimental laboratory section first. After the experiment ended one instructor said that he taught the experimental group earlier in the week and was surprised that it had a higher learning level since he was more "practiced" later in the week. As a result of these modifications, three instructors in a total of eight sections (four experimental and four control classes) participated in the study. The control sections and the experimental sections each included 73 students who took the final examination. Findings are based on these 146 students.

FINDINGS

The students in both control and experimental groups learned Lotus 1-2-3 at the same level of effectiveness, $t=1.328$, Significant at .1862. However, the savings of instructor time and resource utilizaiton was significant (approximately 50% reduction in computer use, and freedom of instructors to provide more individualized instruction). Figure One demonstrates that the experimental and control groups were in fact equally ignorant of the content of BASIC at the outset of the course. A random mean score, in this 30 true/false test in BASIC, would be 15. The score for the control groups was 17.2836 and the mean for the experimental groups was 17.4583. This equivalence was found in the pre-test administered in the first session of the course.

Figure Two details the first set of findings of this experiment (those concerned with learning differences between the control and experimental groups in the BASIC portion of the course).

Figure Three compares the Findings in Figure Two with the findings in the Jacobson & Jackson study (1986). Both final examinations were composed of 50 multiple choice questions. They were essentially identical. Several changes had to be made since the Burstein/Jacobson study was of a six-week duration, while the Jacobson/Jackson study was for a ten-week cycle.

(See Figures One, Two and Three)

DISCUSSION

There is a consistent superiority for experiential learning, as characterized by peer debugging, in laboratory sections of introductory computer programming courses. Lemos (1978, 1979, 1980) taught such courses using batch processing to teach COBOL on mainframes. He obtained results consistent with those found in this experiment. Jacobson and Jackson (1986) found the same superiority for peer debugging when teaching a ten-week course in BASIC with a minicomputer on a timesharing basis. These consistencies would have been predicted from Weinberg's (1971) and Dossett & Hulvershorn's (1983) premises. Weinberg stated that experiential learning was to be preferred when teaching compuer programming. Dossett & Hulvershorn found that air force trainees assigned in pairs to CAI training (a peer training system) took less time than the

individual group and that this gain was further increased since two trainees could be assigned to instruction at one computer. This superiority seems to occur due to additional learning from medial feedback. Such feedback occurs when peers provide a period of "safety rehearsal". This rehearsal gives students a chance to try their new responses on peers rather than with authority figures (faculty) before formal evaluation occurs. Such feedback meets the criteria of immediacy, accuracy, and opportunity for correction (Hamner 1985; Skinner 1969).

Not only does the assignment of two students/trainees to one computer save scarce resources, Dossett & Hulvershorn suggest that the money and time saved could be used by instructors to provide individual student help when required. These savings can also be used by researchers to gain knowledge about the techniques and instructional variables which support improved instruction and research.

Such suggestions and others concerning peer debugging and medial feedback processes could well improve the effectiveness of introductory computer programs at our universities and places of business. Many people without programming skills enter such university and business programs. They will learn faster and with less trauma should they be taught in a paired setting. The experience of "safety rehearsal" seems to promote less costly (economic and emotional) human development. Surely university and business administrators, faculty, training and development staff, human resource managers, and organizational behaviorists should encourage such experiences.

REFERENCES

Burstein, G. & G. Jacobson (1986). "An Empirical Approach to Studying the Effects of Medial Feedback in an Introductory Programming Course", Proceedings, National meeting of the Association for Human Resources Management and Organizational Behavior (HRMOB), New Orleans, LA, November 19, Vol. 1, pp. 92-96.

Dossett, D.L. & P. Hulvershorn (1983). "Increasing Technical Training Efficiency: Peer Training via Computer-assisted Instruction." Journal of Applied Psychology, Vol. 68, pp. 552-558.

Hamner, W.C. (1985). Reinforcement Theory. In H.L. Tosi & W.C. Hamner (Eds.). Organizational Behavior and Management (4th Ed.) (pp. 195-212). Columbus, Ohio: Grid Publishing Company.

Jacobson, G.H. & D.P. Jackson (1986). Measurement of Two Teaching Strategies for Computer Programming Instruction. Interface: The Computer Education Quarterly, Vol. 8(2), pp. 26-30.

Lemos, R.S. (1978). Student's Attitudes Toward Programming: The Effects of Structured Walk-Throughs. In Computer and Education, Vol. 2 (pp. 301-306). Great Britain: Pergamon Press.

---. (1979). An Implementation of Structured Walk-Throughs in Teaching COBOL Programming. Communications of the ACM, Vol. 22, pp. 261-273.

---. (1980). Measuring Programming Language Proficiency. AEDS Journal, Vol. 13, pp. 261-273.

Skinner, B.F. (1969). Contingencies of Reinforcement. New York: Appleton-Century-Crofts.

Weinberg, G.M. (1971). The Psychology of Computer Programming. New York: Van Nostrand Reinhold Company.

Figure One
Results Pre-Test in Basic

Lab. Instruc.	"N"		Mean		SD	
	CONT.	EXP.	CONT.	EXP.	CONT.	EXP.
A	25	22	17.1200	18.3182	3.5275	4.5186
B	19	23	17.0526	16.1739	3.4877	2.7577
B	12	14	17.6667	17.8571	2.3094	2.4133
C	11	13	17.6364	17.8462	2.3779	2.9396
Totals	67	72	17.2836	17.4583	3.1082	3.4271

t= -.3140 Difference between mean scores not significant.

Figure Two
Differences in Learning on Final Examination in Basic: Comparison of Students Using Peer Debugging and those not Using Peer Debugging.

Lab. Instruc.	"N"		Mean		SD	
	CONT.	EXP.	CONT.	EXP.	CONT.	EXP.
A	25	22	31.9600	35.4545	7.3625	6.3375
B	20	23	35.0000	37.0870	6.7043	4.8796
B	14	14	35.7143	37.3571	6.9330	4.0876
C	14	14	31.0714	32.8571	6.8440	7.2096
Total 73	73	33.3425	35.8356	7.1126	5.8405	

t= -2.3145, Difference between the mean scores is significant at the .025 level and approaches the .02 level.

Figure Three
Learning Basic: Differences Found Between Peer Debugging and Non-Peer Debugging in Two Studies.

Researchers	"N"		Mean		SD	
	CONT.	EXP.	CONT.	EXP.	CONT.	EXP.
JACOBSON/JACKSON	143	146	38.007	39.336	5.98	5.25
BURSTEIN/JACOBSON	73	73	33.3425	35.8356	7.1126	5.8405

t= -2.005 (Significant at .0459) JACOBSON/JACKSON
t= -2.3145 (Significant at .025) BURSTEIN/JACOBSON

SUPPLEMENTING BUSINESS APPLICATION PROGRAMMING INSTRUCTION WITH AN ELECTRONIC MAIL AND BULLETIN BOARD SYSTEM

Robert E. Schooley, San Diego State University
Steve Harriman, San Diego State University

An electronic mail and bulletin board system was implemented in an IDS 280 COBOL Programming course, during the spring semester of 1987 at San Diego State University by Dr. Robert E. Schooley. The system's major objective is to open, maintain, and facilitate nontraditional lines of communication between students enrolled in IDS 280 COBOL Programming and Dr. Schooley, the instructor of IDS 280 COBOL Programming. Employment of the electronic mail and bulletin board system has the potential to not only expand and increase the effectiveness of an instructor's out-of-class (office hours) assistance time, but to as well contribute to the growth of a student's "computer ease-of-use" confidence.

INTRODUCTION

An instructor, regardless of his or her subject area, is limited to the amount of time he or she can devote to assisting a student outside of formal class meetings (often referred to as office hours). This limited supply of instructor time may compound a student's frustration because students do not stop needing assistance when an instructor's office hours end.

One way of maximizing the effectiveness of out-of-class instruction and assistance in the area of business application programming would be to implement an electronic mail and bulletin board system. The computer, when employed as a communication medium, becomes an added dimension in the teacher/student relationship. Via electronic mail and bulletin boards, students are allowed to ask clarification questions when the need arises--not when their schedules are in sync with their instructor's office hours.

The strength, however, of the electronic mail and bulletin board system lies not in the area of convenience, but that students are encouraged and allowed to use the computer as a tool. This objective is consistent with the overall objective of the Data Processing Management Association's Computer Information Systems Model Curriculum for undergraduate business students. The goal of the electronic mail and bulletin board system is to assist in developing a student's ability to understand and apply the computer as a cost-effective business decision making tool.

Before students can apply the computer as a tool, they must first develop a high-degree of computer "ease-of-use" comfort. To insure a high probability of successful student usage, the electronic mail and bulletin board system was designed and implemented for a user population with very low, if any,

computer literacy. If students can successfully apply the computer to the very fundamental human need to communicate, their computer "confidence" and "ease-of-use" tends to grow, and possible feelings of computer "anxiety" begin to subside. It is at this point-in-time when students begin to view the computer as a resource as opposed to an obstacle. Such a computer reaction may prove to be one of the most critical aspects of a student's educational experience in the area of information systems.

OBJECTIVE OF THE ELECTRONIC MAIL AND BULLETIN BOARD SYSTEM

An electronic mail and bulletin board system was implemented in the spring semester of 1987 at San Diego State University. The objective of the system was to allow for a nontraditional means of opening, maintaining, and facilitating lines of communication between students enrolled in IDS 280 COBOL Programming and the instructor of IDS 280 COBOL Programming.

ELECTRONIC MAIL AND BULLETIN BOARD SYSTEM

HARDWARE AND SOFTWARE OVERVIEW

The electronic mail and bulletin board system is based on the San Diego State University Cyber 170-750 mainframe computer. The system functions under the domain of the Network Operating System. Both the Cyber 170-750 and the Network Operating System (NOS) are Control Data Corporation products. Student access to the system is available via the San Diego State University coaxial-based local area network or a dial-up wide area network which can support 300, 1200, and 2400 baud transmissions.

The system uses the NOS system package "CSU Mail (Version 4.0)" and three instructor generated files, "COBNEWS," "BBHELP," and "OPTION." The file "COBNEWS" serves as the actual bulletin board file for the system, with the file "BBHELP" serving as an on-line, student help file for using the electronic mail and bulletin board system. The third and final file "OPTION," works in conjunction with the CSU Mail package.

Within the CSU Mail package, an instructor or anyone validated to use the package, may create an "OPTION" file. An "OPTION" file allows the sender of mail to route a message to a group of receivers (up to 100 addresses), by using a common name to describe the group. The "OPTION" file used at for the IDS 280 class at San Diego State University allows an instructor of IDS 280 to send a mail message to all of the IDS 280 students by addressing the one message to: 280CLASS.

IMPLEMENTATION OF THE ELECTRONIC MAIL AND BULLETIN BOARD SYSTEM

Before the beginning of the semester the file "COBNEWS" was prepared with an electronic mail and bulletin board system welcome message. The file also includes instructions on how to use the electronic bulletin board and where the student can get assistance with respect to using the bulletin board. A mail message, prepared and mailed before the start of the semester, welcomes the student to the IDS 280 COBOL Programming class and supplies the student with a short description of how to use CSU Mail and how to receive help concerning the use of the CSU mail package.

During the first class meeting of IDS 280 COBOL Programming, students receive a Cyber computer account and an orientation package which details the use of the Cyber 170-750 mainframe computer, the role of the Network Operating System, and a description of and instructions for the electronic mail and bulletin board system. The remainder of this first meeting of IDS 280 and the entire second meeting of the class are devoted to explaining such things as logging on and off the CYBER,

NOS file types and classifications, using NOS text editors, the role of the NOS compiler "COBOL5", and how to use the CSU Mail package and the electronic bulletin board file "COBNEWS".

Students are encouraged during these first two class meetings to send messages via CSU mail to their instructor and/or attach an announcement to the class bulletin board file "COBNEWS". Great care is taken in these first two meetings to assure students not to worry about making mistakes, and to view these initial electronic mail and bulletin board attempts as experiments. Such an approach tends to eliminate student anxiety with respect to using the computer and allows the student to slowly develop computer ease-of-use comfort. Failure to develop such a comfort with using the computer may prove to be an insurmountable obstacle in a student's attempt of mastering a business application programming language.

As the semester progresses the IDS 280 students are instructed to check their mail and read the bulletin board before coming to class and every time they log on to the CYBER system. Students are encouraged to not only voice their questions and/or failures, but to share their discoveries and successes as well.

Throughout the semester, the IDS 280 instructor may spot trends in student comments, concerns, and questions by scanning mail messages and the class bulletin board. Once a trend in student messages begins to emerge, an instructor may issue a blanket mail and/or bulletin board message describing the area of concern and possible solution suggestions. Such an approach not only allows the instructor to address students who encountered the situation at hand but it also allows the instructor to warn the remainder of the class of a potential concern.

CONCLUSION

The amount of any instructor's out-of-class time and assistance is both fixed and limited. The out-of-class needs of students enrolled in a business application programming class are very atypical and often extremely high. In order to maximize the effectiveness of the business application programming instructor's out-of-class time, an electronic mail and bulletin board system should be implemented.

When students and instructor are unable to meet face-to-face to resolve a class-related situation and/or question, an electronic mail and bulletin board system becomes an attractive alternative. Both students and instructor are allowed the capability of issuing and receiving clarification messages when the need arises, not having to wait for a convenient time when both are available to meet.

Such a system allows students to apply the computer to a very fundamental and realistic need, the need to communicate. As student's continue to successfully use the electronic mail and bulletin board system their computer ability and confidence begins and continues to grow.

The use of electronic mail and bulletin boards as an instructional supplement in the area of business application programming will allow both student and instructor increased flexibility in resolving class-related situations which arise outside of the confines of formal office hours.

BIBLIOGRAPHY

1. Battista, Michael T. and Kathleen J. Steele. "The Effect of Computer-Assisted and Computer Programming Instruction on the Computer Literacy of High Ability Fifth Grade Students." School Science and Mathematics, 84 (December 1984) 649-658.

2. Dalgaard, Bruce R. and Darrell H. Lewis. "Current Status of Computer-Assisted Instruction with Implications for Business Educators." Journal of Education for Business, 61 (October 1985) 20-22.
3. Fourgere, Kenneth and Laurie MacDonald. "COBOL: Getting Started." The Journal of Computer Information Systems, 26 (Winter 1985-86) 15-17.
4. Hausman, Louis. "The ABC's of CIA." In American Education, (November 1967).
5. Jernstedt, G. Christian. "Active Learning Increases Educational Effectiveness and Efficiency." T.H.E. Journal, (May 1982) 97-105.
6. Mondy, R. Wayne, Shane R. Premeaux and Claude L. Simpson Jr. "High School Indicators of Performance in Introductory Computer Courses." The Journal of Computer Information Systems, (Summer 1986) 5-8.
7. Parmley, Janice E. and William K. Parmley. "Computerized Applications in the Accounting Discipline of Selected AACSB Member Institutions." Journal of Education for Business, 61 (January 1986) 174-177.
8. O'Brien, Thomas C. "Five Essays on Computers in Education." Phi Delta Kappan, (October 1983) 110-111.
9. Renfro, Paula C. and John P. Maittlen-Harris. "Study Suggests Computer Time Won't Help Writing." Journalism Educator, 41 (Autumn 1986)
10. Rucker, Maurice. "Closing the Gap Between Microcumputing Technology and Business Education." Journal of Education for Business, 61 (February 1986) 231-232.
11. Simpson, Claude L. Jr., Shane R. Premeaux and R. Wayne Mondy. "The College Level Introductory Computer Course: A Student Turnoff?" The Journal of Computer Information Systems, 26 (Spring 1986) 24-27.
12. Waldrop, Phillip B. "Behavior Reinforcement Strategies for Computer-Assisted Instruction: Programming for Success." Educational Technology, 29 (September 1984) 38-41.

TEACHING COBOL IN CONTEXT

Diane M. Miller
University of Southern Mississippi, Hattiesburg, MS 39406, (601)266-5497

ABSTRACT

COBOL remains the most widely used business language, and for that reason instruction is included in most Information Systems curricula. In order to accurately reflect the nature of the language as a tool in solving business computing problems, COBOL must be presented in the context of operational, data, historical, career, system, and organizational considerations.

INTRODUCTION

COBOL is probably the most widely taught language in Information Systems curricula. The DPMA Curriculum 86 model suggests two courses in applications programming, at least one of which will almost invariably be COBOL (1). In a 1986 survey of opinions regarding which specific courses should be required for an accredited program, EDSIG members overwhelmingly cited COBOL as an appropriate vehicle language for study (2). Inherent in the DPMA's recommendation and in the survey findings is an awareness of varying business environments and a recognition that COBOL is only one of many tools, albeit a pervasive one, used in building business computer systems. Instruction in COBOL ideally should be presented from the perspective of its service as an example vehicle for business software applications, providing a solution to business problems.

However, it is easy to lose sight of COBOL training as a vehicle and not entirely as an end in itself (3). COBOL is a prosaic tool, but a workable one, for teaching concepts. Because it is a sufficient but not a necessary tool for programming instruction beyond syntax, the timing of its presentation varies. When it is taught as a first language, students may be caught up in learning the mechanics of the language, remaining largely unaware of the implications of the file handling and data manipulation they are performing. Even if COBOL is not the first language the students encounter, they may still be oblivious to considerations they have not encountered before.

Many textbooks do an acceptable job of presenting the mechanics of the language [(4),(5),(6) for example], but some do little to examine the nature of the tool. A text that attempts to address the larger issue of COBOL programming as part of a system [(7),(8) for example] may not provide as inclusive coverage of the language itself. A balanced knowledge of COBOL that fulfills the spirit of the DPMA Curriculum '86 recommendation should include both an adequate understanding of the language itself and a sense of perspective gained from examining the context in which COBOL is found in operational systems. It is not enough to assume that the necessary perspective will come from other courses in the curriculum.

Perhaps the optimal approach for providing the COBOL context along with a study of the language is to use one of the standard language-oriented texts and

provide the additional perspective via special lectures or guest speakers, since adequate texts are available to cover language mechanics. However, some contextual considerations must be tailored to reflect the local operating environment, and still others vary over time as computing advances are made.

Specifically, what topics should be considered in establishing the COBOL context? The answer may vary according to local requirements, but generally speaking the basics at minimum should include discussion of the operational, data, historical, career, system, and organizational aspects of COBOL.

THE OPERATIONAL CONTEXT

There is an obvious need to describe the COBOL operational environment as unique to the particular site where student jobs are run. Most textbooks either omit instructions for job submission or present procedures as practiced in the author's environment. Although job submission via screen entry is commonplace, most textbooks which discuss submission at all assume a card-image batch environment as the most generic [(6),(8). (7) is an exception]. Because of local unique requirements, students must be instructed early about signon procedures, command language functions, local naming conventions, Environment Division adaptations, compiler conventions, editing and card-image file building procedures, at a minimum. If students enter a course of study having taken COBOL in another environment, they will need access to this information, just as beginning students do. Because job submission information is essential to submitting the first job, this contextual information is the first provided. It is also the most difficult for students to grasp, partly because of its strangeness to new programmers, partly because of the fact that it diverges from the text, and partly because there are so many other general concepts a first-time COBOLer is struggling to assimilate. In spite of the difficulties, however, this writer's experience has been that it is worthwhile to challenge the students early, because it establishes good lab habits, defines a framework for the language, and sets the pace for solving subsequent more difficult programming problems.

THE DATA CONTEXT

COBOL applications are the workhorses of the business computing world. No other language offers the same degree of standardization and capability in handling large complex standard files. Many exciting new trends in business applications -- Expert Systems and other decision support tools, language generators and end-user computing, to name a few -- seem esoteric by comparison to the typical meat-and-potatoes application supported by COBOL. Businesses have an ongoing need for transaction processing and management information systems involving a large volume of data, repetitious processing, straight-forward and easily definable computations, many concurrent users, and fixed reporting cycles. The need for ad hoc reporting in some applications does not alter the proportionately more voluminous need for transaction-based systems which may remain in place for years, updated as needed but largely unchanged. COBOL offers sufficient maintainability for such systems. It taxes machine resources far less than its racier fourth-generation cousins, thus supporting more potential users, given the same hardware. Because COBOL is largely standard, it is largely transportable. Unlike fourth-generation languages, COBOL compilers are available for virtually any machine.

Because of a limited instruction set, highly user-friendly 4GLs may not have the flexibility to combine data retrieval, multiple parallel sorting, computations, and specialized output format that COBOL affords. Data integrity, from the standpoints both of systems security and of inadvertent misinterpretation, is easier to maintain

in a COBOL environment. Even when full-capability 4GLs are used by computer professionals to provide a large portion of a software system, about 20% of required functions may still have to be supported by COBOL, in order to provide external system interfaces.

Students should be aware of the data handling capabilities which place COBOL among the indicated vehicle languages for most business applications.

THE HISTORICAL CONTEXT

Of all standard business languages currently in use, COBOL is the oldest. Its durability alone demonstrates its basic soundness as a business application tool. It has withstood the challenges of languages designed to "replace" it -- PL/1, to some extent Pascal, and most recently Ada -- in large part because COBOL focuses more specifically than any other language upon that narrow range of large transaction-processing applications so common in the business environment. Because COBOL was never intended to answer a broad range of other applications needs, the resulting singularity of purpose has permitted it to answer the unique needs of business exceptionally well.

However, the writers of COBOL were pioneers in language design. They had few examples to learn from, and no way to anticipate some subsequent applications. In an effort to anticipate conceivable needs, a large number of facilities were incorporated that have later proved unnecessary or ill advised. The COnference on DAta SYstems Languages (CODASYL), which originally undertook the design of a COmmon Business Oriented Language (COBOL), has continued to meet for the purpose of keeping the language current. The conference has succeeded in downplaying unnecessary and/or treacherous features such as 77 levels, PERFORM. . .THRU and MOVE CORRESPONDING; and emphasizing or incorporating timely facilities like the SEARCH verb. The American National Standards Institute (ANSI) has of course formally adopted successive versions of COBOL as the language evolves, although not all manufacturers have immediately incorporated the ANSI standards in their compilers.

Even with concerted efforts to keep COBOL current, technological and philosophical changes have impacted the language. Online processing did not exist when COBOL was first written. To its designers' credit, the criterion that the language should be extensible has been admirably met: Online features have been added without fundamental change to the structure or syntax of the language. Unfortunately, the hardware environment of manufacturers differs, dictating that they accomplish online processing in a variety of ways. The various vendors have uniquely expanded their versions of COBOL to accommodate such processing, particularly screen formatting and concurrent file processing, and in so doing have sacrificed some standardization.

Similarly, structured design criteria did not exist at COBOL's inception. Dijkstra's seminal letter, "GO TO Statement Considered Harmful," appeared in Communications of the ACM in 1968, eight years after the earliest version of COBOL found widespread use. Yet COBOL has lent itself well to a structured approach, largely due to the power of its PERFORM and nested IF statements. Although COBOL has no equivalent to the DOUNTIL and the CASE construct, they can be accomplished through the use of existing statements. DOUNTIL in COBOL requires a flag tested after one execution of a procedure, or alternatively, a PERFORM followed by a PERFORM. . .UNTIL. CASE requires nested IFs, or when many alternatives become unwieldy, it takes a GO TO. . .DEPENDING ON construct.

These historical considerations document the current status of the language and predict its future adaptability. They also point up the type of life cycle

longevity consideration a system designer must take into account, and they provide a basis for comparison of both the specificity and the extensibility of other languages.

THE CAREER CONTEXT

It should have become apparent to students exposed to such discussions as the foregoing, that they are likely to encounter COBOL in the workplace. An estimated 60 to 80 percent of all business applications are written in COBOL, and the average person handles a document or form produced by a COBOL program almost every day (9). Even if the graduate never actually uses COBOL, the concepts learned by an exposure to the language will be used many times over. Since employers are aware of this phenomenon, they often use COBOL preparation as a benchmark requirement for new hires. Curriculum planners, advisers, and students alike will find such considerations relevant to their scheduling choices.

Students and faculty alike have at times protested the verbosity of COBOL. They prefer the conciseness of other languages less adaptable to the large-file business environment, like Pascal, or they like the quick results of languages whose licenses are expensive to acquire and maintain, languages which may run only on equipment produced by a few major manufacturers, like the 4GLs. But students will not have the luxury of deciding what languages they must deal with in a work environment, unless they wish to severely restrict their employability. Perhaps one of the most unkind actions a teacher can take is to treat COBOL in a simplistic fashion, ignoring its scale, complex framework, and input-output mechanics in an effort to make it seem more like the more palatable languages. Pascal, after all, was designed with deliberate simplicity to serve as a teaching language, finding unexpected acceptance for some business applications -- and 4GLs trade upon their transparency. For the sake of sound career planning and advancement, students must not be led into thinking that sanitized COBOL constitutes the real thing.

THE SYSTEMS CONTEXT

We can define the Systems Life Cycle as having the five traditional phases of Requirements Definition, Evaluation of Alternatives, Design, Implementation, and Operation. Of course COBOL, like any language tool, has a direct bearing on the activity of the Design and Implementation phases. It should be apparent also that the characteristics of the language will impact the choice of COBOL as the application vehicle language, a consideration in enunciating alternatives and determining their feasibility during the Evaluation of Alternatives phase. During the Operation phase, COBOL characteristics must be taken into account from the standpoint of program maintenance requirements. Even the initial phase, Requirements Definition, in practice is rarely approached entirely objectively, since background knowledge of language capabilities and tradeoffs anticipate probable outcomes, especially if hardware and data constraints are identified.

COBOL may also be used in an updated cycle employing prototyping, as the target language following design iterations produced by a 4GL.

In either case, the actual COBOL programming effort takes place as a sub-phase of Implementation. The act of writing programs is the focus of a COBOL course, but that focus should not preclude consideration of the systems cycle. If the language has been appropriately chosen and the design work has been done well, programming should be accomplished quickly and definitively, with no false starts. Time spent should be viewed in proportion to effort expended on other phases. Often students get the impression that most programming activity occurs in a development environment. They should be given the

opportunity to perform some program maintenance, as an example of a more nearly representative activity. Overall, students should be reminded of the systems perspective, continuing to focus upon the "why" as well as the "how" of COBOL.

THE ORGANIZATIONAL CONTEXT

Current DP organizational trends reflect the rush of interest in microcomputers, with information centers being established to provide guidance for end-user computing. COBOL still maintains a place in such an environment. We have already identified a large body of business applications that can be well served by COBOL. Such transaction processing systems can maintain the data model for an organization, either alone or interfaced with a DBMS. Extracts may be taken from the model in a variety of formats, to serve management information and decision support system needs in conjunction with end-user tools.

Useful as it is, end-user computing is likely to remain only a practical adjunct to the professionally maintained standard system. A 1984 study of the size range of applications developed by professionals versus those developed by end-users reveals that virtually no applications in excess of 2000 lines of code were developed by end users, and only 5-10 percent of the very small applications (500 lines or less) were end-user developed (10). Standard business applications frequently exceed 500,000 lines of code. Even given a tens order of magnitude saving in lines of code for an end-user tool, most users will have neither the time nor the inclination to develop large applications. However, sheer numbers of small applications in industry indicate that end-user computing will continue to be useful for special-purpose applications. Such processing will not likely make inroads into standard systems.

CONCLUSION

College trained data processing personnel need a sense of perspective encompassing a wide range of considerations. The justification for teaching languages is that they serve to convey the concepts that provide solutions to business computing problems. Ignoring the context of the language misses the point of its inclusion in an Information Systems curriculum. The COBOL context is particularly instructive.

REFERENCES

1. Richard Discenza and Fred R. McFadden, "The DPMA's Curriculum 86 Versus the ACM's Information Systems Curriculum Recommendations for the 80's," ISECON '86 Proceedings, 1986, pp. 45-47.
2. Ronald J. Kizior, "Accreditation Survey Responses - Part II," EDSIG News, vol.3, no. 4, December 1986, pp. 1, 3-5, 22-24.
3. Here we are assuming resolution to the old argument that logic must be taught as well as syntax; that premise is given.
4. L. Wayne Horn and Gary M. Gleason, Comprehensive Structured COBOL. Boyd and Fraser, 1986. This edition contains some contextual discussion.
5. A. S. Philippakis and Leonard J. Kazmier, Structured COBOL. McGraw-Hill, 1986.
6. J. K. Pierson and Jeretta A. Horn, Structured COBOL Programming. Scott, Foresman, 1986.
7. Perry Edwards, COBOL. Burgess Communications, 1986. Of the texts reviewed, this came closest to establishing the context. It also provides a forward look for COBOL.
8. Don B. Medley and Ronald W. Eaves, Programming Principles with COBOL I. South-Western, 1984.
9. David A. Kroenke, Mary Lee McElroy, James E. Shuman, and Marcia C. Williams, Business Computer Systems. Mitchell, 1986, pp. 356-357.
10. Capers Jones, Programming Productivity. McGraw-Hill, 1986, pp. 224-226.

THE EFFECT OF MICROCOMPUTERS ON COMPUTER INFORMATION SYSTEMS AND COMPUTER-RELATED SERVICES WITH IMPLICATIONS FOR THE INTRODUCTION TO COMPUTER INFORMATION SYSTEMS COURSE

Dr. Gary R. Armstrong, Associate Professor

and

Dr. Ruth D. Armstrong, Professor

Financial Administration/Management Science/Information Systems
College of Business
Shippensburg University
Shippensburg, Pennsylvania 17257
(717) 532-1434

This paper reports the findings of a survey of MIS managers and end users on the impact of microcomputers on the development of computer information systems and computer-related services. A number of curriculum implications for the Introduction to Computer Information Systems course will be presented.

Most colleges of business require all business majors to complete an introductory course in computer information systems. Frequently, such a course is offered early in the student's college career; and in many instances, a significant percentage of the course content is devoted to hardware concepts and programming principles. However, as the computer information systems discipline has matured over the last twenty-five years, business personnel and college faculty are realizing that technical expertise and programming skills are not necessarily the essential prerequisites needed by business graduates entering a computerized business world. Most persons believe that the essential knowledge about computer information systems involves the development, utilization, and management of such systems as a corporate resource.

This paper will present the findings of a study which lends credence to this emerging viewpoint--a viewpoint which implies a different focus for the required computer information systems course offered to non-CIS majors in business.

RESEARCH METHODOLOGY

A questionnaire was distributed to 8Ø managers in charge of computer-related activities to determine their views on a number of issues regarding the impact of microcomputers on activities traditionally provided by the Management Information Systems (MIS) Department, in many organizations

referred to as the data processing department. The activities under study included the design of microcomputer systems and the provision of seven computer-related services.

A total of 61 questionnaires were returned from the initial and second mailings, for a return rate of 76 percent. These respondents classified their management responsibilities as General MIS Management (62.3%), System Analysis and Design (14.8%), Computer Center Operations (9.8%), and Other (13.1%), which included areas such as EDP Information Center/User Support, Programming, and Data Base Administration.

The same questions were submitted to 140 end users representing 22 organizations, and 44 of these questionnaires were returned from the first and second mailings for a return rate of 31 percent. Of these 44 respondents, 12 indicated their level of decision making within the organization as strategic; 18, as tactical; and 14, as operational.

The size of these Pennslyvania organizations represented by the MIS managers and the end users ranged from small local companies to Fortune 500 corporations.

FINDINGS

CRITERIA FOR SYSTEM DEVELOPMENT

Both groups were asked to rate the relative importance of six factors in determining if a new application should be implemented as a microcomputer-based standalone system. As shown in Table 1, over 80 percent of the MIS management and end users rated "Required amount of interface with other existing systems" as very important. This high rating, no doubt, reflects the trend toward developing systems within organizations that can share data, thereby necessitating the need for business personnel to understand the role of local area networks, wide area networks, and other telecommunications concepts. The majority of the MIS managers and end users agreed that all of the other factors had some degree of importance in determining whether or not a new application should be implemented as a microcomputer-based standalone system.

Table 1

CRITERIA FOR DEVELOPMENT OF MICROCOMPUTER-BASED STANDALONE SYSTEMS

System Development Criteria	Rating*	MIS Mgt	End Users
Required amount of interface with other existing systems	1	83.6%	87.8%
	2	14.8	9.8
	3	1.6	2.4
Amount of storage required to support application data	1	52.5	63.4
	2	41.0	29.3
	3	6.6	7.3
Availability of application software packages	1	49.2	51.2
	2	36.1	41.5
	3	14.8	7.3
Confidentiality of application data	1	42.6	29.3
	2	36.1	51.2
	3	21.3	19.5
Knowledge or sophistication of the end user	1	29.5	41.5
	2	45.9	34.1
	3	24.6	24.4
Corporate priority of application	1	20.0	22.0
	2	46.7	48.8
	3	33.3	29.2

*1 = Very Important, 2 = Somewhat Important, 3 = Not Important

CURRICULUM IMPLICATIONS

The importance of the introductory computer information systems course in the CIS curriculum is frequently overlooked. Often it is staffed by "junior faculty" or graduate teaching assistants and taught at the freshman level. This foundation course has the reputation for being the least desirable to teach; and, as a result, the course may not receive as much emphasis as more specialized courses. Since this course may be the only computer-related course which all business majors complete, it is essential that the subject content of the introductory course be closely

scrutinized to meet the needs of business graduates who become end users interacting with information systems at various levels within the organization.

One approach for meeting the future needs of both MIS personnel and end users is to teach the introductory course from the end-users' perspective with emphasis on systems analysis and design concepts and factors affecting the development of information systems. With the number of microcomputers used in the business sector now exceeding three million, it is only logical to think that, in many situations, end users will become more involved in the development of systems --especially those utilizing the microcomputer.(1)

As the need for greater access to corporate data increases, the emphasis on developing local area networks and distributed systems will increase. With approximately 70 percent of the microcomputers projected to be linked in a network by the end of the decade, future systems developers and end users must be able to deal with a variety of interface issues and problems.(2) Management, both in the MIS Department and the end-user departments, must be prepared to develop long-term strategies for integrating microcomputers into current and future information systems. These strategies need to address design and development issues at a more decentralized level within the organization and to address provisions for maintaining these systems through improved documentation, data security, and backup procedures. Therefore, the required CIS course must make future business personnel aware of all these aspects.

SYSTEMS DEVELOPMENT

MIS managers and end users were asked to indicate who should have primary responsibility for designing a microcomputer-based standalone system. These responses are summarized in Table 2. Over 50 percent of both groups indicated that this should be done by computer professionals; however, there was some lack of agreement as to which professionals. Most MIS managers assigned this responsibility to the IC staff (25.9%) or Systems Analyst (31.0%). The end users assigned this task to the IC staff (29.5%) and Programmers (22.7%). Only 32.8 percent of the MIS managers believed system design to be the responsibility of the end users (13.8% and 19.0%), and only 13.6 percent of the end users assigned this responsibility to themselves. The "Other" category included combinations of end users and computer professionals.

For systems integrating the microcomputer and the mainframe, the MIS managers were more emphatic in selecting professional people to coordinate these activities--IC Staff (43.3%) and Systems Analyst (38.3%). Approximately the same percentage of end users selected computer professionals to design integrated systems as to design standalone systems; IC Staff (11.4%) and Programmers (43.2%). However, fewer end users selected the IC staff for this responsibility, and more of them selected end-user management for integrated systems, with a decrease in the number for "Other." Once again, "Other" was described as a combination of computer professionals and end users.

These data show that many of the MIS managers and end users responding to this survey believed that computer professionals should be involved in developing microcomputer standalone systems. The lack of agreement as to which professionals should be involved may stem from the end users' lack of knowledge regarding the roles for the different job titles. Moreover, end users showed a tendency toward believing that the responsibility for microcomputer systems should be a

Table 2

PRIMARY RESPONSIBILITY FOR THE DESIGN OF COMPUTER SYSTEMS

Personnel Choices	Standalone Systems		Integrated Systems	
	MIS Mgt	End Users	MIS Mgt	End Users
End User	13.8%	0.0%	0.0%	0.0%
IC Staff	25.9	29.5	43.3	11.4
Programmer	1.7	22.7	0.0	43.2
Systems Analyst	31.0	0.0	38.3	0.0
End-User Mgt.	19.0	13.6	8.3	25.0
Other	8.6	34.1	10.0	20.4

shared responsibility between MIS professionals and from the end user representatives.

CURRICULUM IMPLICATIONS

The importance of stressing systems analysis and design concepts is underlined by the varied responses found in Table 2. Although the level of sophistication of the standalone systems may be quite different when compared with an integrated system, the essential steps for designing and building these systems remain the same. In addition to emphasizing the systems development process in the introductory course, time should be devoted to defining the roles and responsibilities of end users, the information center staff, and the MIS department as computer systems are designed and implemented. Emphasis should be given to the need for collaboration of persons who are technical experts and persons who are specialists in the various business functional areas.

As information systems are developed at all levels in organizations, future end users and data processing personnel must develop a clear understanding of the purpose and interaction between transaction processing systems, management information systems and decision support systems.

COMPUTER-RELATED SERVICES

Both groups were asked who should provide seven computer-related services that have traditionally been the responsibility of the MIS Department. Participants were asked to indicate whether each service should be provided by the MIS Department, the IC staff, or end users. The results, presented in Table 3, show that the majority of respondents in both groups believe that each of these activities should be performed by computer professionals rather than by the end users. The only activity for which end users received over 20 percent of the responses from both groups was for "Coordination of purchasing activities." In view of the purchasing procedures of some organizations, it is easy to see why end users may be responsible for coordinating this activity after having received advice on those purchases from computer professionals.

Table 3

PROVIDERS OF COMPUTER-RELATED SERVICES FOR END USERS

Service	RESPONDENTS		
	Group Selected	MIS Mgt	End Users
Pre-purchase advice/information on hardware choices	MIS	72.1%	44.2%
	IC	26.2	51.2
	EU	1.6	4.6
Pre-purchase advice/ information on software choices	MIS	54.1	34.9
	IC	41.0	55.8
	EU	4.9	9.3
Training on use of hardware	MIS	45.9	37.2
	IC	47.5	58.1
	EU	6.6	4.7
Training on use of software	MIS	42.6	27.9
	IC	49.2	58.1
	EU	8.2	14.0
Coordination of purchasing tasks	MIS	47.5	43.9
	IC	23.0	34.1
	EU	29.5	22.0
Troubleshooting—interface between service and end user	MIS	52.5	32.6
	IC	34.4	58.1
	EU	13.1	9.3
Downloading of data from mainframe to microcomputer	MIS	70.0	51.2
	IC	21.7	25.6
	EU	8.3	23.2

CURRICULUM IMPLICATIONS

The responses of both groups indicate an agreement that end users need to have the assistance of computer professionals for the various computer-related services included in the questionnaire. Also indicative of the end-users' need for assistance is the development of information centers. A report by the American Management Association indicates that 80 percent of billion-dollar companies have information centers (IC). Therefore, students should be exposed to the purpose of the IC and what types of services are rendered.(3)

Software identification, evaluation, and selection criteria and other important components should be included in the introductory course. Although students may receive instruction in the use of several representative software packages, it is important for them to know what factors need to be considered as well as resources available to help them to make recommendations or selections in the future.

Table 4 reinforces the importance of stressing systems development concepts in the introductory course. It also supports the findings from Table 1 by showing that systems development is a joint responsibility of computer professionals and end users. The introductory CIS course must help business students to realize the help available to them from the computer professionals in the MIS Department and the IC. At the same time they must be cognizant of the responsibility of end users to work with these experts who provide information and support regarding knowledge of business procedures and requirements.

Table 4

THE IMPACT OF MICROCOMPUTERS ON THE OVERALL JOB RESPONSIBILITIES OF MIS PERSONNEL AND END USERS

Responsibilities	Percent	
	MIS	End User
The development of microcomputer-based standalone systems will require the shared responsibility of both MIS and the end user	43.3	34.1
MIS personnel responsibilities will not be changed, however, an independent group (such as an Information Center) will be formed to provide end users assistance with microcomputer-based standalone systems	28.3	45.5
MIS personnel will be assigned additional duties requiring knowledge of microcomputers and related software	25.0	15.9
End users will assume sole responsibility for microcomputer-based standalone systems	3.3	4.5

CONCLUSION

These research findings support the view that future systems analysts and end users will be required to deal with a variety of microcomputer-based issues. Topics covered in the introductory course should assist end users in understanding their responsibilities for information processing systems within the organization and the support they can expect to receive from computer professionals. By stressing end-user topics and issues in the course, students who major in CIS will have a better understanding of the end-user perspective; and as additional MIS courses are completed, the CIS major will be able to see both sides of the issues.

REFERENCES

1. Render, Barry, and others, "Perspectives on Computer Ethics and Crime", Business, Jan/Feb/March, 1986, V.36, No.1, p. 31.

2. Rhodes, Bob, "Micro Security That Makes Sense", Computer Decisions, May 7, 1986, V.17, No.9, p. 72.

3. Rhodes, Wayne L., "The Information Center: Harvesting the Potential," Infosystems, November, 1985, V.32, No.11, p.49.

INTRODUCTION TO DATA PROCESSING - A COURSE IN EVOLUTION THE NEED FOR CHANGE CONTINUES

Jerry Frances Sitek
Southern Illinois University at Edwardsville
Management Information Systems Department
Campus Box 1106
Edwardsville, Illinois 62026

ABSTRACT

Technological advances, and marketplace demands, have generated a dramatic change in the way we are teaching our introduction to data processing courses. The course has evolved from one of "nuts and bolts" computer hardware taught in the mainframe environment, to one taught with emphasis on microcomputers and their respective applications. The proliferation of inexpensive microcomputers has also caused a change in the computer literacy knowledge-base of the student entering the course. Younger students enter the course with some degree of computer literacy while their older counterparts are generally computer illiterate. The solution may lie in altering the structure to include offering the course as two separate courses; one dealing with concepts, the other with applications. By adopting this approach to the problem we meet the needs generated by the students' varying backgrounds with minimal disruption to curriculum and teaching load patterns.

BACKGROUND

CHANGING CLASSROOMS, CHANGING STUDENTS

The current trend in introductory data processing courses is one of a shift away from the traditional concepts taught in a mainframe environment, towards one where the emphasis is placed on microcomputers and their applications. Programming in a high level language has given way to learning how to use specific microcomputer application packages. These changes were made in response to a recognized need based on changing technology and marketplace conditions.

We have been caught up in the process of trying to achieve an acceptable balance between teaching traditional course content and the fundamentals of microcomputer-applications, while conforming to accreditation guidelines, and departmental policies. Considerable effort and energy has been spent in defining the contextual issues that make up the course content [1,3,4,5] and in developing a course outline to meet them. However, while we were busy adjusting our course content, the computer literacy knowledge-base of the potential student population was also changing.

The public school system has made a considerable impact on the computer literacy of the students enrolling in the introductory course [6]. The current trend is toward a greater number of students entering the classroom with some level of computer literacy (usually microcomputer-based). Their knowledge base is not uniform, the younger student is better versed than his older counterpart. A single course, structured around teaching to the mid-range (computer-illiterate) student will no longer work.

Today, the majority of students entering the classroom are polarized at the two

extremes of computer literacy. Craig Van Lengen, of Northern Arizona University at Flagstaff, reported that his students are entering the introductory course with a greater degree of computer literacy in the area of concepts [8] while this author has been experiencing just the opposite, with a greater number of the students having stronger microcomputer application(s) backgrounds. It is the contention of this author that the backgrounds will vary by age group and geographic areas. And, that each group brings into the course their own specific needs.

The older, returning (or continuing/part-time) student enters the course with little if any computer exposure. Their needs are the greatest in terms of computer literacy. Students in this category are not familiar with the terms, and usually have never worked on a computer prior to the course. While younger students entering soon after high school enter with a variety of skills and skill levels; depending upon the demographics of the area in which they obtained their secondary education [6]. We can no longer allow the course to evolve based only on the technological advances of the industry. The time has come to begin to restructure the course format to meet the needs of the changing student population that make up the course enrollment. The new structure should be flexible enough to deal with the changing student population's pre-course computer literacy, yet maintain a method to deal with the older student's needs. One course can no longer meet such diversified needs.

The solution may lie in altering the structure to include offering the course as two separate (shorter) courses; one dealing with concepts, the other with applications. Students with adequate skill levels in both, proficiencies out of the course, while those with only concepts or applications skills takes the section they are deficient in and tests out of the other. Students entering the program with neither background can enroll in the course as it currently exists. By adopting this approach to the problem, we meet the needs generated by the students' varying backgrounds while minimizing the disruption to curriculum and teaching load patterns.

We should not approach structural changes without further investigation into the students' changing backgrounds Local demographics, industrial base, public school funding, even the size of the surrounding community may impact on the students access to computer technology in their pre-college education. What works well in one university environment in fact not work as well in another.

PROCEDURES

This author is currently conducting research into the computer literacy background of students enrolling in the introductory data processing course. The School of Business course titled, "Applied Computer Concepts," is a four-credit hour course, covering the following content areas: computers and their capabilities (3%), hardware and software concepts (5%), an overview of programming languages (5%), BASIC programming concepts (35%), software packages on the microcomputer (35%), uses of computers in life (15%), and future impact and uses (2%). Students come from both the metropolitan area and the surrounding (agricultural-based) rural communities.

A survey instrument was used to obtain the data which has been collected over a period of six months, and reflects students enrolled in the day, evening, and weekend sections of the course. The survey results follow.

THE FINDINGS

The survey was administered to a total of three hundred and twenty-five

students; three hundred and seventeen (97.5%) were usable. There were slightly more males (one hundred and sixty-five (52%)) than females (one hundred and fifty-two (48%)) in the sample. The respondents ranged in age from seventeen to fifty-four; with fifty-nine percent (186) between the ages of seventeen and twenty. Figure 1 represents the age distribution.

AGE	RESPONSES	PERCENT
17-20	186	59
21-25	65	20
26-30	22	7
31-54	44	14

Figure 1 Age Distributions

Sophomores made up one third of the class rankings, with thirty-three percent (105). The second largest group by class ranking was freshmen with twenty-seven percent (85). Figure 2 represents the class ranking distribution.

CLASS RANK	RESPONSES	PERCENT
Freshman	85	27
Sophomore	105	33
Junior	81	25
Senior	34	11
Graduate	6	2
Other	6	2

Figure 2
Class Ranking Distribution

Eighty-three percent (261) of the students were classified as full time.

The categories for majors were limited to the following: business non-MIS, business-MIS, computer science, other disciplines, and undecided; in addition a category was included for those students not taking the course towards a degree. Business non-MIS majors made up the largest response category, with forty-seven percent (148). Figure 3 reflects the distribution among the majors.

Major	Responses	Percent
Business-non MIS	148	47
Other disciplines	52	17
Undecided on major	49	16
Business-MIS	46	14
Computer Science	11	3
Non-degree student	11	3

Figure 3
Distribution by Majors

Despite the fact that eighty-nine percent (281) responded that their family did not own a microcomputer, seventy-two percent (227) of the students indicated that they had used one prior to enrolling in the course. Of the microcomputer literate students, seventy percent (159) stated that they had acquired their skills in a classroom; eighty-two percent (130) at the high school level (fifty-nine percent (93) as part of an introductory course).

Students with microcomputer backgrounds were asked to indicate all the applications areas that they were familiar with. Not at all surprising was the largest response category, games (with sixty-four percent (145)), followed closely by word processing (with fifty-seven percent (130)). The two hundred and twenty-seven students who had used a computer, generated five hundred and two responses (an average of 2.26 applications per student). Figure 4 represents the application areas responses; the percentages are based on the total number of responses divided by the total number of students in the group (227).

Application Areas	Total No. Responses	Percent of Students
Games	145	64
Word Processing	130	57
Spreadsheet	59	26
Graphics	57	25
Data Base	52	23
Telecommunications	13	6
Other	46	20

Figure 4 Application Area Responses

Although ninety-one percent (287) of the students indicated that they had never used a mainframe computer, eighty-nine percent (202) responded that they had written programs in a minimum of one high level language. Of these, eighty percent (161) in BASIC.

Survey findings indicate that the students believe knowing how to use a microcomputer is more important than knowing how it works by 2:1. History, numbering systems, diagramming and career opportunities ranked low on their list of priorities.

It is currently possible for a student to proficiency out of the entire course. The student must successfully prove they know technical concepts/terms and demonstrate a working knowledge of: a programming language, a data base management system, a spreadsheet, and a word processor. Given that the background of the students vary so widely, only a minority of those enrolled have an adequate background to successfully pass the exam. The respondents were asked if a proficiency exam were made available for just the applications content of the course, would they take it. Sixteen percent (46) of the students indicated they would take the exam. However, given the current structure of the course, it is not possible to proficiency out of either half of the content area (concepts or applications) of the course.

CONCLUSIONS REACHED

The data supports the premise that the students in today's introductory data processing course comes from a diversified background. The majority of the students in the course already have some degree of computer literacy, primarily in the areas of microcomputer applications. However, like their non-microcomputer literate counterpart, they generally lack an adequate background in fundamental data processing terms and computer concepts and are unable to proficiency out of the course (as it currently exists). Under the course's current structure, the instructor is often faced with teaching the class to the mid-range student, while frustrating the older students and boring the younger ones.

When we establish the contextual issues of course content, we can no longer afford to ignore the computer literacy knowledge-base of our student population(s). The computing background (or lack of it) of the actual student enrolled in each specific university's course needs to be studied in order to identify and assess their needs. Course currency need not interfere with academic freedom. We need to be looking at those specific students and what skills they are or are not bringing into the classroom(s), as these skills can have major impact on what we are teaching and how we are teaching it.

This author contends that the course can be divided into two related subcourses; one addressing traditional computer concepts coverage, the other microcomputer applications. Students could then take the subsection they lack adequate background in and proficiency out of the other. Those students who lack adequate computer literacy in both content areas would be required to enroll in a course section that addresses concepts and provides hands-on experience.

Establishment of this would require a commitment at not only the department/school level but at the university level as well since, for the most part, it is a multidiscipline course taught by faculty from various departments campus-wide. Furthermore, given that we do have student population that enrolls without any computer background, we must continue to offer sections of the course as it currently exists. These are in addition to the needed speciality sub-sections. And, we must do this in an environment that lets us maintain course currency as technology advances by leaps and bounds.

Regardless of how the individual university seeks to solve this problem, they need to establish and retain a commitment to identified student needs, and to incorporate these into their structuring of the format of their introductory IS course. Once identified, the course content can be structured to maximize meeting these needs while working within the flexible framework of the accreditation curriculum regulations.

REFERENCES

[1] Guynes, Valnecek, and Zant, "Computers and Schools of Business, A Comparison of the ACM Curricula in IS and CS," Interfaces Fall 1983, Vol. 5, Issue 3, pp. 36-41.

[2] Howard, Lahey, Murphy, and Thomas, "Student Profiles and Suggestions for Improvement in the Introductory Computer Center," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, October 25-26, 1986, pp. 167-172.

[3] McCleod, "The Undergraduate MIS Course in AACSD Schools," Journal of Management Information Systems, Vol. II, No. 2, Fall 1985, pp. 73-85.

[4] Sackson, "Curriculum for Introductory Computer Courses for All Students," Interface, Summer 1986, Vol. 8, Issue 2, pp. 32-35.

[5] Sumner and Schrage, "Microcomputers in the MIS Curriculum," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, October 25-26, 1986, pp. 149-152.

[6] Tellup, "Computer Literacy of Entering Freshmen," Proceedings of the Annual Convention of the Association for Educational Communications and Technology," Anaheim, January 17-23, 1985.

[7] Torardi, "The Development of a Computer Literacy Assessment Instrument," Proceedings of the Annual Convention of the Association for Educational Communications and Technology, Anaheim, January 17-23, 1985.

[8] Van Lengen, "An Alternative Structure for the Introduction to Computers Curriculum," Interface, Spring 1986, Vol. 8, Issue 1, pp. 34-35.

Keeping Control of Your Microcomputer Laboratory

Janet M. Cook
Department of Applied Computer Science
Illinois State University
Normal, IL 61761

ABSTRACT

The recommendations given are intended to keep a new or existing microcomputer laboratory functioning efficiently and legally regardless of the type of microcomputer used. Some of these recommendations are standard; others are drawn from new technology, products, or law suits. They have been brought together so that you have a single source to cite when you put in your budget request for laboratory support for next year.

HARDWARE

PROVIDE A SAFE ENVIRONMENT

Remove all carpet or use anti-static treatment on it in dry weather.

Remove chalk boards and chalk. Use posters for messages and over-head projectors for teaching to eliminate chalk dust.

Provide air conditioning and humidity control. IBM specifies for the PC, PCjr and PC/XT,

> temperature 60° to 90°F when the system is ON, 50° to 110°F when the system is OFF, humidity 8% to 80%;

and for the Expansion Unit,

> temperature range as above, humidity 8% to 80% when the unit is ON, 20% to 80% when it is OFF.

Personal Computing Magazine says (1), "nearly every computer" needs temperature 65° to 85°F when the system boots and runs.

Keep a desk-top thermometer and humidistat in the lab among the machines. Micros give off heat and so do students. The temperature in the vicinity of the CPU will be higher than the temperature registered by the wall thermostat. On hot days have a lab monitor adjust the room air-conditioning or the thermostat to keep desk-top temperature and humidity within the safe range.

For a lab with 10 or more micros, provide a room air-conditioner. The machine and body heat generated on an 80-degree day are too great for central air-conditioning to handle. The local unit also keeps the lab at operating temperature and humidity on nights and weekends when central air-conditioning may be turned down.

Provide surge protectors for each machine to isolate effects of accidents to single units.

Provide a single power-on switch for each machine so that CPU, monitor and printer come on together.

Mount cables and connections on the under side or back of work surfaces, away from feet, chair legs, and anything that will catch on the cables, and away from spills and cleaning agents used on the floor. (Besides, the custodians will do a better job of keeping down dust if they don't have to vacuum around a macrame of cables.)

Have work surfaces wide enough so that students' disks needn't be placed on or leaned against the CPU, monitor or modem, where magnetic fields can damage data, and disks can block the units' ventilation grills.

Cover the machines at night with plastic - fitted covers or drop clothes - especially if there are water pipes running through the ceiling or overhead sprinklers. Otherwise, one wastebasket fire or broken pipe can wipe out every machine in the lab. (The electrical connections are already above flood level, right?)

DISHEARTEN WOULD-BE THIEVES

Even casual observers should see that the equipment is secured. Bolt CPU's to the work surfaces or attach an alarm that will sound if the CPU is unplugged. Post signs: "These machines protected by ...".

For daytime control, have only one access door, placed where it is under observation whenever the lab is open. Your staff may not have time to watch the door continuously, but the likelihood of observation by someone who will speak out is known to be a deterrent to theft.

For evening and weekend hours when the lab is open, provide a student monitor to help students, report problems needing repair, and make it obvious that the lab is always attended.

For protection when the lab is closed, install window and door alarms and a motion sensor.

KEEP TRACK OF WHAT YOU HAVE

Put an identification number on the front of each component, to simplify inventory checks and to identify what is lost when you have a calamity or theft.

In the department office, keep a record of (1) the serial numbers, purchase date and warranty/service contract data for each component; (2) which components are linked together in a single system; and (3) where each system is.

Keep a back up copy of the department folder where it won't be lost to the same fire or flood that hits the hardware. You'll need that data for insurance claims.

SET UP A FEW RULES FOR LAB USERS

No food or drink in the lab.

No smoking in the lab. Computers do get cancer of the disk from air-borne tars and oils.

Turn down, not off. I.e., blank the screen but leave the machine on until the lab closes, to avoid the wear of frequent 'cold boots' while avoiding screen burnout.

No radios or tape players in the lab. They generate weak magnetic fields.

Don't move the equipment.

Don't fix the machines. It voids the warranty.

SOFTWARE

Use public domain or textbook-bundled software that students can keep to use in later courses. Pirated copies of 'Name Brand' packages aren't needed to teach effective word processing or spread sheet skills.

If software disks are not sold with the textbook, sell/give each student a copy of the public packages used via a materials fee, from the department, from the bookstore, or from a local Copy Shop [2], making sure that students are charged only for the disks, not for the free programs they contain.

Emphasize the distinction between proprietary and public software.

(1) Distribute one of the ADAPSO booklets, "Thou Shalt Not Dupe", [3] or an academic adaptation of ADAPSO's policy statement. [4]

(2) Color-code department-owned disks. Five and 1/4" disks come in 10 or more different hues. Using different bright colors for public-domain and copyrighted software makes plain which disks are proprietary, which are public, and which are students' own black disks.

Place departmental software on hard disk or write-protected disks wherever possible, to minimize the threat of accidental erasure of the program by inexperienced users.

Site-licensed software should be checked out for use in the laboratory only, not on the students' machines at home.

These strategies won't prohibit illegal copying of programs, but they will help you demonstrate that you discourage illegal activity. They might even keep you from being named in the lawsuit if students are found violating copy-protection on popular packages. Courts do ask whether reasonable and standard precautions have been taken.

DATA

Expect each student to have at least two formatted personal data disks, one for current work and one for back-up. If a course provides one or more program disks, you might include two formatted data disks in the package bought by each student.

Where software permits, require data and work files to be kept on the students' personal data disks so that the department disks do not fill with student data. If you have hard disks, erase unauthorized files at undeclared times every week. Students quickly will learn to keep their data to themselves.

Master data disks required for class assignments can be made available by check-out from the department or from the library's Reserve Book Room. They should be a special color to make it easy to see that the disk returned is one that was checked out.

ACCESS TO LABORATORY

If there are not enough work stations for all users at all times, issue an identification card on the first day of class in courses which use the micro lab. At busy times of day, week, and term, have a lab monitor on duty. If people are waiting to use machines, the monitor may ask to see the user's ID card and problem assignment sheet. Students not working on current class projects may be directed give up their machines to people in line.

POSTERS

To accustom lab users to your practices and regulations, put vivid, funny posters around the room. As advertisers know, even when a poster is too well-known to be reread, the familiar colors and pattern will be a subliminal reminder of the message. Some sample posters I'm making are:

(1) Pile of disks spills off a shelf; chair caster rolls across one disk. Legend: "Oops! Got a backup?"

(2) Keyboard with coffee spilled across the keys and dripping out beneath. Legend: "Coffee! Right down the old keyboard!"

(3) Disk melting over the edge of a radiator. Legend: "Talk about floppy disks!"

(4) Desk top littered with dirty mug, crinkled candy wrappers, potato chips, etc. Disk with cartoon face, hand scratching inside its jacket. Legend: "What a crumby place to work!"

(5) Ashtray with cigarette smoke drifting across the front of the disk drives. Legend: "Caution: Tobacco smoke causes cancer of the disk."

(6) Center: Flabbergasted man stares at grinning dog. Legend: Top - "Originals never leave the lab." Bottom: "The dog ate my floppy disk!"

(7) Repairman whose cover-all says COMPUTER SHOP stands over opened CPU talking to worried owner. Legend: Top -"Don't fix the machines." Bottom: "Sorry, we don't clean up after do-it-yourselfers."

(1.) "Answers / This Month's Queries from Our Readers", Personal Computing, March, 1987, p. 201.

(2.) Kinko's, for one, will make copies of public domain software for distribution to students, together with printed copies of the documentation.

(3.) ADAPSO prints two booklets with this title. Both state "We encourage you to reproduce and distribute copies of this brochure." Samples are available free from ADAPSO; 1300 N. 17th Street, Suite 300; Arlington, VA 22209; (703) 522-5055.

(4.) An academic version of ADAPSO's policy is found in my paper, "Defining Ethical and Unethical Student Behaviors Using Departmental Regulations and Sanctions", In the 1987 ACM SIGCSE Proceedings.

REALISTIC SYSTEMS DESIGN PROJECTS

Dale K. Hockensmith
Department of Computer Technology
Indiana University - Purdue University
Fort Wayne, Indiana

ABSTRACT:

Typical systems development projects are relatively small and can be managed fairly easily by the instructors. Those projects are normally of such a magnitude that they can be analyzed, designed and implemented within a one semester course. However, the use of larger, integrated projects over several semesters gives the students a more realistic view of systems development as it really exists in the business environment. One of the major drawbacks to this approach is the increased effort required of the instructor.

INTRODUCTION:

We have heard much said about providing students with the opportunity to study the system development process on a first-hand basis. The prevailing philosophy seems to be that student teams be formed to work on some small project that can be analysed, designed, programmed and implemented within a one semester course. Our feelings on the subject are that this approach, although more easily managed from the standpoint of the instructor, gives the students a false sense of what is truly required for proper systems development. Their expectations may become that all problems can be resolved within a few weeks, regardless of the complexity.

The system development courses should provide more than just technical knowledge. The student's benefits from the experience should:

Increase his analytical skills

Expose him to specific systems commonly found in the business environment

Improve his logical thinking abilities

Instill in the student a sense of professionalism (1)

Teach him to plan for the future.

OBJECTIVE:

At IPFW we have developed a series of courses to address the processes involved in systems development that, we feel, do not give that false sense of being able to solve all problems within a short period of time. These courses run the student through all phases of system development from programming through analysis (2). Our work not only involves the learning of the technical aspects of system design but also how various subsystems must

interface in order to function properly and how the analyst must anticipate what new systems will sometime in the future interface with the existing one.

INTRODUCTION TO SYSTEMS ANALYSIS AND DESIGN:

In the freshman year the students are exposed to systems analysis and design in a fairly cursory manner. They learn what systems development involves from the initial user request through the analytical, design and implementation stages. Rather than delving deeply into the processes, we are content to give the students a broad overview. They are shown what a systems analyst does. They are introduced to structured methodology and the various tools used by the systems designer (3) in a manner that will aid in their retention of that knowledge. In no way do we expect the students to be able to carry out the tasks of an analyst; we do expect that the students understand those tasks.

DETAIL SYSTEMS DESIGN:

In the sophomore year the students are exposed to their second system development course. Now that they have the understanding of what is required in system design they are given the opportunity to experience it on a first-hand basis. It must be noted that we have chosen to follow the general advancement pattern found in industry (i.e. programmer, junior analyst, senior analyst) and their related tasks rather than follow the sequence found in most textbooks (analysis, design, implementation).

The course is an intensive study of detail design methodology and the specific tools used in detail design (4). Students are expected to utilize those design tools in their semester project.

The student teams are provided with system functional specifications for a major subsystem that must be developed. (Last year our project involved the design of a Material Requirements Planning system(5).) To aid the teams in their design they are provided with a sample detail design document so that the required format will be followed.

With the above documentation the teams are expected to develop the new system design documents that can be used by a programmer/ analyst to write and test the new system. One facet of the project that is not often found in the normal academic environment is that the new subsystem must properly interface with other existing subsystems that have been developed by prior classes as well as anticipating the interfaces for future developments.

BUILDING AND TESTING THE SYSTEM:

Although we would prefer to expose the students to this intense programming experience prior to their detail design project, we are limited by time constraints. The students are again involved in a team project; this time they will program and test the subsystem which they designed during the prior semester. Here the emphasis is not only on improving their programming skills but also on developing the types of documentation that should be (but rarely are) found in industry.

Because we have indicated to the teams that the new system under development must interface with existing systems, we provide master files that have been previously developed which must be utilized. Data that are entered manually are considered 'suspect' until proper edits have been performed to assure their accuracy. We stress that, because of the changing conditions within the workplace, programs must be "elegant" (6) so that they can be

quickly modified when (not if) conditions change.

GENERAL SYSTEMS ANALYSIS:

In the junior year the students are able to perform an analysis of the new system. They again are divided into teams and given a set of system requirements. They then have the responsibility to produce a set of system functional specifications similar to what they had been given the prior year.

Because the students have already worked on other subsystems within the total system, they are aware that their new project must properly interface with what has been previously developed. This is an aspect that we feel is not addressed in curricula at other institutions.

Starting with the assumption that the subsystem under study is currently performed manually, the student teams determine the functions that must be provided by the newly requested system. Again, we emphasize that flexibility and simplicity in design are important to enable rapid changes to be made to the system as business conditions change. It goes without saying that the proper structured analytical tools must be utilized.

When the system specifications are completed the students are required to perform a Structured Walkthrough and defend their solutions to the problems presented them. This not only makes the task of analysis more realistic to them, but it also gives them experience in making the presentations which will be required of them in the workplace.

CONCLUSION:

The use of coordinated (and realistic) projects within the system development courses avoids the false impression that program and system development is a simplistic exercise. The students can see that their tasks are difficult but achievable. The students themselves have indicated that the projects we have worked on do indeed reflect reality. As one student indicated, she was working on the same tasks as a friend in industry -- same system, same degree of difficulty.

NOTES:

I. The following documentation is provided by the students for the detail system design

   System overview

   Logical system flowchart (DFD)

   System file descriptions

   Functional specifications for each program within the system

   CRT screen layouts (if applicable)

   Output report layouts

   Time and cost estimates for system development

   Interfaces with existing system

II. For the programming applications project the student teams must produce not only properly structured and well documented programs, but also a detailed user manual including:

   General systems overview

   Glossary of terms

   Description of each system function

   Input form or CRT layout

Sample output report (if any)

Explanation of all input data (valid data types, value ranges, mandatory fields, etc.)

Error messages that may be encountered and the steps to be taken to correct the errors

Troubleshooter's guide

III. For the Systems Analysis of the project we require:

System overview

Basic concepts of the manual system

Forms design and data flow

Organizational responsibilities

On-line processing functions

Batch processing functions

Existing system modifications

Batch reports to be generated

Implementation schedule

REFERENCES:

1. Fritz, Jane M.
"A Pragmatic Approach to Systems Analysis and Design"
ACM SIGCSE BULLETIN volume 19, number 1, February 1987, pages 127-131

2. Barrett, Robert A.
"A Five Course Sequence for Information Systems"
ACM SIGCSE BULLETIN, volume 14 number 1, February 1982, pages 114-122

3. Edwards, Perry
Systems Analysis, Design and Development
Holt, Rinehart and Winston (1985)

4. Orr, Kenneth T.
Structured Systems Development
Yourdon Press (1977)

5. Hockensmith, Dale K.
"Coordination of System Development Courses"
ACM SIGCSE BULLETIN, volume 19, number 1, February 1987, pages 529-531

6. Dijkstra, Edsger W.
EWD648, "Why is Software so Expensive?"
Selected Writings on Computing: A Personal Perspective
Springer-Verlag (1982)

STUDENT PROJECT GUIDE FOR SYSTEMS ANALYSIS AND DESIGN COURSES

John T. Overbey, PhD, CPA
Department of Accounting and Information Systems
Western Carolina University
Cullowhee, NC 28723

In my first few years of teaching systems analysis and design the students and I both went through a great deal of frustration over the course project. As a result of this mutual frustration I began the development of the Student Project Guide which consists of a series of specific tasks for the students to perform in completing their project and a series of descriptions and explanations to aid the students in their work. A portion of the Student Project Guide is included in this paper. The Student Project Guide also includes some forms to aid the instructor in evaluating and controlling the student projects.

STUDENT PROJECT GUIDE FOR SYSTEMS ANALYSIS AND DESIGN COURSES

By now it is almost universal practice that in any systems analysis and design course the student is required to do a system analysis and design project. For the first several years I taught such a course both the students and myself went through a great deal of frustration getting the project started and the student's frustration tended to continue throughout the project over the question of exactly what they were supposed to do. As a result of this mutual frustration I began the development of the Student Project Guide.

WHAT

The Student Project Guide consists of a series of specific tasks for the student to perform in completing their project and a series of descriptions and explanations to aid the students in their work. A portion of the Student Project Guide (hereinafter referred to as the SPG) is included in this paper. The SPG has four main purposes: (1) to assist the students in getting started on their project, (2) to give them specific information about what they are expected to produce, (3) to aid them in doing the work, and (4) to provide a framework for organizing the term's work. The SPG also includes some forms to aid the instructor in evaluating and controlling the student projects. Two sections of the SPG are included in the Appendix: Systems Project Description and Phase I. These are primarily to aid the students. Other forms assist the instructor in control and evaluation of the students' projects are discussed below.

HOW

At the beginning of the semester the following portions of the SPG are

distributed to the students: Systems Project Description, Phase Outline, Steps for Phases I-IV, and Phase I. They are also given an assignment schedule for the semester showing when we will begin each phase, when they must have it in rough draft, when they must have each phase in semi-final form and when the completed project is due to be turned in. As I use the terms in this setting, "begin" means when I will discuss the phase in class, "rough" means it must be in writing (pencil is ok for rough) and "semi-final" means it must be on the word processor. They are free to change anything up to the final hand-in date for the entire project.

We begin the project the first week and it is due during the last week of class. Class time is divided approximately one-half for explanations of the steps in the SPG and related material for the text and one-half for group meetings. In the class time group meetings I review what the groups turn in, give them suggestions for improvement, and answer their questions. They must, of course, have additional meetings outside of class.

The Instructor Control Form gives a summary picture of the status of all the groups in the class. The Instructor Group Comment Form can be used to make weekly notes on the status of a particular group. One of these forms is maintained for each group. The Project Evaluation Form is used by the instructor whenever it is desired to have a tentative or final grade for the project. I use it at mid-term to give the students an idea of how they are doing and then at the end of the term to determine the final project grade. The Group Member Evaluation Form is used by the students to evaluate their group members at mid-term and at the end of the semester. Both the Project Evaluation Form and the Group Member Evaluation Form are given to the students for information purposes at the beginning of the semester. The Group Work Distribution and the Group Meeting Attendance Form are maintained by the students and periodically reviewed by the instructor.

SUMMARY

The Student Project Guide was developed to aid systems analysis and design students and instructors in the completion and control of the systems project in the systems analysis and design course. I have used it in several sections of the undergraduate course where the project constituted essentially the whole course with considerable success. For the students it has provided an answer to such questions as what are we supposed to do next, what is it supposed to look like, how does it fit in with the overall project, etc. For myself as instructor it has provided a way or organizing, controlling, and evaluating the course and the student projects.

I have also used it in my graduate Information Systems course in the MBA program. In this course, which is an overview of information systems in the organization and current information systems technology, I use the Student Project Guide in a different manner. I give all the phases and steps to the students and have them work in groups as in the undergraduate course but I omit many of the steps and require only samples from some of the other steps (e.g. two output forms as opposed to all output for

the system). The Student Project Guide has been valuable in this course in showing the graduate students (who almost all have non-information systems undergraduate majors) what an information system is, how it is designed, and what the role of management is in this process.

APPENDIX

SYSTEMS PROJECT DESCRIPTION

The purpose of this project is to give you an opportunity to apply as many of the steps in the systems life cycle as possible to a realistic systems problem. The first thing you must do is to form your group. Your group will use the handout material from the Student Project Guide as a guide in completing your project.

The first Phase of the project is concerned with defining the problem, organizing your group and establishing a relationship with the client. Do not jump to solutions at this point.

Your next step should be to describe the present system in that area. Your description should be comprehensive (document flowcharts, systems flowcharts, organization charts, job description, controls used, brief narrative descriptions, etc.) and should include a statement of the main problems your team finds in your analysis. Early in your work, you should clearly define project objectives, scope and requirements. You should define time and priority elements - what are your deadlines and what steps depend on the completion of other steps. Also, early in your work, you should establish your project control system - when will you meet? Who will do what? How will it be coordinated? etc.

You are to assume that you will be purchasing hardware and writing at least a portion of the software. A partial RFP will be required.

The most important part of the report will be the description of the new system (systems flowcharts, document flowcharts, organization charts, job descriptions, controls used, narrative descriptions, etc). Great attention should be given to the question of controls. You must include a description of your training and implementation plans and any expected training and implementation problems. Your report must have a table of contents, an introduction, and a summary. Your report will be evaluated on such things as the following: organization, controls, flow charts, present system description, design alternatives, forms, implementation plans, interviews, questionnaires, input layouts, output reports, etc.

PHASE I: DEFINITION OF THE PROBLEM AND THE PROJECT

Objectives of Phase I:

The principal objectives of Phase I are:

(1) to determine the nature of the user's system problems

(2) to develop a working relationship with the user

(3) to arrive at a preliminary definition of the project

Task

(1) Define the problem

(2) Define the project objectives, scope and boundaries

(3) Establish project control system

(4) Define time/priority elements

(5) Prepare engagement letter

Explanation and discussion of tasks to be performed in Phase I:

1. Define the problem

Purpose: To insure that there is agreement between analyst and user/management as to what the actual problem to be solved is. Preliminary to the preparation of the initial statement of the problem there must be an extensive discussion between the analyst and the person requesting the systems engagement (usually the user). After this discussion the analyst prepares a memo to the user stating what his/her impression of what it is that the user wants. When both parties are satisfied with contents of that memo, both should initial it. This memo and the one from step two become the basis for the engagement letter which in effect is the initial contract between the analyst and user. It may be that later on in the course of the engagement the user or the analyst may determine that this initial statement of the problem should be changed, but both parties must be in agreement about this at all times.

2. Define the project objectives, scope and boundaries

Purpose: To determine what work must be done to solve the problem defined in step one. The objectives of the project should specify why the project is being undertaken, e.g., the project of designing a new accounts payable system might be undertaken for the purpose of getting the bills paid on time. (One objective of your project is to help you pass the course!) Scope and boundaries have to do with what is to be included in this engagement and what is not, how far the project is to go and where it is to stop. For example the scope of a payroll system includes calculating wages, writing checks, taking care of payroll taxes, etc., but the preparation of the balance sheet and income statement are outside the boundaries of a payroll system. The scope and objectives memo must state what the objectives, scope, and boundaries are for this project. This document must have the agreement and initials of both user and analyst.

3. Establish the project control system

Purpose: To determine who is to do what. This Student Project Guide along with your instructor will tell you what work must be done to complete your project. Within your group you must decide such things as who is going to do what, where are you going to meet, etc. The suggested output from this step is called an "internal memo". The term "internal memo" means that this memo is one that is prepared by you and remains in your organization. It does not go to the user. This memo will probably have to be expanded as you go along. (Steps 4-5 omitted)

SYSTEMS ANALYSIS: A PROJECT APPROACH WITH VISIBLE PAYOFFS

S.K. SHAH
HOFSTRA UNIVERSITY

ABSTRACT

This article presents an approach to educating students in information systems. Following prescribed guidelines, students are required to carry out team projects which involve analysis, feasibility report, and system design of operations in actual companies. The beneficiaries of this approach are: students, academic and business communities.

INTRODUCTION

A systems analysis course carries with it the usual disadvantages of a highly theoretical framework unless it is transformed into a strong tool by its applications in the real world. It requires the teacher to make the very abstractness of its concepts into vehicles for developing the students' thinking process along analytical lines. The objective of the course should be to make the students not only well-versed in the theory as a tool of analysis and design, but also give them a relevant and exciting experience which enhances their understanding of the concepts because they are able to practice them.

The project approach to a systems course dates back to the early 1970's. The short-term goal brings as much realism into the course as possible and captures student interest in system concepts and methodology by direct involvement. Having worked in the systems field for several years with Fortune 500 companies, and retained interest and contacts in the profession, it is only natural for me to incorporate in the course salient features of the real-world experience.

BUSINESS EXPECTATIONS OF SYSTEMS TRAINING

The common sentiment in the business world suggests that a lead time of six to twelve months is necessary to make systems persons productive once they are hired. Depending on the complexity of the organization, it is believed that newly hired persons need--

- about one month to orient themselves to the physical environs of the work place;
- two to four months to acquaint themselves with the people and organizational authority and responsibility linkages;
- four to six months to learn about company policy, process, and practice to delve meaningfully into different problem situations.

For many employers, personal skills

and aptitudes are almost as essential as the experience which a system person brings to a job. It is obvious that people with experience and special skills are likely to be productive in a lot less time than people without. Usually students fresh out of college are not familiar with the ways of the systems world. Granted today's youth is intelligent, well motivated, and purposeful. The question is what can we in the academic world do to help them in becoming productive sooner and be themselves in the process of adjustment to their new environment. The project approach offered a unique method of developing the multifarious aspects of the future business persons' technical and inter-personal skills.

A PROJECT APPROACH

The course is designed to familiarize the student with various activities associated with the task of systems analysis in an organizational setting. The focal point of the course is the term project.

METHODOLOGY

Students form three to four person teams for a formal systems project. Each team selects a particular firm to research. In selecting the firm, students are encouraged to seek help from working parents, friends and relatives, previous employers, teachers, and secondary sources, and/or make cold calls. Once a contact is established, students conduct a preliminary investigation to familiarize themselves with the company management, its philosophy, products, markets served, etc. prior to setting up a specific appointment.

Student groups meet frequently to chalk out their own objectives, strategies, identify individual roles based on skills, strengths, and schedule activities to accomplish their task. These activities range from meeting with managers to identify a specific problem and their verbal acceptance of a feasibility report to the final in-class presentation and defense of the completed task.

The study of information systems is viewed from developmental and/or diagnostic points of view. In a developmental approach it is assumed that in the normal course of events, it is the system goal that dictates system design which is then implemented after appropriate tests. The said system is monitored as it operates, and changes, if any, are observed and modified in light of the system goal. Among the questions raised in this analysis are--

1. To what extent does the system achieve specified goals? Why or why not?
2. Were the means used to implement the system adequate for the task?
3. What caused the system to change? Constraints?
4. How does the environment influence the system in the success or failure of its mission?

Diagnostic approach on the other hand, examines a given system for its yield versus expectations at any point in time and focuses on the following steps:

1. Identify system malfunction(s).
2. Find clues or symptoms for grouping them into appropriate causes.
3. Suggest short and long range remedies to improve system operations.
4. Generalize solution to future design or operational practices to prevent

system malfunction(s).

The two approaches are not mutually exclusive and therefore, students are encouraged to use either or both approaches to strengthen quality of their report.

A weekly progress report on the term project activities is submitted reflecting a) accomplishments-to-date, b) items of concern (bottleneck, obstacles, difficult-to-resolve situations, unanticipated surprises, etc.), c) next weeks goals and expectations, d) any suggestions, recommendations, help needed, etc. The project is required to be complete in every way so that it serves as a useful tool for both the decision maker and the programmer. Students are required to specify all inputs, outputs, files, system and process flow charts, logic diagrams, data flow diagrams, program specs., time and cost analysis, benefit analysis, and any narrative needed for complete understanding and explanation of their recommended design.

Some students include a procedure plan that clearly shows the sequence of operations and the programs needed in the solution of the problem. Contingent upon the nature of the problem, a technical and/or user's manual is also prepared.

Students submit their final reports adhering to proper style, format, and neatness. While representatives from participating firms are invited at the time of presentation, only a few actually make the trip. Nevertheless, copies of the report are sent to them for their review and comments. Students are graded based on the content, completeness, clarity, and style of their written and oral presentations.

In the past, participating companies have ranged from small retail outfits engaged in bicycles and accessories, video equipment, pet supplies, etc. to medium and large firms such as Levitz Furniture, Cohen's Optical, Allstate, Avis, Pan Am, etc. Topics of the project have been just as varied as the size and type of participating companies. For example, inventory management system, material surplus system, communications network for credit card authorization, job order control and invoicing systems, automating assessment department of a municipality, etc.

PROJECT PAY-OFFS

Exposure to the real world increases students' general understanding in all areas. It yields benefits which are crucial to their future careers. More specifically, students gain in the following areas:

- effective oral and written communication
- ability to win friends and influence people
- adaptability to organization and its hierarchy
- goal orientation and interdepedence
- analytical ability, creativity, and pragmatism
- diplomacy and resourcefulness
- receptivity to ideas
- flexibility, cooperation, responsibility

The tangible payoffs to students are various—

1. Some firms send letters applauding their efforts in bringing their budding enthusiasm and freshly acquired knowledge to bear on the firms' problems.
2. The university's constructive visibility in the community is a by-product which acts as a

continuous linkage between the two.

3. Some companies offer students consulting assignments and serve as their references when necessary.
4. The job market opens up for new graduates who maintain contacts with participating firms.
5. Students use their final report as a passport to prospective employers.

The biggest pay-off to me is the realization that my students have used their keenly developed analytical faculties in many facets of their lives, personal and professional.

CONCLUSION

The philosophy embodied in this course is the specific version of a broader one which underlies business education of a high caliber. The objective is to create leadership qualities as applied in the complex milieu of a professional business person. Imparting of the common body of knowledge in the field of information systems will equip the student with the needed skills of communication and problem solving under uncertain conditions. These will be further strengthened by giving him/her a wider perspective. The fast growing nature of knowledge in the field makes this an exciting venture, both as to the depth and breadth of its impact on molding a student not only to succeed materially, but also to pursue service to society as an integral part of the educational process.

BIBLIOGRAPHY

Beccue, Barbara, and Chrisman, Carol, "Balancing practice and Theory in Teaching Data Base: A Project Approach, "Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 103-105.

Deutsch, Ilene Siegel, "Information Systems Training in a Living Lab: A Model for Developing Student Generated 'Living' Case Studies in a Systems Analysis Curriculum," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 121-122.

Green, Lavon, "A Phased Approach to the Case Study in Systems Analysis and Design courses," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 109-112.

Khosrowpour, Mehdi, "Assessing the Current MIS Curriculum," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 35-39.

Leifer, Richard, and DeHaemer, Mihael J., "Behaviorally Based Teaching Methodologies for CIS courses," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 73-76.

Solano, Judith L., Leitner, Jak E., and Bloom, Kathleen C., "Computer Information Systems Development: An Interdisciplinary Approach to Design and Implementation," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 77-79.

Summer, Mary, and Sitek, Jerry, "Structured Methods and Their Practical Use: Implications for Educators," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 85-87.

Wojtkowski, Wita, Brender, Susan, and Wojtkowski, Gregory W., "Methodologies for Teaching CIS Courses," Proceedings of the Fifth Annual Information Systems Education Conference, Atlanta, 1986, pp. 81-84.

Structured On-Line Programming

Andrew M. Suhy

Associate Professor
Department of Computer Information Systems
School of Business
Ferris State College
Big Rapids, Michigan 49307

Abstract

Structured programming has been shown to result in substantial productivity gains in batch programming. With the shift toward on-line programming, particularly CICS, structured programming in on-line systems has been neglected. Since CICS has become the overwhelmingly dominant on-line environment, this paper proposes a structured programming model that can be (and has been) applied successfully to large CICS systems.

Introduction

While the concept of structured programming for the batch environment has finally been adopted many years after the original ideas were introduced, millions of lines of COBOL code have been written in unstructured spaghetti code. Indeed, one of the most common complaints about existing programs is that they are extremely difficult to maintain because they are unstructured.

Ironically, as new batch programming is becoming less common,the newer batch programs are being written in structured code. The damage,nevertheless, has been done. Indeed, largely because of this, many companies now resort to expensive 4GL's and other packages to try to bypass existing code.

Unfortunately, a trend backwards toward spaghetti code is now being revived with the increasing dominance of on-line programming. While the importance of structured coding has been almost universally acknowledged for batch programming, the importance of structured on-line programming has largely been overlooked. In reviewing the literature available on CICS, we found that most books and manuals on the subject reverted to unstructured coding that was saturated with GO TO statements.

It appears that as technology progresses, the lessons of the past are forgotten.

Two Major Rules in Structured Batch Programming

While programming in CICS is different from batch programming in both syntax and logic, there are elements from batch structured programming that can be applied to structured CICS code. In traditional batch programming, two major concepts are prominent:

(1) GO TO's should be avoided.
(2) There should be only one exit point per program.

These concepts are highly useful in batch because excessive GO TO's produce confusing branching, and the one exit point per program precept is valid because batch programs start execution with the first instruction, process data, and then end.

Major Differences Between Batch Logic and CICS Logic

CICS, however, is run in a pseudoconversational mode. This means the program will initiate execution and stop many times in many places before a single session is completed. Every time a message or response needs to be sent to the screen, the program ends. This is done because under CICS there is only one copy of any given program in memory at any given time regardless of how many users are accessing that application.

Thus, if the program did not end for a user while it was waiting for him/her to respond, then all the other users on the system would have to wait, too. As a result, unlike in batch, a CICS program must physically exit many times while it is processing a single record. This is sharply different from batch where the program will exit only once regardless of the number of records processed.

Thus, a CICS program can either exit immediately (the CICS command is RETURN) after a screen is sent, or a batch logic approach might be attempted whereby a single RETURN statement is used but it is reached by using many GO TO's. It is more logical, as well as easier, to execute the RETURN's in the places where the program physically exits rather than attempt to maintain a single exit point as is done in batch programming and to reach that point through multiple GO TO's.

The Physical Need for Multiple RETURN's in CICS

It is generally futile to use GO TO's to avoid having more than one exit per program because most CICS programs need to have more than one type of RETURN statement. This is because unlike a STOP RUN in batch COBOL, there are several different ways a program can be ended in CICS. One way to end a CICS program is to pass data back to itself when a user enters data at a terminal. This type of RETURN would allow the program to re-start execution where it left off before it was interrupted.

Another way to end a CICS program is to pass data to another

program and then end execution. Yet another way to end a CICS program is simply to return to the operating system. Generally, all of these types of exits are found in a single program, thus requiring multiple RETURN statements regardless of how many GO TO's are used.

The other times when GO TO's are usually inserted unnecessarily into CICS programs are to counteract the GO TO DEPENDING ON code that is generated automatically by CICS whenever a HANDLE AID or HANDLE CONDITION command is invoked.

But here, too, GO TO coding by the programmer is not needed. This is because after the condition is handled by the program logic, a map is usually sent out and the SEND MAP command is followed by a RETURN command. This has the effect of not only ending the program immediately, but also returning control to the beginning of the program. Thus, it is preferable to code a SEND MAP/ RETURN sequence rather than to try to use a GO TO to another paragraph that contains the RETURN statement. The following diagram will contrast the unstructured approach with a CICS structured approach using pseudocode.

The Unstructured Approach

```
Start Program
If First Execution
   Send map
   GO TO RETURN-TYPE-1-PARAGRAPH
Endif

Receive map
Condition-1
   Move data-1 to map
   GO TO SEND-MAP-PARAGRAPH
Condition-2
   Move data-2 to map
   GO TO SEND-MAP-PARAGRAPH
Condition-3
   Move data-3 to map
   GO TO SEND-MAP-PARAGRAPH

SEND-MAP-PARAGRAPH
   Send map
   GO TO RETURN-TYPE-1-PARAGRAPH

At End
   If user selects AID key 1
       GO TO RETURN-TYPE-1-PARAGRAPH
   ELSE IF user selects AID key 2
```

```
       GO TO RETURN-TYPE-2-PARAGRAPH
ELSE IF user selects AID key 3
       GO TO RETURN-TYPE-3-PARAGRAPH
ELSE GO TO RETURN-TYPE-1-PARAGRAPH
ENDIF

RETURN-TYPE-1-PARAGRAPH
   RETURN (type 1)
RETURN-TYPE-2-PARAGRAPH
   RETURN (type 2)
RETURN-TYPE-3-PARAGRAPH
   RETURN (type 3)
```

RETURN (type 1) would represent an exit with same program restart.

RETURN (type 2) would represent an exit with transfer of control to another program.

RETURN (type 3) would represent an exit to the operating system.

A Structured Approach

```
Start Program

If First Execution
   Send map
   RETURN (type 1)
ENDIF

Receive map

Condition-1
   Move data-1 to map
   Send map
   RETURN (type 1)

Condition-2
   Move data-2 to map
   Send map
   RETURN (type 1)

Condition-3
   Move data-3 to map
   Send map
   RETURN (type 1)
```

```
At End

   IF user selects AID key 1
       RETURN (type 1)
   ELSE IF user selects AID key 2
           RETURN (type 2)
   ELSE IF user selects AID key 3
           RETURN (type 3)
   ELSE RETURN (type 1)

   ENDIF
```

Coding the RETURN statements exactly where the program must exit has several advantages:

(1) It is easier to follow the logic if a RETURN statement is coded when the program will physically exit.rather than using a GO TO to branch to a paragraph that contains a RETURN.
(2) As seen in batch programming, programs that avoid GO TO's are generally easier to maintain than programs that use many GO TO's.
(3) By using the RETURN command in the places where you wish to exit rather than using a GO TO to a paragraph that contains a RETURN, the program is easier to debug because CEDF--the on-line debugging equivalent of READY TRACE will display each of the RETURN statements but it will not display GO TO's because GO TO's are not CICS instructions.
(4) Editing logic is vastly simplified because every time a field fails an edit check, the program ends after sending out a new map with an appropriate error message. This structure will also catch any changes the user may have made to other fields on the map that had previously passed the edit checks. Without the immediate RETURN's editing logic becomes dramatically more complicated. The alternative is to develop very intricate sequences of switches, pointers, and counters to keep track of error fields, correct but changed fields, correct but unchanged fields and so on.

If, however, there is a RETURN statement for every failed edit check, then control automatically passes to the beginning of the program and all of the data fields on the screen are re-edited when the user presses the enter key (or any other AID key).

Generally, there are only a few data fields on the screen and by re-editing every field whenever there is an error on the screen, the need to determine which fields are changed and incorrect, changed but correct, changed but incorrect in a new way, is eliminated. The insertion of RETURN statements in the appropriate locations simplifies editing logic enormously.

(5) Locality of Reference--important for reducing processing delays, is greatly improved. In on-line systems it is important to prevent excessive branching

to distant modules (also known as locality of reference) because this tends to slow down response time substantially in time-sensitive applications. By eliminating GO TO's, branching is reduced quite sharply.

Summary

In CICS it is generally not possible to eliminate multiple RETURN's because unlike in batch, there are many different types of exits and several different types are needed in virtually every program. However, it is possible and desirable to reduce and even eliminate GO TO's in structured CICS programming.

While structured code has largely replaced unstructured code in new batch systems, it is critical that a structured approach should be used in developing CICS programs before millions of lines of unmaintainable code are generated.

Selected References

Customer Information Control Systems/ Virtual Storage (CICS/VS) Version 1 release 5 Application Programmer's Reference Manual Command Level. White Plains: IBM Data Processing Division. 1981.

Jatich, Alida. CICS Command Level Programming. New York: John Wiley and Sons Inc. 1985.

Kacmar, Charles, J. On-Line Systems Design and Implementation Using COBOL and Command Level CICS. Reston: Reston Publishing Co. 1984.

Lim, Pacifico Amarga. CICS/VS Command Level with ANS COBOL Examples. Van Nostrand Reinhold Co. 1982.

Lowe, Doug. CICS for the COBOL Programmer. Fresno: Mike Murach and Associates. 1984.

A TOP-DOWN APPROACH TO THE CONTROL BREAK ALGORITHM

by Wayne C. Summers, Hilton Chen and Anthony J. Duben
Department of Computer Science
Southeast Missouri State University
Cape Girardeau, MO 63701

One of the more important algorithms in Business Applications programming is the algorithm for generating a report with control-break logic. Not only is this a significant algorithm to discuss in an introductory business applications programming course but it is also a fairly simple algorithm that can be used to illustrate top-down design and stepwise refinement. One of the difficulties that we have observed from teaching this algorithm in our introductory programming courses in COBOL and PL/I , is that there are just about as many variations of this algorithm as there are textbooks and that many of these algorithms are relatively complicated in their ways of handling special cases. This paper is an attempt to address this issue and to discuss one version of the control break algorithm that we have found to be extremely successful in teaching COBOL and PL/I.

Background. A control break is essentially a pause in the normal flow of output to print out totals or perform a summary when the end of a set or subset of data is encountered. For example, the printing of totals for a set of data when that data has been processed can be considered a control break and is detected when the "end of file" is read. Other control breaks depend upon the use of control fields or key fields which are generally not unique among the records and which are used to sort the records. An example of output generated by a control break algorithm where the student major is the control field is illustrated in Fig. 1.

One-level control break algorithm. The major difficulty in the control break algorithm is identifying when to perform a control break and then continuing on once the control break has been printed. This is usually accomplished by using a controlling loop to print the individual detail lines and then within that loop comparing the current control field with its previous value. When these control fields are different, a control break is performed. Because the algorithm must compare two consecutive control fields, it is necessary that the data records be sorted on this control field. All the control break algorithms are comprised of three components: preparing for the control break, processing records, and processing the control break.

There are two unique problems that arise in the processing of records using the control break algorithms. The first is to what to initialize the previous

control field, before comparing it with the control field of the first record. If the value of the previous control field is anything but the value of the first record's control field, then the result is a false control break. Many of the algorithms in textbooks go out of their way to insure that this does not happen. The other special case that needs to be considered is at the end-of-file. Even though the last record has been printed, there are still control breaks that need to be printed, so once again many of the textbook algorithms handle this as a special case. These two cases can result in extra and oftentimes confusing coding if they are not handled well.

The algorithm that we use at our university(Fig 2) was first discovered by us in Grauer[2] and is now used in the most recent edition of Welburn[8]. It addresses these two problems without any special handling of first and last cases. It is also very adaptable and expandable to multi-level control breaks and most importantly is easy to understand. It is also very easy to implement this algorithm in PL/I and COBOL using DO-WHILE and PERFORM-UNTIL loops. Most of the other algorithms found in [1,3,4,5,6,7] (Fig. 4) other textbooks involve setting the previous control field to the current field twice, once outside the loop and once after the control break has been processed. This can be seen in Fig. 4 where lines 1. and 3. are repeated again as lines 4a iii. and 4a iv. These other algorithms also involve processing the control break twice, once in the loop and then again after the loop. This is observed occurring in Fig. 4 where line 4a i is repeated again in line 5. Although each of the algorithms found in the texts [1,3,4,5,6,7] contains its own variation and implementation of the control break algorithm, each essentially suffers from the same deficiencies found in Fig. 4. They all require the use of duplicate code to assign values to the testing key field and to initialize the accumulators for the control break and they all require the double coding of the call to the module to print out the control breaks.

Multi-level control breaks algorithm. The differences between these two algorithms(Fig. 2 and Fig. 4) becomes even more apparent as we move to multi-level control breaks.(Fig. 3) A multi-level control break consists of two or more control fields. Each control field must be compared with the previous value of the control field and when it changes, a control break occurs. The result is one or more control break summary lines. When you attempt to extend control breaks to more than two levels the logic becomes quite difficult for students to comprehend. As you can see in Fig. 5 and 6 even with just two levels of control breaks, the amount of coding that becomes necessary in the standard algorithm(Fig. 6) has become excessive. The algorithm in Fig. 6 requires the initialization of the accumulators and test fields for each level of control break to be performed twice and also requires the code to format and print each control break summary line to be written twice. This proves to be especially confusing to the students. The best feature of the SEMO algorithm is that to extend to a two or more level control break from a one level control break, simply requires the addition of a similar block of code for each level and an extra condition on the DO-WHILE. Since the control break algorithm requires that the entire data file be processed, while processing the major control group data set within the entire data file, while also processing any intermediate and minor control group data sets within each major control group data set, each additional level of control breaks requires a stepwise refinement. This is an excellent example of top-down design and structured programming. The block of code for each level will always consists of the three modules to (i) initialize the variable fields for that level of the control breaks, (ii) another DO-WHILE or PERFORM-UNTIL and (iii) the module to print out that level

of the control breaks. These features are even more obvious in the three level control break algorithms which really emphasizes the difference between using structured programming principles with top-down design and stepwise refinement using the SEMO algorithm in contrast to the less structured approach which now requires three levels of duplication.

Conclusion. The result of using the SEMO algorithms rather than the standard algorithms is that the students understand control breaks much better. These algorithms are also especially valuable when emphasizing top-down design and stepwise refinement. This is particularly true when modifying a 1-level control break program to handle multi-levels of control breaks.

MAJOR	NAME	HRS
CS	Mr. Computer	18
CS	Ms. Printer	16
CS	Ms. Keyboard	17
hours completed in CS		51*
MA	Mr. Chalk	14
MA	Ms. Eraser	16
hours completed in MA		30*
Total hours completed		81**

Fig. 1 1-level control break(output)

1. Initialize report variable fields
2. Do the priming read
3. Do While not end of file
   a. Initialize control break accumulators and test fields
   b. Do While current control field = previous control field and not end of file
      i. Print the detail lines
      ii. Do any arithmetic
      iii. Increment the accumulators
      iv. read the next record
   c. Print the control break line
      i. add accumulators to report totals
4. Print the report totals

Fig. 2 1-level control break algorithm-SEMO version

Bibliography

1. Abel, P. Structured Programming in PL/I and PL/C, Reston Publishing, 1985.
2. Grauer, R. and Crawford, M. The COBOL Environment, Prentice Hall, 1979.
3. Horn, L.W. and Gleason, G.M. Beginning Structured COBOL, Boyd & Fraser Publishing, 1983.
4. Philippakis, A.S. and Kazmier, L.J. Structured COBOL,3rd. Ed., McGraw-Hill, 1986.
5. Popkin, Gary S. Comprehensive Structured COBOL,2nd. Ed., Kent, 1986.
6. Stern, N. and Stern, R.A. Structured COBOL Programming4th. Ed., John Wiley & Sons, 1985.
7. Teglovic Jr., S. and Douglas, K.D. ANSI Structured COBOL, Irwin, 1986.
8. Welburn, T. Structured COBOL,2nd. Ed., Mayfield Publishing, 1986.

COLLEGE	MAJOR	NAME	HRS
BUSINESS	CS	Mr. Computer	18
BUSINESS	CS	Ms. Printer	16
BUSINESS	CS	Ms. Keyboard	17
hours completed in CS			51*
BUSINESS	MA	Mr. Chalk	14
hours completed in MA			14*
hours completed in BUSINESS			65**
SCIENCES	CH	Mr. Testtube	18
hours completed in CH			18*
SCIENCES	PH	Mr. Moments	18
SCIENCES	PH	Ms. Inertia	15
hours completed in PH			33*
hours completed in SCIENCES			77**
total hours completed			158***

Fig. 3 2-level control breaks(output)

```
1. Initialize report variable fields
    and control break accumulators
2. Do the priming read
3. Initialize the control test field
4. Do While not end of file
  a. If current control field is not
       = to previous control field then
    i. Print the control break line
   ii. Do any arithmetic
  iii. Add accumulators to report totals
   iv. Reinitialize accumulators
    v. Reinitialize control test field
  b. Print the detail lines
  c. increment the accumulators
  d. read the next record
5. Print the control break line
6. Print the report totals
```

Fig. 4 1-level control break algorithm standard version

```
1.Initialize report variable  fields
2.Do the priming read
3.Do While not end of file
  a.Initialize accumulators and test
     fields for major control break
  b.Do While major control field =
     previous major control field and
     not end of file
    i.Initialize accumulators and test
      fields for minor control break
   ii.Do While minor control field =
      previous minor control field and
      major control field = previous
      major control field and not end of
      file
       a.  Print the detail lines
       b.  Do any arithmetic
       c.  Increment minor accumulators
       d.  read the next record
  iii.  Print minor control break line
       a.  add minor accumulators to
           major accumulators
  c.  Print major control break line
    i. Add major accumulators to report
         totals
4. Print the report totals
```

Fig. 5 2-level control break algorithm SEMO version

```
1.Initialize variable fields for report
  and the major and minor accumulators
2.Do the priming read
3.Initialize major and minor control
   test fields
4.Do While not end of file
  a.If current major control field not
    = previous major control field then
    i. Print minor control break line
      a.Add the minor accumulators
         to the major accumulators
      b.Reinitialize minor accumulators
      c.Reinitialize minor control test
           field
   ii. Print major control break line
      a.Add major accumulators
        to the report totals
      b.Reinitialize major accumulators
      c.Reinitialize major control test
           field
  b.Else If current minor control field
    not = previous minor control field
    i. Print minor control break line
      a.Add minor accumulators
        to the major accumulators
      b.Reinitialize minor accumulators
      c.Reinitialize minor control test
           field
  c. Print the detail lines
    i. Do any arithmetic
   ii. increment the minor accumulators
  iii. read the next record
5.Print the minor control break line
6.Print the major control break line
7.Print the report totals
```

Fig. 6 2-level control break algorithm standard version

AN EMPIRICAL INVESTIGATION OF THE COMPLEXITY OF STRUCTURED PROGRAMMING TECHNIQUES

R. Wayne Headrick & George C. Fowler
Department of Business Analysis and Research
Texas A&M University
College Station, Texas

There is typically more than one way to solve a programming logic problem with "good" structured programming techniques. Because those techniques would not be expected to be equally complex, it is important from both teaching and operational systems development perspectives that knowledge regarding the relative complexity of the techniques be gained. This paper describes a study in which various techniques for implementing control break logic were compared by a group of novice programmers. The results indicate that the novice programmers were able to discern differences in the relative levels of complexity of the techniques under investigation.

INTRODUCTION

The wide acceptance and high popularity of structured programming techniques is well recognized today. However, initial acceptance of the concept of structured programming came rather slowly. It was five years between the time its theoretical basis was presented by Bohm and Jacopini [2] and IBM first made significant use of it. Since that time, the level of acceptance has increased to the point that the question is no longer whether structured programming techniques should be used, but rather which implementations of those techniques should be used. Most programming professionals have long recognized that many programming problems can be solved using more than one valid structured programming technique. This recognition, and the desire to identify the technique(s) that result in easily written, understood and maintained code, has provided the impetus for a significant volume of research related to the measurement of the complexity of code written to provide solutions to a variety of programming problems [3,4,5,8,11].

Despite the ever-increasing volume of literature devoted to software complexity measurement, no real concensus has been reached regarding acceptability of the measures developed thus far [7]. One major reason for this is that the measurement techniques have not been developed with any particular use in mind [1,7]. This research effort is motivated by the desire to gain knowledge regarding the complexity of various structured programming techniques for the specific purpose of improving the process of teaching those techniques in the classroom.

This paper focuses on one part of our continuing research program relating to structured logic complexity issues. Specifically, it describes the results

of our efforts to evaluate the relative levels of complexity of the various "good" structured programming solutions to a common programming logic problem as perceived by a group of novice programmers (i.e., students in their first COBOL programming class). The problem selected for investigation was the two-level control break problem typically included in the first COBOL programming course.

METHODOLOGY

A survey of the COBOL texts specifically designed for use in the first COBOL programming course identified three basic techniques for accomplishing two-level control break logic. These included the un-nested IF-tests [9], the nested IF-tests [6] and the nested PERFORMs [12], with the nested IF-test logic far outnumbering the other two techniques. (Only one major reference has been cited for each of the techniques.) Since most texts used nested logic, it was determined that two additional techniques, the IF-test with a nested PERFORM and the PERFORM with a nested IF-test, were legitimate two-level control break techniques and should be included in the investigation. In this study, the logic that makes use of the un-nested IF-tests is referred to as technique A; nested IF-tests, technique B; nested PERFORMs, technique C; PERFORM with nested IF-test, technique D; and IF-test with nested PERFORM, technique E.

As the basis for subsequent statistical testing, it was hypothesized that there would be no significant differences among the perceived complexity levels of the "good" structured code implementations of the programming techniques presented to solve the selected problem. To test this hypothesis, a fixed effects model analysis of variance procedure was used. The treatments were pair-wise comparisons of the five structured solutions to the control break logic problem presented above, and the observations were perceived levels of relative complexity of the techniques being evaluated. Duncan's multiple range test was employed to test the hypothesis that the treatment means were not significantly different.

The instrument used in this phase of the study consisted of two tasks that related to one pair of well-written (i.e., written using logical indentation, descriptive procedure and data names, etc.) COBOL-85 programs that could easily pass as structured English pseudo-code. The specific tasks the students were required to perform were to:

(1) Use some program documentation technique to show the logic of the main parts of the two programs. This task was designed to ensure an adequate level of comprehension of the programming techniques being used.

(2) Indicate which one of five statements most accurately reflects their perceptions of the relative complexity of the two subject techniques.

A pilot study was used in an effort to validate the instrument. Analysis of the results of this preliminary sample using Duncan's method revealed that the pair-wise comparisons involving technique E had treatment means that were significantly different (in the direction indicating greater complexity) from those of the other comparisons ($\alpha<.05$). Because this result confirmed the authors' earlier hesitation to even include that particular technique in the study, it was eliminated from further consideration.

The revised instrument was administered to 84 students, which provided 14 observations for each of the six treatments. It was administered at a point in the semester where the students had an adequate knowledge of the COBOL language to ensure they could readily

understand the code being presented, but before the subject of control breaks had been covered in the lecture.

RESULTS

Applying Duncan's multiple range test to the data generated by the second administration of the instrument, it was possible to state that the treatments that compared techniques C to D and A to D were significantly different ($\alpha<.05$). The major contribution that this result provides is highlighting identification of the most complex and least complex techniques included in the study, techniques C and A, respectively.

Viewed in terms of the relative levels of complexity as indicated by the signs and magnitudes of the treatment means, regardless of statistical significance, we found that there was consistency in the results of this phase of the experiment (i.e., no contradictory comparisons were found). Specifically, the pattern exhibited was, technique A is less complex than technique B, which is equal in complexity to technique D, which is less complex than technique C.

DISCUSSION

This experiment has shown that novice programmers, defined here as those programming students that have a working knowledge of the COBOL language but have not yet learned about the logic associated with the programming techniques under consideration, perceive the un-nested IF-test implementation of control break logic to be less complex than the other implementations. It has also shown that the nested PERFORM implementation was perceived as being the most complex, with the nested IF-test and PERFORM with nested IF-test falling somewhere between them.

Prior to the experiment, we surveyed a sample of 18 Introductory or Comprehensive COBOL textbooks to determine which techniques for implementing multi-level control break logic were being taught. We found that a majority (10 of the 18) were using the nested IF-test technique, with the remaining 8 using either the un-nested IF-test or the nested PERFORM techniques (5 and 3, respectively). Assuming that students are being taught to use the techniques presented in their texts, we must conclude that most are being taught to use the nested IF-test technique. It should be noted that we both teach control break logic using the nested PERFORM technique because, prior to undertaking this study, we believed that it was the most logical, straightforward and easily understood of the commonly used techniques.

What do the results of this experiment lead us to conclude about the techniques that many of us are using to teach control break logic? On one hand, we could conclude that we should be teaching the un-nested IF-test technique, and that we are simply making it harder than it needs to be when we teach one of the more complex techniques. On the other hand, we could also conclude that we really are right, and that although the nested techiques are somewhat more complex, they are more efficient than the un-nested IF-test technique and should be taught anyway. (This is, of course, related to the "no pain, no gain" theory.) Finally, we could conclude that further study needs to be accomplished.

As we are not yet ready to either simply ignore the results of this phase of the study or accept them and begin teaching un-nested IF-test control break logic, our conclusion is that further study needs to be accomplished. Previous research has shown that the method used to present logic can either facilitate or inhibit its use [10]. To investigate the effect that presentation method has on the novice students' perceptions of the relative

complexity levels of the techniques under consideration, additional studies are being conducted. In addition, the perceptions of experienced programmers are being studied to provide other bases for evaluating the complexity levels of the various techniques.

REFERENCES

1. Basili, VR and Hutchens, DH, "An empirical study of a syntactic complexity family," IEEE Transactions on Software Engineering, SE-9(6):664-672, November 1983.

2. Bohm, C and Jacopini, G, "Flow diagrams, turing machines and languages with only two formation rules," originally presented at the International Colloquium on Algebraic Linguistics and Automata Theory in Jerusalem, Israel in 1964, and printed in Communications of the ACM, 9(5):366-371, May 1966.

3. Gordon, RD, "Measuring improvements in program clarity," IEEE Transactions on Software Engineering, SE-5(2):79-90, March 1979.

4. Gordon, RD, "A qualitative justification for a measure of program clarity", IEEE Transactions on Software Engineering, SE-5(2):121-128, March 1979.

5. Halstead, MH, Elements of Software Science, Elsevier North-Holland, New York, 1977.

6. Horn, LW and Gleason, GM, Comprehensive Structured COBOL, Boyd & Fraser Publishing Company, San Francisco, 1986.

7. Kearney, JK, Sedlmeyer, RL, et. al., "Software complexity measurement," Communications of the ACM, 29(11):1044-1050, November 1986.

8. McCabe, TJ, "A complexity measure," IEEE Transactions on Software Engineering, SE-2(4):308-320, December 1976.

9. Teglovic, S and Douglas, KD, ANSI Structured COBOL: An Introduction, Richard D. Irwin, Inc., Homewood, IL, 1986.

10. Vessey, I and Weber, R, "Structured tools and conditional logic: An empirical investigation," Communications of the ACM, 29(1):48-57, January 1986.

11. Weissman, LM, "Psychological complexity of computer programs: An experimental methodology," ACM SIGPLAN Notices, 9(6):25-36, June 1974.

12. Welburn, T, Structured COBOL: Fundamentals and Style, Mayfield Publishing Company, Palo Alto, CA, 1986.

TEACHING ARTIFICIAL INTELLIGENCE IN THE BUSINESS SCHOOL

Richard G. Vedder, Assistant Professor of Information Systems
BCIS Department, North Texas State University, Denton, Texas 76203

ABSTRACT

The long-awaited entry of computer products incor- porating artificial intelligence (AI) technology means a growing market in business and industry for persons with AI training. Business schools need to revise their curricula accordingly. This paper examines some of the relevant issues and offers a plan of action for meeting this educational challenge.

INTRODUCTION

One does not have to read far in the literature these days to realize that artificial intelligence research is at long last introducing significant application products into the marketplace. What does the advent of this long-awaited technology mean for education, specifically for the mission of schools and colleges of business? Which subjects need to be taught? How might they be worked into the curricula? What software and hardware resources are needed? Answering these and other questions is the purpose of this article.

Before proceeding, we must note some limitations. In this paper, the phrase "artificial intelligence" and the acronym "AI" are meant in a restrictive fashion, referring specifically to knowledge-based systems like natural language understanding programs and expert systems. It is the author's view that other subdivisions of AI (such as robotics and cognitive learning systems) are more appropriately the domain of computer science, cognitive psychology, and engineering (although their impacts on organizational activity and society in general are worthy topics for discussion in business school curricula). Furthermore, issues connected with promoting AI research will be addressed only as they relate to teaching AI in the business school environment.

The commercialization of AI technology poses a major challenge to business schools. The marketplace certainly expects AI to be profitable. For instance, DM Data, a Scottsdale Arizona market research firm, estimates a total market for AI products of $4.2.

For instance, DM Data, a Scottsdale Arizona market research firm, estimates a total market for AI products of $4.2 billion in 1990, up from this year's estimated $1 billion [1]. On the other hand, great profits frequently are accompanied by great risks, and the so-called "high technologies" have had their share of expensive failures (for example, RCA's $500 million loss with videodisc players [2]). Business school graduates need to acquire sufficient technical background concerning AI technology in order to fully appreciate both the potential risks and benefits of AI applications.

Complex technologies require trained specialists. Unfortunately (and not unexpectedly) when a new technology such as AI comes into the marketplace, trained personnel are very scarce and in high demand. This problem forces organizations wanting to stay competitive with the new technology to spend large sums of money on re-educating part of their current

workforce. The author personally knows of corporations which are spending between $100,000 and $200,000 per person to train knowledge engineers in-house. This crisis poses an educational (and financial) opportunity for business schools, training both students and professionals. In addition, meeting the acute shortage of trained knowledge engineers is not just a challenge for information systems departments; AI applications will span the entire range of corporate interests, and thus all business school departments will share in the educational responsibility.

It is important to realize that teaching AI in a business school does not have to be an expensive proposition. Large and/or well-endowed business schools need not monopolize the educational market in this field. Small schools and colleges of business can offer quality AI programs for two reasons. First, when any new technology appears every prospective user must begin at the same starting line, with no one user necessarily any better off than another. Each business school, given the shortage of AI experts, must find professors interested in the new discipline and encourage their retraining. Each school must acquire software and perhaps hardware resources. Each school must decide where to insert the teaching of AI in the curricula. And so it continues. These problems are certainly not easy to solve; nevertheless, all business schools must deal with them.

The second reason why every business college or school can integrate AI into their course offerings is because it has never been cheaper or technologically easier to do so. Just a few years ago, no department could hope to teach AI without a very considerable expenditure of dollars for expensive AI software and hardware (whether specialized LISP-type machines or more general computing systems, such as VAXes). Today one can purchase workbench tools for less than $3000 (and many for under $1000) which operate on microcomputers and make it easy to learn many techniques of knowledge engineering. Moreover, and again in opposition to the recent past, there is now a good variety of textbooks on the subject.

TEACHING MISSION

Assuming a decision to incorporate AI into the business curricula, what goals or aims should be pursued? Minimally, the information systems group needs to offer a course which introduces students to the rudiments of knowledge engineering and examines some of the managerial implications of bringing AI products into the organizational environment. Additionally, expert systems must be included in any study of computerized decision support tools or intelligent design/coding workbench environments. Both database management and office automation courses need to introduce natural language programs (such as Microrim's Clout™). Interested students should also be encouraged to take specialized AI courses (in, for example, LISP or Prolog) from other departments. Eventually the information systems group should offer an additional, more intensive course on building AI applications.

Beyond this level of commitment, other departments in the business school must discuss AI in any course using computerized tools having an AI capability. One software package of special interest here is Micro Data Base System's Guru™, an integrated application package incorporating a knowledge engineering environment for developing specialized expert systems which can utilize and interpret information supplied by other parts of the package (such as the 123-patterned spreadsheet). Students need not build their own expert systems in order to take advantage of this part of Guru. Appropriate expert system applications

can be supplied by professors or graduate students, as is commonly done now with more conventional software. Moreover, Guru's data base component uses a natural language interface, and thus exposes students to this aspect of AI technology.

THE PROBLEM OF FACULTY

The acute shortage of trained AI personnel is not just a business or industry problem. With easily more than 30 job opportunities available for every graduating Ph.D. specialist in AI, the personnel crisis definitely involves the educational community. Discussion here will not center on ways to attract these new Ph.D's. The considerable discrepancy in salary levels that exists between industry and education alone makes recruitment a problem far beyond the scope of this article. Instead a few suggestions will be made for encouraging present faculty to develop AI as a new specialty.

Support for faculty re-education centers on two issues. First, there must be a clear, coordinated, and articulated policy by deans and department chairs to make AI an important part of each business student's education. Second, this commitment must be backed by money. Faculty interested in learning AI need their travel expenses covered for AI symposia or conferences, such as the National Conference on Artificial Intelligence (scheduled for Seattle, WA in 1987). Library funds must be allocated for AI books and serials (exs., Artificial Intelligence, AI Magazine, IEEE Transactions on Systems, Man, and Cybernetics, Expert Systems, Cognitive Science). Business schools must also provide hardware and software support for teaching and research in AI. As will be explained later, this is not nearly as expensive an activity as it would have been even a few years ago. The bottom line is that while schools need not spend the colossal sums business and industry have spent (and are continuing to spend), there must be a meaningful level of organizational and especially financial backing if current business faculty are to learn and teach AI.

THE INFORMATION SYSTEMS COURSE

At this point we turn our attention to consider which AI topics to cover in the information systems introductory course. Discussion will assume that enrolled students have background in systems analysis and design concepts, and that they will use an expert system development program operating on a XT or AT level of microcomputer.

The introductory AI course offered by the information systems group should have two components, a technology-oriented part and a managerially-oriented part. The first part of the course must supply enough material so that students can build small knowledge (expert) systems. Subjects that need addressing include:

(1) an overview of AI history and directions;
(2) an overview of expert systems, including application-based and skill-based topologies and construction of expert systems;
(3) knowledge acquisition methodologies;
(4) knowledge representation schemes;
(5) knowledge utilization or processing;
(6) a sampling of tools for constructing expert systems; and (time permitting)
(7) an introduction to natural language systems, programming apprentices,and other types of knowledge systems.

Early in the course, students form project teams and choose a problem area suitable for building a small-scale expert system (such as a personal tax advisor). It is not necessary to

restrict projects to only business-oriented scenarios. Non-business problems are equally, and sometimes more, instructive than ones grounded in the business world. Each team uses the course-supplied expert system applications generator (also called a "shell" program) to incorporate knowledge gained from interviewing human experts into their expert system.

Instructors should expect to spend anywhere from three to six classroom hours explaining the operation of the chosen ES generator. Depending on the shell's hardware and operating system requirements, time might have to be spent discussing these topics as well. It is important to give students "hands-on" experience with the ES generator as soon as possible. A simple, yet effective method is to have them key in a ready-to-go expert system application.

Late in the term, each team can demonstrate their project. A useful scenario is to pretend that the students are knowledge engineers working for a software development firm specializing in expert system applications. Each team gives a full dress rehearsal of their intended presentation to their client. The rest of the class, as fellow knowledge engineers, critique their work.

The second part of the course deals with some of the key issues confronting managers planning to introduce AI technology into the organization. Useful topics include:

(1) the potential value of expert system applications to an organization (ex., for decision support);
(2) the present limitations of AI technology;
(3) selecting appropriate subjects for expert system development;
(4) integrating AI technology with traditional DP and MIS functions;
(5) changes in the workforce brought about or accelerated by introducing AI-based or AI-supplemented products;
(6) legal problems connected with AI (ex., the risks of being sued for using or not using an expert system); and
(7) organizational and societal implications from using AI products.

The manner in which class time divides between the technological and managerial portions may depend on the intended audience. For instance, undergraduates may receive more technical material while graduate students may focus more on managerial concerns.

When AI concepts are integrated with other course offerings (both inside and outside the information systems group), the focus should be on using AI techniques to improve the efficiency of tasks under consideration, and on possible consequences for management and organizations at large. Technical issues of necessity cannot be fully addressed, but students should be taught the general concepts behind the AI technology under discussion. For example, if students use Guru's expert system, they should learn about knowledge bases, inference engines (i.e., knowledge processing), and knowledge acquisition. Or, if Clout is being used with Microrim's R:Base series, students should be exposed to the process whereby a natural language program parses and understands written English commands.

Among the AI subjects worthy of business course offerings for professional development are: introduction to AI, introduction to expert systems, and courses on specific AI products such as IntelliCorp's Knowledge Engineering Environment (KEE)™, Texas Instrument's Personal Consultant™ series, and Guru. The

introductory courses must discuss what can be accomplished with AI technology now, what kinds of AI applications a firm could pursue with current organizational resources, and what managerial and organizational impacts may ensue from introducing AI products.

SOFTWARE CONCERNS

Dramatic improvements in AI development software have occurred over the past few years. It is now possible to purchase AI software tools of considerable power for well under $3000, and indeed for under $1000. Moreover, these products operate on XT/AT level microcomputers and are capable of building sophisticated applications.

For development purposes, the most elemental approach involves prototyping in an AI language such as Gold Hill's Golden Common LISP, Texas Instrument's Scheme (a reduced set of Common LISP), Borland's Turbo Prolog, or Production Systems Technologies' OPS/83™, and at some stage perhaps rewriting the application for ease of portability in a conventional language like C. There is much to be said in favor of using a software tool of this type, especially on grounds of programming flexibility, and in fact many computer science departments use them in their introductory AI courses.

In the business school, however, AI programming languages should be left for students in advanced AI courses. Far greater and more immediate benefit can be realized from using knowledge engineering environments (i.e., shells), at least in the introductory course. Because they already incorporate the framework needed to build a knowledge system application, shells greatly speed the development process and free business students from having to wrestle with programming minutiae.

It is the advent of commercial shell programs that is largely responsible for the migration of AI technology from the computer labs and into the marketplace. At first the only products available (ex., IntelliCorp's KEE™) cost tens of thousands of dollars and required VAX or LISP-type machine environments. Now, however, there are a number of commercial shells which cost far less and will run on XT/AT level machines, such as EXSYS, Inc.'s EXSYS and Texas Instruments' Personal Consultant series. The power of these less expensive shells should not be taken lightly, as already they have generated significant commercial applications. And for educational purposes, the reduced flexibility afforded to the knowledge engineer is compensated by the much improved ease of use.

Finally, there are commercial software products (like Clout and Guru) that incorporate AI technology and are starting to appear in the marketplace. Unfortunately, since "AI" is currently a hot buzzword, some software publishers have tried to jump on the bandwagon by dropping the acronym into their advertising copy, irrespective of whether the extolled product really uses true AI technology. Even though the buyer must beware, the fact remains that more and more software products incorporating genuine AI technology will appear on vendor shelves. It is this type of AI software product that is best suited for incorporation into other, non-IS business curricula.

HARDWARE CONCERNS

Given the variety of AI software available, it is no longer necessary to use VAXes or LISP-type machines for teaching introductory coursework. XT or AT level microcomputers will perform quite capably. The recommended configuration is 640K main memory, a color monitor (EGA now, and VGA when the AI software will support it), and a hard disk; a mouse is still optional. AI applications tend to consume great amounts of main memory. Thus expanding

from 640K to at least 1Mb should be considered once OS/2 is released in 1988 and expert system tools are revised to take advantage of this new operating system environment. VAXes or LISP-type machines are more likely required for advanced coursework, research, or for developing large applications, but then other, more powerful and more costly software may be required also.

TEXTBOOKS

The author has found the following list useful for an introductory undergraduate course in the information systems group:

Paul Harmon and David King. Expert Systems: Artificial Intelligence in Business. Wiley Press, 1985.

Patrick H. Winston and Karen A. Prendergast, eds. The AI Business: Commercial Uses of Artificial Intelligence. MIT Press, 1984.

Edward A. Feigenbaum and Pamela McCorduck. The Fifth Generation. Signet, 1984.

All three volumes are available in paperback editions.

For an introductory course on the graduate level, the author recommends replacing the Harmon and King book with:

Donald A. Waterman. A Guide to Expert Systems. Addison Wesley, 1985.

This hardback volume contains a more comprehensive treatment of expert system building and many excellent topical bibliographies.

Additional AI books of interest include:

Aaron Barr, Paul Cohen, and Edward A. Feigenbaum. The Handbook of Artificial Intelligence, 3 vols., William Kaufmann, 1981-1982.

Holsapple, Clyde W. and Andrew B. Whinston. Business Expert Systems. Irwin, 1986 (This volume uses Guru as its demonstration AI environment.).

Liebowitz, Jay. Introduction to Expert Systems. Mitchell Publishing, 1987.

Pamela McCorduck. Machines Who Think. Freeman, 1979.

Elaine Rich. Artificial Intelligence. McGraw-Hill, 1983.

Roger C. Schank and Peter G. Childers. The Cognitive Computer. Addison Wesley, 1984.

M. Yazdani and A. Narayanan, eds. Artificial Intelligence: Human Effects. Halsted Press, 1984.

REFERENCES

1. Livingston, Dennis. "Expert systems and AI hardware reach commercial markets." High Technology (July 1986), 22.

2. Kharbanda, O. P. and E. A. Stallworthy. "RCA's Slipped Disc" in Management Disasters And How To Prevent Them. Gower Publishing Co., 1986, pp. 117-126.

TEACHING A PROJECT ORIENTED INTERDISCIPLINARY COURSE IN EXPERT SYSTEMS

Amjad Umar
School of Management
University of Michiagn-Dearborn, Dearborn, Michigan, 48128
and
Program for Research in Information Systems Engineering (PRISE)
Industrial and Operations Engineering Department
University of Michigan, Ann Arbor, Michigan, 48109-2117

ABSTRACT

The experience of teaching a senior/graduate level project oriented course in expert systems for students in engineering, management and computer science in a small university is described. Due to the diverse interests, academic preparation, and expectations of the students, several adjustments had to be made during the course. The topics covered, projects assigned, and lessons learned are discussed.

1. INTRODUCTION

In the fall of 1986, an interdisciplinary senior/graduate course in expert systems was taught at The University of Michigan-Dearborn (UM-D), one of the three campusses of The University of Michigan. The purpose of this course was to introduce the basic concepts of artificial intelligence and show the application of these concepts in building expert systems for management, engineering and scientific applications.

Courses in artificial intelligence and expert systems are becoming increasingly common. However, most of the existing courses attempt to teach either AI concepts and AI research trends, design methods and programming languages for knowledge-based systems, or quick techniques to enable "end-users" to build their own expert systems. Some end-user oriented courses for special audiences, for example engineers, have also been initiated at some universities [1]. The expert systems course at UM-D was offered under the following circumstances:

1. This course had two purposes: first, develop an understanding of the theoretical foundation and limitations of artificial intelligence, knowledge representation and reasoning; and second, through projects show how this understanding can be used to build realistic, practical and useful, expert systems.
2. UM-D is a small campus (about 7,000 students) with an interdisciplinary program in Computer and Information Systems (CIS) which draws faculty and students from engineering, management and mathematics. The course was intended for a mixture of CIS and non-CIS majors with different interests, expectations and prepartions. Due to the small number of students on the campus, it was not possible to teach different expert systems courses for different audiences.
3. Most students had interest in the practical applications of expert systems in their own areas of expertise. For example, some students were sent to this course by their employers to learn how to build specific expert systems. On the other hand, some students were interested in theoretical basis of expert systems in order to pursue further studies.
4. The backgrounds of the students were very diverse. Some students had practical experience in AI, others had already taken or were enrolled in a course in AI, while still others had no prior knowledge of AI.
5. Due to the newness of the field, adequate textbooks were not available.
6. Due to the budgetary constraints, it was not possible to purchase expensive expert shells like M.1. In addition, there was little help available in terms of teaching as-

sistants and graders because no qualified students were available.

These characteristics placed unique demands on course content, projects assigned and teaching method: the course content had to be carefully planned and tradeoffs had to be made. The approach adopted was to communicate the practical issues of expert systems through controlled projects, while focussing on the theory in the lectures. This is discussed in sections 3 and 2 respectively.

This course has been received enthusiastically by students and will be offered on a regular basis in the future. The approach adopted and lessons learned are discussed so that others with similar situations may consider the experience in developing similar courses.

2. COURSE CONTENT AND TOPICS

Three main topics were covered in this course: fundamental features and theory of expert systems, expert systems tools, and design/ implementation of expert systems. The books by Waterman [9] and Clocksin and Mellish [10] were used as textbooks.

The material covered in the texts was augmented through required readings of papers on foundation of expert systems, limitations of expert systems, examples of expert systems, knowledge representation, rule-based systems, logic programming and decision support systems. Table 1 lists the required readings. These readings provided a basis for more detailed coverage of the topics. Other books listed in references [2-7] were recommended as suggested readings.

Table 2 shows the outline of the topics discussed with reading assignments. The course material, as displayed in Table 2, was covered in three phases with each phase terminated by a project and an examination. The course material is briefly discussed here and the projects are discussed in the next section.

In phase 1, the fundamentals of expert systems were explained and supplemented with several required readings. The main emphasis in this phase was on identifying the key concepts, in particular the theoretical aspects of knowledge representation and inference. For AI concepts, the first chapter of Jackson [4] was found to give a good overview. Due to the diverse interests and backgrounds of the students, reading assignments and classroom examples covered applications of expert systems to management, computer science and engineering. Chapters 6 and 25 of Waterman [9] combined with the required readings papers 6 through 12 illustrated applications in management, engineering and computer science. For knowledge representation, the course pack articles from the IEEE Computer Special issue on Knowledge Representation (October, 1983) adequately enhanced the textbook material. The students chose a problem area of their interest and implemented an expert system by using an expert shell.

In phase 2, the tools employed in expert systems development were discussed. Several expert shells were discussed and illustrated and students were given an opportunity to work on two freeware shells. The main emphasis in this part of the course was on logic programming and Prolog. Lisp was covered briefly. The project in this phase required the students to program in Prolog and Lisp.

The last phase of this course consisted of discussing the life cycle of expert systems and applying it to building realistic expert systems. The analysis, design and implementation issues of expert systems were illustrated by requiring the students to build a knowledge based software engineering project manager (KBSEPM). The reading assignments were reduced to allow the students to work on the project. The research and development trends in expert systems were discussed at the end of the course.

3.TOOLS AND PROJECTS

Four tools were used in this course: Prolog, Lisp, and freeware shells ESIE and EXPERT. The shells were chosen because they were free and were readily available from the University of Michigan Computing Center. Lisp was covered briefly (2 hour lecture, 1 program). Prolog was chosen as the main programming language for three main reasons. First, Prolog can be used effectively for knowledge representation and has been used for several commercial expert system implementations. Second, Prolog offers a reasonable tradeoff between Lisp and

"high level" shells for classes that have students with mixed backgrounds. Third, Turbo Prolog is available at a very cost effective rate for university usage.

The projects were designed with the following objectives in mind. First, the students must be exposed to writing expert systems in Lisp, Prolog, and at least one expert shell. Second, the students should be allowed to develop an expert system in their own area of interest. Third, the students should be exposed to the task of knowledge acquisition. Fourth, the students must learn the life cycle activities of a realistically large expert system which requires integration of common data processing functions with knowledge processing and reasoning.

The first project required the students to individually build a small expert system of their choice by using one of the expert shells (ESIE or EXPERT). The focus of this project was on learning the main features of an expert system, the process of knowledge acquisition, and coding knowledge as rules. There were two main requirements: 20 to 30 rules had to be coded and an outside expert had to be consulted. Student application choices were diverse: database design, software design, network diagnosis, network configuration, staff selection and training, budgetting and planning, football training, school bus routing, etc. A motly group served as experts: employers of students, friends, family, university professors and businessmen.

In the second individual project, the expert system developed in the first project was divided into two parts with about ten rules each. Then the two subparts were implemented in Prolog and Lisp, respectively. This excercise was intended to show the concept of subdividing (decomposing) an expert system and to illustrate the tradeoffs between Prolog, Lisp and ESIE/EXPERT. Due to the time limitations, much time could not be devoted to this project. However, the students did learn the difference between using expert shells and using programming languages and did develop an appreciation of Prolog versus Lisp tradeoffs. The decomposition of the expert system led to lively discussion of the issues concerning learning and the differences between a rule based expert system and a typical software system. Two issues were raised but not completely resolved: if learning implies addition of rules, then what does subtraction of rules mean; and can the decomposition methods used commonly in software engineering (e.g. functional decomposition, data flow decomposition, data structure design) be employed to decompose a rule based system?

The last project was a major project in which the class was subdivided into three teams, where each team was responsible for building a knowledge based software engineering project manager (KBSEPM). Each team had 10 to 12 members, supervised by a team manager. The purpose of this project was to expose the students to the process of building a realistically large expert system in a team environment. The software engineering project management problem was chosen as a class project because this subject area is of interest to students with diverse backgrounds.

The students were given literature on software engineering project management to develop expertise in this area. The KBSEPM was divided into the following subsystems:

1. Planning subsystem to determine the activities and the sequence of the activities for a given project.
2. Scheduling subsystem to determine the computing, staffing and training requirements of a given project.
3. Directing subsystem to monitor the progress of the project.
4. Controlling subsystem to control and monitor the changes introduced in the software (software configuration management).

The students were assigned to various subsystems based on their areas of interest. For example, the management students were more interested in the planning and scheduling subsystems while the computer science students chose to stay with software change control.

Three main requirements were imposed on each team. First, the students were asked to concentrate on expert system features of KBSEPM and to ignore the typical data processing features such as record keeping. The framework by Luconi et al (required reading paper #7) was used

to isolate such features. Second, each team was to identify and explicitly code the knowledge of each subsystem by using rules. Third, the students were required to minimize the number of explicit relationships and maximize the inference by using the inheritance principles of AI. For example, for a given activity in a project, say software design, several pieces of information are needed: the precendence of the activity, the resources (computing, non-computing) needed by the activity, the skills needed to perform the activity, the time needed to perform the activity, and the authority needed to extend the scope of the activity. It is possible to explicitly code all of this information for each activity in the KBSEPM. However, an intelligent knowledge based system should be able to infer most of this information. The minimization of explicit coding of information was enforced by giving the students basic project relationships (is-a, has-subparts, etc) and requiring them to justify any additional relationships and attributes. This forced the students to develop innovative inference mechanisms.

The teams built their KBSEPM in Turbo Prolog with occasional calls to Turbo Pascal subroutines for procedural calculations (e.g. software cost estimation). The KBSEPM developed by each team had 80 to 100 rules and supported about 40 queries. Details about KBSEPM are reported elsewhere [11].

4. SUMMARY OF EXPERIENCE

The course evaluations from the 33 undergraduate and 5 graduate students were very positive. Students found the projects to be the most valuable part of the course. Several students suggested that a separate course on just building large expert systems should be introduced. Students were also very enthusiastic about the required readings which gave them different perspective and understanding of the subject matter.

It was a surprise to notice that the students with prior experience and/or coursework in AI also found this course to be worthwhile and did not find appreciable duplication between the AI courses and this course. The students with no prior background in AI found that the course was at the right level and had little difficulty in understanding the material.

There were two major complaints. First, several students complained about the number of reading assignments especially in phase 1 of the course, even though they felt that the readings were very useful. Second, most students felt that there was not enough time for project 3. Other observations about the course are:

1. The book by Waterman [9]is not a suitable textbook. In future the book by Jackson [4] or Holsapple and Whinston [8] will be considered. A good text should eliminate some, but not all, of the outside readings.
2. It was found that the Turbo Prolog manual that accompanies the Turbo Prolog package is a more effective teaching tool than the book by Clocksin [10]. This is mainly due to the large number of well organized examples (over 65) discussed in the Turbo Prolog manual. In addition, these solved examples accompany the Turbo Prolog compiler on a diskette which the students can run to develop an understanding of Prolog. In the future, the Turbo Prolog manual will be used as a textbook.
3. Due to the time limitations, the students did not have an opportunity to excercise knowledge acquisition techniques adequately. For example, several students in the first project did not seriously look for outside expertise. This requirement that the expertise has to be acquired from outside needs to be enforced more strictly.
4. The expert system tools should have been covered earlier. One possible approach would be to devote an hour a week to tools starting in the second week. This would allow the theory and tools to be covered simultaneously.
5. The major project should be started early, perhaps in the fifth week of the class and the students should start building portions of the third project by week 6 of the class. It is possible to eliminate the second project thus allowing more time for the major project.

Teaching this course has been a very time consuming, nevertheless a very enjoyable experience. The diversity of student backgrounds provided several opportunities for healthy and insightful discussions.

Acknowledgements

The help, guidance, and encouragement of Professor Daniel Teichroew of the Industrial and Operations Engineering Department at the University of Michigan, Ann Arbor, is greatly appreciated.

REFERENCES

1. Terry Bahill and Ferrel, W. R., "Teaching an Introductory Course in Expert Systems", IEEE Expert, Winter, 1986, pp. 59-63
2. Eugene Charniak and Drew McDermott, "Introduction to Artificial Intelligence", Addison Wesley, 1985
3. Paul R. Cohen and Edward A. Feigenbaum, "The Handbook of Artificial Intelligence", Volumes 1, 2, 3. William Kaufmann. 1982
4. Peter Jackson, "Introduction to Expert Systems", Addison-Wesley, 1986
5. Paul Harmon and David King, "Expert Systems", John Wiley, 1985
6. Philip Klahr and Donald Waterman, "Expert Systems: Techniques Tools and Applications", Addison Wesley, 1986
7. Patrick H. Winston, "Artificial Intelligence", Addison Wesley, 1984
8. Clyde Holsapple and Andrew Whinston, ""Business Expert Systems", Irwin, 1987
9. Donald Waterman, "A Guide to Expert Systsems", Addison Wesley, 1986
10. W.F. Clocksin and C.S. Mellish,"Programming in Prolog", Springer-Verlag 1984.
11. A. Umar and D. Teichroew, "A Knowledge-Based Software Engineering Project Manager", International Business Computer Systems User Group Conference, Flint, Michigan, July, 1987. Also available as a PRISE report.

TABLE 1: REQUIRED READINGS

(1). Denning, P.J., "Towards a Science of Expert Systems", The American Scientist, Vo. 74, No. 1, Jan-Feb. 1986, pp. 18-20.
(2). Myers, W., "Introduction to Expert Systems", IEEE Expert, Spring 1986, pp. 100-109.
(3). Sternberg, R. J., "Inside Intelligence", American Scientist, Vol. 74, March-April, 1986, pp. 137-143.
(4). Dreyfus, H., and Dreyfus, S., "Why Expert Systems Do Not Exhibit Expertise", IEEE Expert, Summer, 1986, pp.86-90.
(5). Hayes-Roth, F., "The Knowledge-Based Expert System: A Tutorial", IEEE Computer, Sept. 1984, pp.11-28.
(6). Blanning, R. W., "Issues in the Design of Expert Systems for Management", National Computer Conference, 1984, pp. 490-495.
(7). Luconi, F. L., Malone, T. W., and Scott Morton, M. S., " Expert Systems: The Next Challenge for Managers", Sloan Management Review, Summer 1986, pp.3-14.
(8). Gilald, T., and Gilald, B., "SMR Forum: Business Intelligence - The Quiet Revolution", Sloan Management Review, Summer 1986, pp.53-61.
(9). Jakobson, G.; Lafond, C.; Nyberg, E.; and Piatetsky-Shapiro, G., "An Intelligent Database Assistant", IEEE Expert, Summer 1986, pp.65-79.
(10). Sriram, D. and Rychener, M., "Guest Editor's Introduction: Expert Systems for Engineering Applications", IEEE Software, March 1986, Special Issue on Knowledge-Based Systems for Engineering, pp. 3-5.
(11). "Knowledge-Based Engineering Systems: Research in Progress", IEEE Software, March 1986, Special Issue on Knowledge-Based Systems for Engineering, pp. 48-59.
(12). Hinden, H.J., "Intelligent Tools Automate High-Level Language Programming", Computer Design, May 15, 1986, pp. 45-55.
(13). McCalla, G. and N. Cercone, "Guest Editor's Introduction: Approaches to Knowledge Representation", IEEE Computer, October 1983, pp.12-21.
(14). Woods, W. A., "What's Important About Knowledge Representation", IEEE Computer, October 1983, pp.22-29.
(15). Brachman, R.J., "What IS-A Is and Is'nt: An Analysis of Taxonomic

TABLE 2: COURSE OUTLINE

	Topics	Readings
PHASE 1: Fundamentals of Expert Systems		
Week 1	Introduction to expert systems	Waterman(ch.1), Reqd. Read.(#1, #2)
Weerk 2	Basic artificial intelligence concepts and theory	Jackson[4], Cohen [3] Reqd. Read.(#3,#4)
Week 3	Features of expert systems	Waterman (ch.2-5), Reqd. Read. (#5)
	Examples of expert systems	Waterman (ch.6, 25) Reqd. Read.(#6-12)
Week 4	Knowledge representation	Waterman (ch.7), Reqd. Read.(#13-15)
Week 5	Inference and control Rule based systems	Jackson [4] Reqd. Read.(#16)
Week 6	Midterm exam 1 Project 1 due Review	
PHASE 2: Expert Systems tools		
Week 7	Expert system tools	Waterman (ch.8-10) Klahr [6]
Week 8	Logic programming Prolog programming	Reqd. Read. (#17, #18) Clocksin (ch.1-3)
Week 9	More Prolog programming	Clocksin (ch.4-6)
Week 10	Lisp programming	Charniak [2]
Week 11	Midterm Exam 2 Project 2 due	
PHASE 3: Building expert systems		
Week 12	Requirement analysis of expert systems	Reqd. Read. (7, 4)
Week 13	Design of expert systems	Waterman (ch.11-15)
Week 14	Implementation of expert systems	Waterman (ch. 16-19)
Week 15	Expert versus decision support systems Review of existing expert systems	Reqd. Read. (19, 20) Waterman (ch.20-23)
Week 16	Final exam Project 3 due	

ISSUES IN TEACHING AI / EXPERT SYSTEMS FOR BUSINESS

Hugo Moortgat
Department of Business Analysis and Computing Systems
School of Business, San Francisco State University

ABSTRACT

This paper addresses a number of issues which the author believes should help shape an AI / Expert Systems course for a Business School. Several of these issues would not appear as separate items on a course syllabus, rather they would guide an instructor in providing a general perspective on AI and Expert Systems, or in relating these fields to others the student is already familiar with. Principally, the issues deal with the general nature of computing, the use of AI technology in applications other than those strictly labeled AI, the similarities between database management and expert systems, and between knowledge engineering and systems analysis. Collectively, they suggest how the lessons of AI can be conveyed to the business student.

INTRODUCTION

The field of Artificial Intelligence in general (and that of Expert Systems in particular) has demonstrated the validity of its approaches and its applicability to a range of important real world problems. The time is ripe to begin to take this body of knowledge and to introduce it in the curriculum of Business Schools.

Before we attempt to do this, it is worthwhile to reflect a little on how this may be best accomplished. This paper addresses a number of issues which can help shape a course in AI / Expert Systems and make it an integral and unifying part of the larger Information Systems curriculum.

The material in the subsequent sections suggests that such a course provides a chance to recapture often missed opportunities for teaching students fundamental notions about computing and problem solving in general, that besides the relatively narrow area of Expert Systems AI teaches new approaches to solving problems and doing things in smarter more user-oriented ways, and that information systems students will learn to appreciate the power of the artificial intelligence approaches all the more if at least part of the applications discussed apply to information systems development and if the tasks of developing knowledge-based applications is directly compared with those of systems analysis and design of more conventional applications. The main thrust is that what the business school student is to learn from the course is a general approach to solving problems and the use of related high productivity tools which will make it possible to solve complex, ill-structured business problems.

The introduction into the Business School curriculum of an AI / Expert Systems course with objectives as outlined in the previous paragraph is not readily done with off-the-shelf materials. The textbooks which treat the general principles of the field, the documentation describing the software tools and programming systems used for the implementation of AI applications and Expert Systems are in a form which is either not adapted to the Business School student and the business user or is oriented towards a specific software package. As has been the case with Mathematics and Computer Science in the past, the materials require a change in orientation and a change in emphasis if a new class of people is to share in the benefits of this field. This adaptation should not, however, be one which strips off the fundamental concepts and leaves only the button-pushing instructions.

THE NEED FOR UNDERSTANDING THE FUNDAMENTALS OF COMPUTING

A computer, because of its programmability, is a universal machine. The same computer can be applied to a myriad of different problems as long as we devise a suitable way of converting the input data to digital form, an abstract representation of the problem class and an algorithm which specifies the transformation of input data (an instance of a problem of the given class) to the desired solution or result. Programming the computer hardware enhances its functionality and may turn it into an accounting machine, a text-editing machine, a chess-playing machine or a robot.

Students taking AI / Expert Systems courses in Information Systems programs will typically already have taken a good number of computer related courses. This would mean that they would have a certain maturity in computer matters and that they would be ready to address some of the deeper questions about computing. This can be done without undue reliance on formalism and mathematical notation. After all, business schools teach a considerable amount of economics and production management without turning these courses into plain applied mathematics courses even though many of the underlying problems are mathematical optimization problems. One can do mathematics in a qualitative verbal way and capture much of the essence without doing the detailed symbol or function manipulation.

THE SCOPE AND LIMITATIONS OF AI

Many of the fundamental concepts of computing have either come out of Artificial Intelligence research or have had their limits tested within that field.

In introducing AI applications one needs to explain what is meant by hard problems, easy problems, the complexity of a problem. One needs to give students a firm grasp of how one solves problems, of solution strategies one can employ and of the need to use heuristics, i.e. non-optimal strategies, when the search space becomes too large to apply optimal strategies.

This is self-evident, as is the idea that AI techniques are inapplicable either when one does not have a clue as to how to begin to solve a particular problem or when one understands a problem so well that a low-degree polynomial algorithm produces a unique or optimal solution to the problem. But, to basically every problem we claim to understand there is another level we do not understand. We may say that we know how to solve a set of linear equations and we have effcient algorithms for doing it, thus no AI is needed for these problems. But, on second thought, there are details dealing with numeric stability and rounding errors which are not fully understood or managed optimally and AI techniques can be used to help in that area. As another example, consider the notion of GIGO, garbage in – garbage out. We teach it to our students and it makes a lot of sense at first glance. But journalist who have scavenged the garbage cans of high government officials will tell you otherwise. And, of course, we recycle garbage to produce quality materials. Or take signal processing, which is a computer application where signals, which have been corrupted by various kinds of noise, are processed in an attempt to reconstruct the original signal. By having some notions of the prevailing kinds of errors, it is possible to look for their characteristic pattern and to filter them out, but not without some risk of removing part of the signal.

Whenever a new discipline is getting widespread recognition the danger exists that the impossible will be expected. This is definitely the case now with Artificial Intelligence. Students in Information Systems cannot be expected to become experts in cognitive science but they should have an appreciation for what classes of problems are amenable to solutions and should be cognizant of the principal issues in the current debate about the limits of formalism.

AI APPLICATIONS TO PROGRAM DEVELOPMENT ENVIRONMENTS

Artificial intelligence programs use knowledge about some application domain in deciding on a action strategy to solve a problem in that domain. The AI researchers have early on applied the fruits of their own field to the development of intelligent program development environments. Since, at least in the US, the AI programs were mainly written in LISP, their efforts went towards building LISP program development environments. Many of the ideas and tools are applicable to other environments and are the key to improved productivity in program devel-

opment. Some deal with intelligent user interfaces. Others deal with language sensitive editors, incremental program development, debuggers, browsers and so on. The programs have a degree of understanding of the activities they support and are thus able to help the user complete them. Students used to developing programs in a traditional environment, say writing COBOL programs using a standard text editor, would have a hard time appreciating the power of one of these developments environments wihtout experiencing it. Hands-on experience with a LISP environment is an important element in gaining understanding of AI's contribution to programmer's productivity.

AI AND PROGRAMMING LANGUAGES

Computers are called amplifiers of the power of the mind since they can be applied to virtually any problem the mind addresses itself to and can perform most tasks much faster than the human mind can. Of course, the computer must be programmed to perform a particular task. The mind can switch very quickly from one tasks to another and if the computer is going to be amplifying the power of the mind at all times it must be possible to change the computer's functionality or program very quickly. For instance, when prompted for a value by a program a user may realize that the requested value is the result of another, as yet, unsolved problem. The user may therefore want to suspend the execution of the currently executing program and write a function which will produce the needed value and pass it on to the program that was interrupted. Further, to be really productive in such an environment one needs computer languages with great expressive power which allow the user to express what needs to be done in terms which are close to those typically used in discussions about the application domain. The AI languages such as LISP and PROLOG can be used to illustrate the interactive, incremental development of high-level capabilities.

ROLE OF EXTENSIVE EXAMPLES

When it comes to computer application development – and even more so when new paradigms are used as is done in AI – it is important that students not only learn by doing themselves, but also by studying in detail carefully-crafted non-trivial example systems in much the same way that students learn good writing style from reading good essays and books. It is all too typical to find textbooks with sample programs a dozen or a few dozen lines long. At the end of the chapter there may be a suggestion for a project that requires a number of lines of code two orders of magnitude larger. The student has never seen anything like it! Almost invariably one is able to understand more complex systems than one can produce. Substantial example AI systems developed in LISP, PROLOG or using expert systems shells are invaluable to the students in that they give them insight into the nature of AI programming and application environments whithout having to be capable to develop these systems themselves. These systems can form the general framework within which small exercises can be fitted. Variations on parts of these can be assigned as projects.

DATABASE MANAGEMENT SYSTEMS AND EXPERT SYSTEMS

Most Information Systems students taking an AI / Expert Systems course will have taken a Database Management Systems course. This can be used to great advantage. There exist a number of analogies between database management systems and expert systems. Also, there are important applications and uses of expert systems in database systems and vice versa.

In database management systems data is separate and independent from the application programs which operate on the data. The database management system's programs can be used to manipulate many different databases. In expert systems the inference engine which contains the strategy for solving problems is separated from the knowledge base which is a set of facts and rules. The inference engine can be used with many different knowledge bases.

Current database management systems are deficient in that meaningless operations can be performed, e.g. joining two relations on attributes, such as age and skill code, having the same domain, say small integers, but having totally different meaning. In general they ignore the semantics of the data. Enhancing the database with a knowledge base can help insure the integrity of the data. Another possible application of expert systems to databases is semantic query optimization. An expert system may be able to detect that a condition could be added to a query expression wihtout affecting the result of the query but making the exe-

cution much faster. Many different types of knowledge about a database can be stored in a knowledge base and can be used by an inference engine to guide and control database operations.

On the other hand, the management of the knowledge base in a large expert system presents itself a substantial data management problem to which all kinds of data management techniques must be applied.

KNOWLEDGE ENGINEERING AND SYSTEMS ANALYSIS

Another analogy to be exploited is that between systems analysis and knowledge engineering. Information Systems students know what the rôle is of the systems analyst in the context of traditional application development and implementation. The knowledge engineer implementing an expert system is like the systems analyst, not a tool builder, but a person who has a good general understanding of the application domain and translates the expert's knowledge into a set of rules, much the same way that the systems analyst translates the user's requests into a set of system requirements specifications. Many of the same skills are needed for these two kinds of people, which is, in fact, one of the reasons why the business school is the place to train knowledge engineers who are going to implement business expert systems.

THE NEED FOR PRESENTING BUSINESS EXAMPLES OF AI / EXPERT SYSTEMS

Most expert systems texts use as examples expert systems which have been successful in the past but which have nothing to do with business. Among the systems most frequently presented are MYCIN, a medical diagnosis system and PROSPECTOR, a geological exploration system. While it is good to read about systems which have proven their expertise, the extensive presentation of rules and reasoning involving either medical or geological knowledge detracts from the understanding which students might get if the systems under discussion applied to business problems. This means that there is a need for instructional materials directly geared to the business school audience.

HARDWARE AND SOFTWARE TOOLS

A few general observations about possible hardware and software are in order. Much of the AI software has been developed on proprietary LISP machines. For research and development purposes those types of machines are indicated. However, the high-end personal computers are becoming important as delivery vehicles for expert systems and as low cost development systems. No doubt, many business schools will use the high-end personal computers to satisfy their instructional AI hardware needs. In the software area, it is important to look for standard implementations of languages, and for implementations of languages or expert systems shells which provide a growth path in terms of software and/or hardware capabilities. For LISP there now exist a Common LISP standard. For PROLOG many people consider the language as described in the book by Clocksin and Mellish as an unofficial standard and more or less complete implementations for MS-DOS machines exist. There exists a large variety of expert systems shells, with at least one vendor offering a system with a growth path going from a PC to a LISP machine. The main point to be made here is that in order to achieve the objectives discussed earlier it is important to avoid dead-end tools. It should be possible to represent knowledge using a variety of schemes. Extensibility is the quintessential property of the AI approach and the AI tools. Teaching students to take that approach and to integrate these methods and tools with standard business systems is one of the main objectives of the AI / Expert Systems for Business course.

REFERENCES

(1) Barstow, David R., Shrobe, Howard E., Sandewall, Erik, Interactive Programming Environments, McGraw-Hill, 1984.

(2) Clocksin, W. F., and Mellish, C. S., Programming in Prolog, 2nd edition, 1984.

(3) Holsaplle, Clyde W., and Whinston, Andrew B., Business Expert Systems, Richard D. Irwin, 1987.

(4) Sowa, John F., Conceptual Structures: Information Processing in Mind and Machines, Addison-Wesley, 1984.

(5) Waterman, D., A Guide to Expert Systems, Addison-Wesley, 1986.

(6) Winston, Patrick H. and Horn, Berthold, LISP, 2nd edition, Addison-Wesley, 1984.

(7) Winston, Patrick H., Artificial Intelligence, 2nd edition, Addison-Wesley, 1984.

SIMULATING NETWORK DESIGN IN THE INTRODUCTORY TELECOMMUNICATIONS COURSE

Robert A. Schultheis
Management Information Systems Department
Southern Illinois University, Edwardsville

ABSTRACT

Many introductory telecommunication courses are unable to provide hands-on experiences with telecommunications hardware and software. This lack often reduces the power of the introductory course for students. This article describes the use of design cases which require students (1) to simulate the deployment of actual hardware and software using catalogs, source books, and vendor contacts, and (2) to respond to variations in user application needs with design changes. The simulations provide students with life-like experiences and increase motivation, transfer of training, and depth of knowledge.

INTRODUCTION

A difficulty encountered by many instructors of introductory telecommunications courses is providing experiences with actual communications hardware and software, especially wide area network (WAN) hardware and software. The difficulty may be caused by the expense of WAN hardware and software, to the lack of WAN facilities on campus, or the inability to use those facilities which do exist.

The lack of telecommunications facilities creates pedagogical problems. Without experience with actual communications hardware and software, the introductory course may be confined to generalities about communication design. On the surface, this limited objective may not appear to be a serious problem since the teaching of general design principles is the basic goal of an introductory course. However, many students leave such a course without the advantage of receiving the kinds of hands-on experiences which reinforce and enrich those general principles, and thereby, improve the transfer of those general principles to practical applications in the real world. Furthermore, many students find that the "general principles" make considerably more sense when they are able to "touch and feel" the boxes and software which implement these principles.

There is a wide gap between understanding a simplified diagram of a communications network which uses standard icons to represent channels, hosts, front end processors, modems, and other equipment, and connecting real equipment to real channels. It is a lot easier to design a system with generalized symbols than to find the hardware and software which must be employed to make the design work.

Like many programs which offer the introductory course in telecommunications, the author's program is not able to provide actual experience with communications hardware and software. To provide a measure of real-life experience to students, they are required to complete a series of telecommuni-cations projects which simulate network design in the real world. The series of projects includes wide area networks (WANs), local area networks (LANs), and PBX networks. A description of the WAN projects and how

they are used in the introductory course follows.

THE BASIC WIDE AREA NETWORK PROJECT

The first design project which students must complete is the "basic WAN project." The basic WAN project requires students to (1) perform simple line-loading algorithms to identify the number of terminals needed at each location and the number, types, and speeds required for the network channels, to (2) diagram a topology for a wide area network, and to (3) identify the specific hardware, software and common carrier products for the network by brand name and model, version, or product number. The basic WAN project is presented in Appendix 1.
The basic project limits the network to a single application: the transmission of sales orders for an order entry department with both local and remote sites. The actual order form is not presented to students, nor are they required to cull through files of orders to generate the minimum, average, and maximum number of data and control characters the system produces daily. Instead, this information is provided to them. The instructor plays the role of the user, or the sales manager. Students are expected to query the "sales manager" in order to clarify user requirements for the order entry application.

Students are also expected to use an elementary algorithm for estimating the number of terminals needed. The algorithm is:

(1) CPD = OPD x (CPO + CCPO) x (TE + GR)

(2) NOT = CPD / ACP

Where: CPD = characters per day, OPD = the number of orders per day, CPO = the number of characters per order, CCPO = the number of control characters per order, TE = an allowance for retransmissions due to transmission errors, GR = an estimated allowance for the growth rate of traffic volume over the short term (such as five years), NOT = minimum number of terminals needed, and ACP = average characters produced per day by an order entry clerk.

Students are then expected to use the following algorithm for estimating the volume of data traffic produced by each site in bits per second (bps):

MLS = (CPD x 10) / SPD

Where: 10 is the estimated number of bits to produce each character, SPD = the number of seconds in an eight-hour work day, and MLS = the minimum line speed stated in bits per second.

Once the minimum number of terminals required at each site and the minimum line speeds for the data traffic have been determined, students develop a tentative topology of the network. The topology is developed as a diagram shows the host mainframe, the terminals at each site, and whatever equipment and lines are needed to carry the signals between this host and each site. Students test the feasibility of the topology by identifying specific hardware, software and common carrier products which would be needed to complete the online network if the network were live instead of simulated.

RESOURCES FOR PROJECT

A number of resources are used to help students identify vendor and carrier products. Some of these resources have been placed in an MIS Library available for MIS students. Included as resources in this library are a wide collection of communications equipment and software catalogs from WAN, LAN, and PBX equipment and software vendors, the multi-volume sets of Data Pro Reports On: Telecommunications, Data

Communications, and Office Systems, and a variety of communications trade journals, such as Telecommunications, Teleconnect, Telecommunication Products Plus Technology, Micro Communications, Data Communications, and On Communications.

Students are also encouraged to use selected vendors of communications systems in the St. Louis area and selected contacts in St. Louis area common carrier firms to obtain specific data on hardware, software, and carrier products for use in the network. The point is that students are expected to use these resources to provide specific brand names, model numbers, and prices for the hardware and software products they recommend for the project--right down to the cables needed to connect the equipment and to make the system work. Examination of the "The Project Report" section of the project will provide an understanding of the detail required of students in the project.

VARIATIONS ON THE BASIC PROJECT

The basic project is followed sequentially by one or more variations which add sophistication to the design and additional, comparative design experiences for the student. Students must redesign the network to account for these varia-tions and submit additional "Reports." A description of these variations follows.

VARIATION 1: COMPUTING TRAFFIC VARIATIONS WITH A DESIGN SPREADSHEET

Once students have had experience in using the two algorithms to determine the number of terminals and data volume, they are asked to develop a spreadsheet template to compute these data. Using the template, they are asked to recom-pute and, if necessary, redesign the network to accommodate these modifications:

(1) Growth in sales. They are to plan for an estimated 10% growth in traffic volume for the next five years

(2) Busy-hour traffic. The orders peak during two busy-hour periods each day. One period is from 10-11 a.m., and the other is from 2-3 p.m. During these time periods 40%, and 30% of the traffic volume occurs, respectively.

VARIATION 2: SWITCHING TO BATCH PROCESSING

The basic WAN project is designed for an online order entry system. In the second variation, students are to redesign the network for batch processing of the sales orders. The redesign means considering switched lines and post-eleven transmission of data to reduce costs.

VARIATION 3: PROVIDING BACKUP SERVICES

Students are asked to provide backup equipment, software, and channels to insure that the online system can continue to function in the event of equipment or channel failure. This variation forces students to consider disaster planning and the use of switched lines and autodialing equipment to handle failure smoothly and swiftly.

VARIATION 4: PROVIDING MULTIPLE APPLICATIONS

In this variation, other applications are added to the network, such as payroll and purchase order applications. The forms are described and the minimum, average, and maximum number of forms shipped per day along with the minimum, average, and maximum number of data and control characters are provided to students. Students are asked to adjust their design spreadsheets to produce terminal and line information when

multiple application traffic occurs and to redesign the network for these added applications. The addition of other applications to the basic WAN project presents students with the design considerations necessary for both online and batch processing on the network.

OTHER NOTES

The lectures and discussions which support the projects provide students with many more factors which can be used to add sophistication to the line loading and terminal determination than are included in the algorithms the students use. It is important that students recognize that the algorithms are somewhat simplified planning tools. For example, the ASCII asynchronous protocol of 10 bits is used to convert data and control characters from bytes to bits. Students should recognize that other message framing protocols would use other conversion ratios.

The purpose of the catalogs, Datapro volumes, vendors, and carriers is to extend student experience beyond textbook generalities and tp force them to consider real vendor and carrier products, to face some of the real decisions inherent in network design, and to recognize the detailed nature of an actual network. For example, use of the catalogs quickly informs students that a modem is not always merely a modem; that many vendors provide varieties of modems which perform functions typically attributed to other DTE and DCE devices. They soon learn that there are many variations of front ends, modems, multiplexers, and other DTE and DCE equipment which can be fitted to specific network problems; that the descriptions of such equipment usually included in textbooks are far too simple to describe the real world.

The purpose of the variations is to permit students to see the impact on network design of differing applications, online versus batch operations, varying traffic patterns, and security precautions. The variations also provide practice in the use of network design principles.

The LAN and PBX projects are similar in nature. The LAN variations include adding (1) modem pools for external data bases, (2) bridges to extend the LAN, and (3) gateways to the host computer. The PBX variations include adding (1) electronic key systems, (2) voice mail, and (3) data switching to multiple hosts. Students complete all the projects in two-member teams. Teamwork generally has served to increase the discussion of network options by students, and to lead to greater learning from the projects.

APPENDIX 1
THE BASIC WIDE AREA NETWORK PROJECT

You have a multiport IBM 4341 host computer at Granite City, Illinois. You wish to locate a group of terminals in a town about 25 miles north of the city, another group of terminals in a town 300 miles north of the city, and another group of terminals at the host site. You wish to acquire IBM 3278 terminals for these three locations and you expect that these terminals will need to communicate with the EBCDIC host during the regular working day (8 am to 5 pm).

The purpose of the terminals is to provide online order entry departments at the three locations. You have estimated that the average number of characters in an order is 300 and that there is also an average of 40 control characters per order. The average number of orders you expect to be produced daily at each site are:

Granite City	1,200
25 Miles North (North 1)	3,000
300 Miles North (North 2)	4,000

You wish to allow for a 1/2% transmission error rate, and you

estimate that the order entry clerks will produce one character per second for an effective work day of six hours, given breaks and interruptions, from Monday through Friday.

THE PROJECT REPORT

You are to submit a final report which includes the following:

(1) A brief description of the network, any assumptions you found necessary to complete your work, and the rationale for the topology, channels, equipment, and software you selected.

(2) A detailed map of the topology showing the types and location of all equipment and channels.

(3) A list of the specific carriers, the names and prices of their channel products, and speeds of the channels you selected.

(4) A list of the brand names, model numbers and prices of all the software and hardware products you selected.

BALANCING THEORY AND PRACTICE IN A COURSE ON NETWORKS AND DISTRIBUTED SYSTEMS

Amjad Umar
School of Management,
University of Michiagn-Dearborn, Dearborn, Michigan, 48128
and
Program for Research in Information Systems Engineering (PRISE)
Industrial and Operations Engineering Department
University of Michigan, Ann Arbor, Michigan, 48109-2117

ABSTRACT

The experience of teaching a senior/graduate level course in computing networks and distributed systems for students in engineering, management and computer science in a small university is described. A balance between theory and practice had to be reached due to the diverse interests, academic preparation, and industrial exposure of the students. The topics covered, projects assigned, and experience gained are discussed.

1. INTRODUCTION

A quality education in contemporary computer and information science requires that students gain theoretical knowledge as well as practical experience with realistic systems. This is especially important for introductory courses in computing networks and distributed systems, a topic where a great deal of literature on current practices as well as underlying theories is accumulating rapidly.

A senior/graduate course in computing networks and distributed systems was first offered in the Winter Semester of 1986 at The University of Michigan-Dearborn (UM-D). The campus is small, constituting about 7,000 students, with an interdisciplinary program in Computer and Information Systems. The experience of teaching this course has shown that an interdisciplinary course in computing networks and distributed systems can be offered in a small university in a cost effective manner. The course has now been adopted as a regular course to be offered at least once a year.

This course has three objectives: to describe how the communication hardware and software are synthesized into network architectures, local area networks and distributed systems; to show the current practical applications of computer networks and distributed systems in organizations; and to prepare students for advanced studies by introducing the main research and development trends. This course differs from courses being offered at other universities [13] and the "intensive" industrial seminars which primarily focus on vendor products because this course attempts to cover the basic theory as well as the practical applications of networks and distributed systems. Furthermore, the course is intended for a mixture of engineering, management and computer science students with different interests, expectations and preparations; and qualified student assistants and adequate lab facilities are not readily available.

The purpose of this article is to share the experience of teaching such a course by describing the course topics, reading assignments, student projects, and lessons learned.

2. COURSE CONTENT AND TOPICS

This course was first offered in the Winter of 1986 and has been offered again in the Winter of 1987. About eighty students have enrolled so far. The experience gained in the first course has been utilized to develop the contents, reading assignments and projects for the second course which is discussed in this paper.

The book by David Stamper [14] is currently being used as a textbook. The main reason for choosing this book as the text is that a case study is introduced in the first chapter and is expanded systematically to introduce and illustrate various concepts throughout the text.

One limitation of the text is that distributed processing is not covered adequately for this course. Consequently, required readings were prepared with papers on computer communication trends, use of PC networks in organizations, ISDN, local area networks, distributed systems, and AI in networks, to augment the textbook. The required readings are shown in Table 2.1. In addition, several books [1-3, 5-13] and IEEE tutorials [4] listed in the references, were placed on reserve in the library.

Table 2.2 shows the outline of the course topics with reading assignments, and examinations. The course topics are divided into four phases: communication hardware, communications software, network architectures and local area networks, and distributed systems. Each phase is terminatd with a project. The phases are briefly described here and the projects are described in the next section.

In phase 1, the applications of distributed systems and networks are illustrated and the physical layer of the ISO/ANSI reference model is covered. The main emphasis in this phase is to introduce the basic concepts of communications theory and explain the various communication hardware devices. Students are assigned readings to review the state of the art in modems, T1 communication devices and ISDN.

In phase 2, the focus is on developing an understanding of communications software. The functions of low level communications software are illustrated by explaining the IBM PC serial board (the 8250 chip) and then developing a "skeleton" terminal emulator. Several textbooks describe the 8250 chip (see for example Willen and Krantz [14]). A project, described in detail in the next section, which expands the terminal emulator into a file transfer package, is assigned to students. The data link and higher order software is illustrated by using terminal emulation and file transfer examples.

In phase 3, the communication hardware and software concepts are synthesized and illustrated by network architectures and local area networks (LAN). The network architecture concepts are explained through a detailed discussion of IBM's System Network Architecture (SNA). Other network architectures (DNA, BNA) are covered by homework assignements. The concepts of LAN are illustrated by developing an abstract LAN which satisfies given user requirements of connecting 3 PCs to share a database. The students are assigned several current papers on commercially available LANs. The topics in this phase also include discussion of protocol converters, gateways and bridges.

The issues of distributed systems and management of networks are introduced in the final phase. The focus is on the research and technical issues in distributed systems like database synchronization, database translation, query optimization and distributed application design. The reading assignments cover these topics in detail. The issue of managing distributed systems and networks is discussd in the last week of the course. This session integrates the technical concepts with management issues of planning, scheduling, directing and monitoring of distributed systems. In this phase, the students review current literature and report on trends.

3. PROJECTS AND FACILITIES

The projects assigned in this class were designed to serve several specific purposes, within the overall objectives of the course. First, the analytical paper and pencil analysis of computer communication networks must be emphasized. Second, the students must learn some of the commercially available tools and packages (terminal emulators, file transfer packages, file/print servers). Third, the students must have "hands on" exposure to basic low level communication software needed for terminal emulators and file transfer packages. Fourth, the students must learn how to synthesize the hardware/ software concepts in evaluating and choosing LAN. Finally, students should be given an opportunity to study current research efforts for more advanced studies.

Four projects are assigned to support these objectives. The laboratory facilities consist of four IBM PCs connected to a remotely located Amdahl host under the Michigan Terminal System (MTS). In addition, an AST local area network consisting of four PCs was installed for student

experimentation. Turbo Pascal was used as programming language.

In the first individual project the students design a communication system for a case study described in Table 3.1. To augment the case study, current literature on available multiplexors/concentrators with cost figures is placed on reserve in the library so that the students can extract useful information from commercial literature. The students are expected to propose a minimum cost configuration for this problem.

In the second project, the students are given the "skeleton" terminal emulator for an IBM PC written in Turbo Pascal shown in Table 3.2, and two assignments: complete the skeleton program into a functional emulator and then extend it further to perform file uploading/downloading from the host. Turbo Pascal has been found to be especially useful in this excercise because the PORT command in Turbo Pascal allows students to directly read/write from port addresses like status registers, transmission/receive registers and line/modem control registers. For example:

```
PORT[$3F8] := $1A ;
```

writes hexadecimal 1A ($1A) to absolute location 3F8, which is the communication I/O register of an IBM PC serial board (8250 chip). Similarly:

```
INC := PORT[$3F8] ;
```

shows that the location hex 3F8 was read and the results were placed in variable INC. This use of PORT command in Turbo Pascal combines the high level programming language features of Pascal with low level programming needed to directly access the ports of the 8250 chip. Without this feature, it would be necessary to use the 8088 assembler; this is not acceptable for a class with mixed backgrounds.

The third project consists of three parts. In part a, the students select and use an existing terminal emulator and file transfer package to transfer files between the host and an IBM PC. In part b, they study the documentation of the AST LAN, develop a reference card of instructions and then use these instructions to bring up the AST LAN and share files. In part c, the students survey the current literature on local area networks and select and recommend a local area network to satisfy the needs of an organization. The actual problem studied by the students is shown in Table 3.3. The students are given access to several magazines and articles (e.g. PC Magazine, Dec.9, 1986 issue; PC Week, November 11, 1986 issue; PC World, February, 1985 issue) so that they learn to use current literature in choosing a LAN.

The final project is a research project where the students choose, from various topics, an area of interest and make short presentations to the class. The topics suggested are: local area networks performance evaluation, comparison of network architectures (MAP, BNA, DCA, SNA, ISDN), special facilities (fiber optics, sattelites, etc), application of AI in networks and distributed systems, management, policy and security issues in networks, current and future status of communication industry,and research directions in distributed systems and networks. The students are expected to display an understanding of the area chosen and the main issues involved. The projects chosen by students varied widely; in the Winter of 1986 with roughly 30% choosing management issues, 40% selecting network architectures/LAN and the others primarily concentrating on hardware facilities like fiber optics and satelites. This year 50% of the students have shown interest in LAN, 20% have chosen application of AI in networks and the rest have selected projects in network management, distributed databases and network architectures.

4. SUMMARY OF EXPERIENCE

Teaching this course has convinced us that it is possible to cover a good mix of practical as well as theoretical aspects of computing networks in one three credit hour introductory course with students from mixed backgrounds (several students had one programming and one "computers for management" course). The main challenge in developing and teaching this course is to choose a suitable middle point between a terminology course and a specialized course in one area (e.g. programming or performance evaluation).

The Winter'86 course required more programming and less outside readings. Based on the student suggestions, one programming assignment was replaced with the LAN operation and selection excercise

(project 3), and several outside readings were assigned. The main observations about the Winter'87 course are:

1. Surprisingly, most students did not complain about additional readings. The students have been especially enthusiastic about reading assignments on LANs.
2. Project 2 takes the longest and the deadline had to be postponed twice. However, most students, including those with prior experience in networking, feel that this is an excellent project and should be continued.
3. Some "hand holding" is necessary for students with non-technical backgrounds, especially in project 2. In future, a basic course in architectures may become a prerequisite.
4. Several students owned their PCs and wanted to do project 2 at home. However, the students could not borrow Turbo Pascal compilers and thus had to work in the Computing Center. Policies may be needed to allow borrowing software in a manner similar to borrowing books from the library.

Comments and course evaluations of this course are very encouraging. For example, several firms in the local area have directly asked to reserve seats for their employees in this course. While it is not possible to entertain these requests, it does appear that a good balance between theory and practice in a contemporary field like computing networks and distributed systems is greatly appreciated by the students and their employers alike.

Acknowledgements

The help, guidance, and encouragement of Professor Daniel Teichroew of the Industrial and Operations Engineering Department at the University of Michigan, Ann Arbor, is greatly appreciated.

REFERENCES

1. Bartree, T. C., 'Data Communications, Networks, and Systems', Howard Sams and Co., 1985
2. Chorafas, N. D., 'Designing and Implementing Local Area Networks', McGraw Hill, 1984.
3. Ellis, R., 'Designing Data Networks', Prentice Hall, 1986
4. IEEE Tutorials on:
   a. Computing Networks by Ebrams, Blanc and Collins
   b. Local Computer Networks by Thurber and Freeman
   c. Local Network Technology by Stallings
   d. Microcomputer Networks by Abrams
   e. Distributed System Design by Palmer and Marianni
   f. The Security of Data Networks by Davies
   g. Principles of Communications and Networking Protocols by Lan
5. Kreager, Paul, 'Practical Aspects of Data Communications', McGraw Hill, 1983
6. Lane, M., 'Data Communications Software Design", Boyd and Fraser Publishing, 1985
7. Martin, J., 'Telecommunications and the Computer', Prentice Hall, 1969
8. Martin, J., 'Design and Strategy for Distributed Data Processing', Prentice Hall, 1981
9. McNamara, John, 'Technical Aspects of Data Communication', Digital press, 1977
10. "Networks and Communications Buyer's Guide", Digital Equipment Corporation Publication, Fall, 1986.
11. PC Magazine, Special LAN Issue, Vol. 5, No. 21, December 9, 1986.
12. Sherman, K., 'Data Communications - A User's Guide', Reston Publishing, 1981
13. Sherman, M. and Mark, A. "Using Low-Cost Workstations to Investigate Computer Networks and Distributed Systems", IEEE Computer, June 1986, pp.32-40.
14. Stamper, David, "Business Data Communications", Benjamin/Cummings Publishing company, 1986.
15. Tannenbaum, A. 'Computing Networks', Prentice Hall, 1981
16. Willen, A. and Krantz, J., "8088 Assembler Language Programming the IBM PC", Howard Sams & Co., 1983

TABLE 2.1: REQUIRED READINGS

1).J. Mayo "Computer Communications: Today and the Decade Ahead", First Pacific Computer Communications Conference, Seoul, Korea, Oct. 1985.
2).J.G. Burch, "Will This Finally be The Age of Integration?", Infosystems, 5/86, pp.38-40.
3). N. Watkins, "Integrating PC users and MIS", Infosystems, 5/86, pp.56-59.
4).A. Kavanaugh, "Survey of University Telecommunications", Business Communications Review, Nov. Dec. 1986, pp.9-15.
5).E. Mier, "The Future of Modems: Will it be boom or gloom?" Data Communications, McGraw Hill, Sept. 1986, pp.55-64
6).Umar. A, "ISDN and Future of Telecommunications", Working Paper, School of Management, Univ. of Michigan-Dearborn, Jan. 1987.
7).E. Mier, "Long overdue, T1 takes off- but where is it heading?", Data Communications, McGraw Hill, June 1985, pp.120-140.
8). J. Sachs, "Six Leading LANs", PC World, Feb. 1985, pp. 108-128.
9). F. Derfler, "The IBM Token-Passing Network", PC Magazine, January 13, 1987
10). A. Baratz et al ,"SNA Networks of Small Systems", IEEE Journal on Selected Areas in Communications, Vol. SAC-3, No. 3, May 1985, pp.416-426.
11). J. Larson, " A Flexible Reference Architecture for Distributed Database Management", ACM Thirteenth Annual Computer Science Conference, March 1985, New Orleans, pp.58-72.
12).A. Umar, "Distributed Data Processing: General Considerations and a User Experience", Association of Educational Data Systems Conf. Proc., May, 1982, Orlando, Florida, pp.59-72.
13).G. Champine, "Six Approaches to Distributed Databases", Datamation, May, 1977.
14).J. Rothnie and N. Goodman, "A Survey of Research and Development in Distributed Database Management", Third Intern. Conf. Very Large Data Bases, 1977.
15).A. Umar and T. Teorey, "A Generalized Approach to Program and Data Allocation in Distributed Systems", First Pacific Computer Communications Conference, Seoul, Korea, Oct. 1985, pp. 462-472
16).J. Rockart, "The Management Issues in Centralization vs Decentralization of Information systems", SHARE 48, March, 1977.
17).L. Mantelman, "AI carves inroads: Network Design, Testing and Management", Data Communications, McGraw Hill, July 1986, pp.106-123.
18).L. Mantelman, "Turtle Geometry: Unused AI Tool Opens a Window on Networking", Data Communications, McGraw Hill, July 1986, pp.127-140.
19). J. Foley and Y. Weon-Yoon, "The Current Status of MAP", SIGCOMM'86 Symposium on Communication Architectures and Protocols, August, 1986, Stowe, Vermont, pp. 6-12.

TABLE 2.2: COURSE OUTLINE

Week	Topics	Readings
Phase 1: Communication Hardware		
Week 1	. Introduction and background . Communication concepts	Stamper (ch.1) Reqd. Read. (#1)
Week 2	. Basic communication theory	Reqd. Read. (#2,#3) Martin [7]
Week 3	. Data communication hardware	Stamper (ch. 2, 3)
	. More data communication hardware . Terminals, front end processors . Switches, multiplexors	Stamper (ch.4, 5) Reqd. Read.(#4-#5)
Phase 2: Communication Software		
Week 4	* . Project 1 due . Data link controls (BSC, SDLC)	Stamper (ch. 6) Reqd. Read. (#6,#7)
Week 5	* Midterm exam 1 . Review	
Week 6	. Communication software principles . Terminal emulation, file transfer	Lane [6] Stamper (ch. 7)
Week 7	. Commun. programming in Pascal . More on communication software	Willen (ch. 7) Lane [6]
Week 8	***** Midterm recess - no classes	
Phase 3: Network Architectures and LAN		
Week 9	* Project 2 due . Network architecture principles	Stamper (ch. 8)
Week 10	. Network examples . IBM's SNA, Digital's DNA	Stamper (Ch. 9) Reqd. Read. (#10)
Week 11	* Midterm Exam 2 . Local area network principles	Stamper (ch. 8)
Week 12	. LAN analysis and examples	Stamper (ch.9,10) PC Magaz. (Dec.86) Reqd. Read. (#8,#9)
	. Network performance analysis	Ellis [3]
Phase 4: Distributed Systems		
Week 13	* Project 3 due . Network design . Intro. to distributed systems	Stamper (Ch.11,12) Reqd. Read. (#16)
Week 14	. Distributed processing and distributed databases	Stamper (Ch. 13) Reqd. Read. (#11, #12, #13)
Week 15	. Management issues (planning, staffing, organizing) . Trends in networks and distributed systems * Project 4 due	Reqd. Read. (#16) Martin [8] Reqd. Read. (#14, #15, #17, #18,#19)
Week 16	* Final exam	

TABLE 3.1: A Communication System Design Problem

ABC Supermarkets is a large regional grocery chain headquartered in New York with 40 stores in Pennsylvania, Michigan, Virginia and New Jersey. ABC has recently installed a computerized checkout system to control inventory and to speed up the checkout process.

This new checkout system has a stand-alone processor at each store which is connected to each checkout position. This processor maintains a database of store inventory, a database of daily transactions and a database of prices of items sold at this store. Several point-of-sale (POS) terminals are connected to the processor.

At checkout, a point-of-sale terminal reads the code on an item and gets the price information from the store's price database. The POS terminal also posts each sale to the transaction database. The processor updates inventory every hour.

At the end of each day, the main computer facility in Washington reads the transaction and inventory files and sends a new price file.

Given the information below, determine the least expensive configuration (WATS, leased, direct dial) to allow the main computer to read/send files to the 40 stores each night (within 7 hours).

Detailed Information

1. Each store handles 1000 transactions of 100 items each and each item requires a record of 40 characters.
2. Each store carries 25,000 items, which are represented in the inventory database by a 100 character record and in the price database by a record of 20 characters
3. Data are transmitted at 10 bits per character
4. Hourly rate for WATS service is $6.11 and the access charge is $30.6 for each line.
5. The leased line charge is $220 per month
6. Direct dialing line cost is 20 cents for the first minute and 12.8 cents for each additional minute
7. Modem costs are given below:

Speed	Cost (monthly)
2400	$85.36
4800	$171.00
9600	$227.0

8. You should consult current literature on multiplexors and concentrators for costs

TABLE 3.3: Local Area Network Selection

You have been asked to select and recommend a LAN for the branch office of an insurance company. The branch office has a small budget of about $10,000 dollars for this project. The branch office currently has three IBM PCs for their claims department for entering claims and have just discovered that entering claims on standalone PCs does not help them to access information from their corporate headquarters. As a matter of fact they have found that standalone PCs are not very useful except for word processing. They would like to:

1. Allow the three PCs to share a laser printer
2. Connect the three PCs so that all three PCs can share information
3. Somehow connect the three PCs to the headquarter computer (an IBM MVS system running SNA) so that each PC can directly send/receive information from/to the host in addition to other branch office PCs.
4. Use the current wiring in the building if possible
5. Allow some growth so that in future 20 to 30 PCs could be supported in the branch office.

Your proposal should show all the hardware and software needed with costs.

TABLE 3.2: A Skeleton Terminal Emulator

```
    (* TERMINAL EMULATOR SKELETON  PROGRAM *)

PROGRAM TERMULATOR (INPUT, OUTPUT); VAR    INC, OUTC : CHAR;
      SERIAL, MODSTAT, MODCTRL, LINESTAT, LINECTRL, CNT : INTEGER;
      STAT1, STAT2, STAT3 : BYTE;

BEGIN
      (*INITIALIZE*)
      (* ASSIGN PORT ADDRESSES  *)
      SERIAL := $3F8; MODSTAT := $3FE; MODCTRL := $3FC;
      LINESTAT := $3FD; LINECTRL := $3FB; CNT := 0;
      INC := ' '; OUTC := ' ';


      (*INITIALIZE UART TO 7 BITS, ONE STOP BIT, EVEN PARITY *)
      PORT [LINECTRL] := $1A;

      (*SET BAUD RATE TO 300 BPS, DIVIDE THE CLOCK BY A NUMBER *)
      (*1. SET HIGH ORDER BIT OF LINECTRL TO  1 ... DO IT YOURSELF *)

      (*2. SET SPEED LIMIT ... VERIFY IT *)
      PORT [$3F8] := $80; PORT [$3F9] := $01;

      (*WAKE UP HOST*)
      PORT [SERIAL] := $0D;
      WRITELN (TRM, 'TERMINAL EMULATION PROGRAM IS PROCEEDING');

      (* ******** MAIN CODE **********)
      WHILE (OUTC [] '^')       (* THIS TERMINATES PROGRAM *)
        DO
          BEGIN
            STAT2 := PORT [LINESTAT];   (*  BIT 0 = 1, BYTE RECEIVED*)
            (*READ FROM THE LINE*)
            IF ((STAT2 AND $01) = $01)
            THEN  BEGIN
                INC := CHR(PORT [SERIAL]);
                   (* PORT RETURNS A BYTE, CONVERT TO CHAR  *)
                WRITE (TRM, INC);
                (*ERROR CHECKING*  ... DO IT YOURSELF*)
                (*CHECKING FOR END OF MTS TRANSMISSION, ?, #, etc*)
            (*DELAY TO CHECK FOR FURTHER INCOMING DATA*)
            (* MAY NEED TO DO *)
            END;      (*END OF READ LOOP *)

            (*WRITE TO THE LINE*)
            (*GET STATUS  *)
            STAT2 := PORT [LINESTAT];
            (* IF BIT 5 =1, THEN TRANSMITTER EMPTY, CAN SEND,, DO IT*)
            IF  .........
            THEN BEGIN
                 READ (TRM, OUTC);
                 PORT [SERIAL] := ORD(OUTC);
             END;
          END;
      END.
```

DATA COMMUNICATIONS IN THE INFORMATION SCIENCES/SYSTEMS MAJOR

Jack E. Leitner
Division of Computer and Information Sciences
University of North Florida
Jacksonville, Florida 32216

ABSTRACT

With the growing interdependence of computing and communications a thorough understanding of data communications is imperative for information processing professionals. The University of North Florida's Division of Computer and Information Sciences has added a one semester senior undergraduate/beginning graduate course to its curriculum to address this need. The goal of this course is to give the student a knowledge of the vocabulary of data communications and an intuitive grasp of its concepts so that they will be able to read and understand the literature, converse intelligently with vendors, and make informed decisions when selecting communication systems.

INTRODUCTION

Most of the changes in the data processing environment in the recent past have been dynamic and obvious. The increasing capabilities of software plus the combination of new hardware technology and falling hardware prices mean that new approaches to problems must be explored. Central to this, though often not recognized, is the increasing importance of data communications.

Data communications affects every organization that uses a computer system. From the small microcomputer that drives a single printer to the largest systems connected to a variety of devices and networks, all depend on sending information from one place to another. It is important that a data processing professional be at least conversant with major data communications concepts.

To provide its students with a knowledge of data communications the University of North Florida has added to its curriculum a one semester course in data communications for senior undergraduates and beginning graduate students. The course resides in the Information Sciences and Information Systems majors and does not assume an extensive mathematical background. Prerequisites for the course include Data and File Structures, Systems Software and the students are encouraged to

have a course in Computer Hardware Organization. A constraint on the material covered is that in a one semester course it is impossible to deal in depth with the entire spectrum of data communications. With this in mind, the objectives for the course are to prepare the student to be able to read the literature with understanding, to converse knowledgeably with vendors and communications engineers, to make informed decisions regarding the implementation of data communications systems, and to work effectively with systems using data communications.

COURSE CONTENT

The method chosen to meet the objectives of the course is to emphasize concepts and terminology. By concentrating on a conceptual approach to data communications, it is possible to cover the major aspects of the technology without spending to much time on nonessential details. The only technical details included in the course are those that facilitate an intuitive grasp of the subject.

Data communications has an extensive vocabulary and without a knowledge of its vernacular it is difficult to discuss any topics meaningfully. Unfortunately, because the field of data communications has arisen from a combination of other fields many of the terms have more than one meaning. In addition, the advances in hardware have resulted in equipment that enables a single "box" to perform operations that previously required several different "boxes." Conversely, the change in the regulatory environment has permitted some previously combined operations to be unbundled into multiple "boxes." These new products must be named so the result is that vendors have added to the vocabulary confusion by inventing new terms or stretching the meaning of existing ones. To help overcome this, terminology that emphasizes the functionality of the concept or device is used in the course and possible misinterpretations explained.

In keeping with the target audience of the course the orientation is neither wholly technical or engineering, nor wholly managerial. Instead, these views of the subject are combined into a user viewpoint that includes aspects of both. That is, the approach is taken that these students are going to have to function in an environment that includes data communication systems.

This course is organized into four general areas: physical concepts, logical organization, communications hardware, and the public environment. Physical concepts includes basic communications theory, i.e. how to physically move the data. Logical organization is a discussion of how the data is organized as it is moved. Communications hardware are the various devices used in a communications system. The public environment refers to the different services offered by communications companies. All these are overlapping areas and throughout the discussion each are tied in with the others.

To be able to fully understand the communications process there must be an underlying appreciation of the various

physical considerations involved in electronically transporting data. The students are shown why many times it is necessary to convert signals from one form to another (e.g. from digital to analog), different technologies for doing that, how noise interferes with the process, and why there are limitations on the speed of transmission. The various transmission media are covered as well as topologies the media can assume and the standards for connection to the devices.

A logical ordering of the transmitted data is required so that the receiving device can make the correct interpretation of the received data. The students learn the fundamentals of representative protocols and some techniques used to provide data integrity. At UNF there is also a course in Networks and Distributed Processing so the discussion of communications software does not extend beyond the data link layer (connection of two adjacent devices) of the International Standards Organization's reference model for interconnecting communications devices into networks.

In data communications, depending on the size of the system, many hardware devices can be required for the movement of information. These devices handle not only the physical transport but also the logical processing before and after transmission. Some of these devices function in a straight forward way but others are increasingly complex. The students gain a familiarity with the functional requirements of a complete system and how the devices interact with each other. Among others, modems, multiplexers, terminal controllers, concentrators, and front end processors are covered.

To be able to communicate off-site an organization must use a common carrier to provide the transmission facilities. With the continuing program of government deregulation and the recent breakup of AT&T there is a vast array of products and services that must be evaluated when implementing a data communications system. The students are exposed to the basic services offered by the common carriers as well as the more recent offerings such as digital termination services and packet networks.

COURSE PROJECT

A term project is assigned that involves writing the software to allow communication between two or three devices. The students are expected to produce a fully functional, documented system and demonstrate it to the instructor at the end of the term. Two variations of the project have been used.

One variation of the project is to have the students write a terminal emulation program to enable an IBM PC to function as an asynchronous terminal connected to an IBM 4341 through an IBM 3705 front end processor. The program must allow the PC to emulate all the functions found on a typical terminal as well as permit text file uploading and downloading and output to a local attached printer.

Another variation of the project that has been used is to have the students design and implement a system that will

allow two PC's (three for graduate students) to communicate with each other through their serial ports. The system definition that the students are assigned is a subset of IBM's popular Binary Synchronous Control protocol (better known as BiSync or BSC.) The system must be able to accept input from the keyboard and diskette files and deliver that input to the receiving PC's screen or a diskette file. The transmitted information can be either text or binary data.

Student enthusiasm for both projects has been good with several students reporting that the resulting programs have been useful to them outside of class in their personal systems or at work.

SUMMARY

To operate effectively in today's environment a data processing professional must be aware of communication issues and be able to make informed decisions. With the knowledge from this course the student will be able to function competently when dealing with data communication concerns. A measure of the interest in the course is that it has proven popular, not only with the regular students, but also with professionals currently working in the field who wish to upgrade their skills.

BIBLIOGRAPHY

1. Held, G. Data Communications Networking Devices. John Wiley & Sons. New York, New York. (1986).

2. Housley, T. Data Communications and Teleprocessing Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey. (1987).

3. Loomis, M. E. S. Data Communications. Prentice Hall, Inc. Englewood Cliffs, New Jersey. (1983).

4. DeNola, Lynn A. Data Communication Fundamentals and Applications. Merrill Publishing Company. Columbus, Ohio. (1987).

5. Pooch, U. W., Greene, W. H., and Moss, G. G. Telecommunications and Networking. Little, Brown and Company. Boston, Massachusetts. (1983).

6. Sherman, K. Data Communications. Reston Publishing Company, Inc. Reston, Virginia. (1985).

7. Stallings, W. Data and Computer Communications. Macmillian Publishing Company. New York, New York. (1985).

LOCAL AREA NETWORKING IN AN ACADEMIC ENVIRONMENT

Dr. Michael Bronner
New York University
New York, NY 10003

KEYWORDS

Local Area Network(s), Token-Ring Networks, LAN(s), Academic, University Networking, Academic Advisement.

ABSTRACT

While the local area network (LAN) has proven to be advantageous for many areas of business and business-related activities, there has been little research exploring the academic use of the LAN in this environment. With the advent of IBM's relatively-new Token Ring Network in late 1985, one university academic program has explored the technology on a small scale with some interesting results.

When considering a LAN for institutional use--such as in an academic environment--a number of factors must be carefully weighed. These factors include not only the costs of the technology, the probable outcomes and the training needed, but intangibles such as faculty attitudes and the added interest of internal (as well as external) colleagues.

The LAN can provide effective support in an academic environment with regard to student records and advisement; sharing of programs, files, and hardware; and enhanced communication among faculty members.

On the other hand, the LAN can be expensive, require extensive learning time, and be subject to problems in both hardware and software configurations.

This paper, therefore, details the experiences involved in establishing a protypical LAN in an academic program and suggests some future concerns and opportunities within an academic environment.

THE SETTING

New York University (NYU) is the largest private university in the country with nearly 47,000 students enrolled in 14 schools and divisions within New York City. In addition, NYU maintains overseas programs in approximately 27 foreign countries around the world. The Program in Business Education, begun in 1926, is housed in the School of Education, Health, Nursing and Arts Professions (SEHNAP) under the Department of Curriculum and Instruction. The Program consists of three full-time faculty members administering to the needs of undergraduate students in five curricular areas and graduate students at the Master's, 6th-Year, and doctoral levels. The Business Education graduate program is currently the largest such graduate program in the nation and prepares students for both teaching and non-teaching roles in a variety of academic and business settings.

One of the emerging curricula developed by the Program involves Office Systems at both the undergraduate and graduate levels. The undergraduate Office Systems curriculum, recently approved by the New York State Education Department, is the first Office Systems Research Association's (OSRA) curriculum in force in the country and prepares students for office systems careers in business and industry. The graduate concentration in office systems encompasses personnel in both academic and business environments.

Because of the emergence of office systems as a career, a strong leadership role was taken by the Program, developing the 10-course major at the undergraduate level and a like number of courses at the graduate level in order to meet this need. The involvement of technology, obviously, was a critical ingredient to this end.

THE NEED

The curriculum does not involve technical training on equipment. When the decision was made to address the LAN question, only one IBM personal computer was in use by the faculty--obtained through an earlier research grant--and a dedicated word processor. The interest, however, in the emerging curricula as well as the felt need to utilize this technology at the faculty level prompted the three faculty to mutually investigate the acquisition of personal computers for individual use. All three faculty took advantage of NYU's volume buying arrangement through IBM and purchased their own individual personal computers for home use, thereby enhancing their involvement with the PC technology.

Thus, faculty involvement in the technology as well as the emerging office systems curricula encouraged the consideration of additional purchases of PCs for their respective offices. However, since one of the faculty members had his office on the 6th floor and the two remaining faculty members had their offices and files--as well as the Program secretary--housed on the 5th floor, communications were, at best, difficult. And, because student advisement involved document files, students had to

visit the 5th floor first, obtain their confidential files, move to the 6th floor for advisement, and then return to the 5th floor to replace their files. Appointments, likewise, suffered the same constraints and when the Program secretary was out of the office or occupied elsewhere, communications between floors were difficult at best. It is easy to see the need, therefore, for a solution to this as well as other related problems.

The solution, of course, was the decision to acquire a LAN.

THE SUPPORT

Naturally, support for such a purchase does not come easily in today's academic environment. This is even more critical in a school where enrollments and budget constraints negatively impact upon such requests for support unless they can be determined to be in the long-term interests of enhanced enrollments and income.

However, with Program faculty support and professional enthusiasm made obvious to all who would listen, the Dean of the School agreed to the concept, offering some financial as well as personal encouragement. The first--financial support--was, out of budget constraint necessity, minimal. A $5,000 sum was made available along with support for installation and maintenance on an 'as needed' basis. Maintenance contracts were not considered.

The second--personal encouragement--was more helpful. Support via increased and enhanced communications between Program faculty and administration opened lines of communication and fostered increased rapport between these two academic areas. The LAN tied nicely with other technology projects of the School, including interactive videodisks, telecommunications, and an electronics media facility. The LAN, however, was essentially a Program activity with no interface with any of the foregoing projects.

The major source of funding came through the Peter L. Agnew Foundation who's administration enthusiastically supported the project from the beginning. The funds provided through the Foundation via gifts to the Peter L. Agnew Business Education Fund, allowed the faculty to purchase the necessary equipment and software to implement the LAN.

THE DECISION

The decision to purchase the LAN having been made, work began in earnest. Decisions concerning the nature of the equipment, the wiring involved, the type of LAN to be involved, and the support elements took considerable time and effort. These efforts were encouraged by the faculty's completion of a research project concerning LAN implementation in business settings in the United States. The results of this research project provided not only needed information for the LAN decision-making process, but also provided the substance for an article, "The Local Area Network: Is it Living Up to Early Expectations?" scheduled for publication in the Journal of Systems Management.

The search culminated in the decision to acquire the IBM Token-Ring Network, shortly after its announcement in October of 1985. Obviously, wheels turn very slowly in academe, but they DO turn!

IBM was selected for a number of reasons. Firstly, the Program wanted to standardize on one type of personal computer and since faculty had the IBM PC at home as well as one in the office, it was a logical conculsion. In addition, since the University had a purchase agreement with IBM, customer service support was more assured. While the latter was not exactly forthcoming, some support was provided, especially during the installation period. Second, the Token-Ring software and the NETBIOS program were, at that time, the state of the art for small LANs, and since there were to be only four nodes on this LAN, larger network programs tied to mainframes or minicomputers were not appropriate. Third, if any problems were to develop, a single-system operation would assist in identifying the problem more easily than if multiple components were involved. The decision was then made to acquire two IBM PC/XTs with dual floppy disk drives, enhanced to 640K, and one IBM PC/AT with a 30MB hard disk drive, one 1.2MB floppy disk drive and one 360K floppy disk drive. Three IBM Proprinters were also acquired to fill out the printer requirements. Color monitors were purchased for all PCs with the AT tied to an enhanced color monitor.

The faculty member on the 6th floor received the original PC, now enhanced to 640K, along with a Proprinter for his use; one faculty member on the 5th floor received the XT for his use--with the then-existing IBM Wheelprinter to serve his printing needs; and the other XT was placed at the Program's secretarial workstation with a Proprinter for her use. The AT went to the remaining faculty member on the 5th floor along with a Proprinter for her use. The AT was to serve as the LAN file server and the 'traffic cop'--the multistation access unit connecting the individual PCs--was placed with the AT in that location. All units were hard wired, and this was completed using Type #9 wire over one Saturday during the 1986 Christmas holiday period. The wiring connected the 6th floor PC with the three units on the 5th floor. All wiring was run through the dropped ceiling and down the wall nearest the PC involved.

THE INSTALLATION

While the wiring installation went smoothly, as each faculty member had previously indicated the placement of the LAN unit, the connector installation was less smooth. Attribute it to the relative newness of the Token-Ring Network or whatever, the local IBM people remained on site for a total of more than a dozen working days to install the connectors, figure out the AT hard disk configuration, and install the LAN program. Testing and debugging took another couple of days of intensive work. IBM's assertion that the LAN is 'user installed' was overly optimistic.

Security was another problem that had to be solved. Since

the secretarial workstation was in a large open office area, shared by four other secretarial stations, security was a consideration. And after rejecting a number of alternatives--most of them cumbersome, tedious, and expensive--a 'fortress' was purchased for the keyboard and disk drives and mounted to the workstation desk. The printer was fastened by a simple wired lock to the edge of the desk, and the monitor was mounted atop the 'fortress' and secured from below according to the supplier's instructions.

THE TRAINING

As with most cases, PC training relied heavily on self-taught methods. The manuals were read, of course, but a complete understanding of them is still an ongoing process even at this date. Most of the faculty, happily, are quick studies, and with a certain amount of logic, have finally 'locked in' the basics. Everyone has participated in courses offered through the University's various departments; however, these courses have centered primarily on applications involving the use of word processing packages, data bases, and spread sheets. IBM has also offered training; however, at this point, little has been forthcoming. The most effective training appears to have come from the traditional 'trial and error' and from the local 'help keys'--other faculty and staff members involved in some aspects of the applications processes. Training--the weakest link in any technology installation--still remains as the weakest link. However, as all become more familiar and proficient in the LAN applications, colleagues and administrators are made aware of each new development and the interest continues to grow.

THE USES

At the present time, the uses of the LAN technology center primarily on stand-alone applications. Word processing for each faculty member is generally conducted individually and shared periphials await the delivery of a laser printer. Classroom demonstrations have been conducted as illustrative of the LAN; however, the primary function of the LAN was not to be instructional, but administrative and academic. LAN activities at present include electronic mail and messaging--both individual and broadcast--and project manuscript sharing. The latter involves a Program project where report manuscript is produced on each invidual PC and shipped to the AT for storage. It is then available for individual faculty members to recall for additions and editing. In addition to this type of file management, dBase III Plus provides the wherewithall for student records and advisement activities. The 6th floor faculty member can now access doctoral students' files through his PC as he advises students individually. He can also edit files accordingly and provide update notes where appropriate. Mailing lists can be created in a number of ways, including mailings to current students, announcements of special events to graduates, and mailings to students in different programs and locations--metropolitan/statewide/off

campus/ and our Puerto Rico Residence Center in San Juan, Puerto Rico.

Student records are primarily managed by the secretary, who updates records individually upon receipt of the semester transcripts. Address changes, modifications in course requirements and committee membership revisions are also managed by her; however, any faculty member may edit any file as the need arises. File security is accomplished through a simple password and to date, this has proven effective.

In the near future, the LAN is planned to be used for electronic calendaring (scheduling two-hour committee meetings with students and their committees sometimes involved 30-40 telephone calls with a two-month delay), appointment and room scheduling, electronic bulletin board use, shared storyboard programs for class lectures, articles, and presentations, and statistical programs for research activities. Stored programs on the AT--such as Word Perfect and dBase III Plus--can be downloaded to each faculty's respective PC as needed, and upon receipt of the laser printer, printing tasks will also be assigned. With the recent purchase of a modem and a subscription to the Source, faculty will be able to conduct 'on line' research. It is anticipated that by the time of this presentation in California, many of the foregoing 'future' tasks will have already been accomplished.

THE OUTCOMES

Simply, the LAN has changed the way we operate! We share more resources and find new ways of handling technological problems. We produce more (quantity) and better (quality) work. We are more current in our respective fields, and our sharing of information occurs on a more timely basis. Finally, we are more creative. New ways have been found to handle old tasks and new uses developed for new activities. As a result, our efficiency as well as our respective productivity has increased. Our secretary? Her job, too, has changed as she has become more managerial due to the changed tasks, is more creative and innovative in dealing with technology, and more efficient in her output. She has joined in our enthusiasm for the technology, and she has involved herself in the training of graduate assistants, interns, and work study students on the PCs as well as the LAN. During her 7-year service with our Program she has always been a member of the team; however, she has now moved into a more creative role--a technological peer, so to speak.

THE FUTURE

Among the LAN-oriented activities foreseen, the following are most prominent: additional printers and plotters shared not only through the current AT but through the upgraded secretarial workstation XT; a wider LAN connecting Departmental faculty and staff on separate floors of the same building, which will involve additional nodes to the existing LAN; and at least one and

possibly two LAN gateways connecting School and University resources. School gateways would connect budget and finance offices, graduate admissions, and academic affairs; University gateways would involve other Departments and other Schools, undergraduate admissions, and the academic computing center as well as connecting us to the existing on-line library resources. Finally, a School-expanding to a University-wide electronic bulletin board is a distinct possibility in addition to extending us into satellite networks linking other universities and institutions. This latter phase could also deliver instruction to the University's many residence centers worldwide as well as link like-minded institutional interests such as may be seen with the 94-member Holmes Group, Ford and Carnegie Foundations, and UNESCO activities.

SUMMARY AND CONCLUSIONS

The selection of a LAN for an academic environment offers a wide variety of challenges, the selection of a LAN itself notwithstanding. While the return on investment is not measurable in terms of dollars and cents, it can be measured in terms of faculty performance and research applications. However, with limited budgets, faculty need to use creative financing--research grants, alumni contributions, etc.--in order to acquire the needed resources. Since faculty are not considered technological experts, LAN functions are largely application-oriented with individual concerns overriding more global aspects. The LAN is a tool only as effective as the faculty that use it, and technological performance is limited only by the imagination and creativity of the individual. While the LAN can assist faculty in a number of arenas--academic, research, and instruction--it is in its educational infancy with a great deal of growth ahead of it. What it can do in the future, is yet to be ascertained.

The opportunities for 'local' area networks in an academic setting need not remain at the 'local' institutional level or within the 'local' area network. The many opportunities for sharing information are limited only by the enthusiasm and imagination of the individual. The desire to communicate has always been the first step in the process; the LAN technology provides the wherewithall to enhance this communication effectiveness.

# # # # # #

OFFICE AUTOMATION RESOURCE CENTER

EDNA CHISENA, DIRECTOR
TRIDENT TECHNICAL COLLEGE

The Office Automation Center of Trident Technical College opened its doors to the public in July 1987. The Center functions under the Continuing Education Division of the two-year community college. The purpose of the Center is to serve as a resource for local businesses and students of the college by:

(1) Providing area business and industry with a model office environment equipped with brands of automated equipment representative of current trends in office automation, ergonomically sound furnishings and atmosphere, and current information on software

(2) Offering an informational resource for business and industry on office layout and equipment selection as well as customized training for personnel

(3) Providing students with practical work experience in a realistic office setting

(4) Offering a schedule of seminars and workshops related to office automation for both the general public and specifically targeted audiences.

Evaluations of client interest and market trends in the local area determine which products are featured in the Center. The products are sponsored by local vendors for a temporary period of time, ensuring availability of the latest technology. Among the items currently on display are personal computers, application software, furnishings, printers, communication systems, copiers, filing systems, and electronic typewriters.

Anyone interested in learning more about office automation is welcome to visit the Center. There is no charge for using the Center to evaluate products on display or gain information to help assess office automation needs. Competitive rates are charged for customized training and are the only source of revenue resulting from the operation of the Center.

ASSESSMENT OF NEEDS

One of the problems faced by educators for decades has been providing adequate training with limited budgets. Trident Technical College offers its students advantages that many schools only dream of. In 1984, a computer center dedicated to student use containing four classrooms and an open lab area of over 125 microcomputers was established for use by all academic departments. It didn't take long, however, to realize that the technology being developed in the area of office automation was changing so rapidly that

we couldn't possibly offer educational opportunities on many of the products being used in the business community.

After several months of casual observation, several items began to emerge as community needs:

(1) Students could gain valuable experience by having a resource room to try out office automation equipment and train on specific software they may encounter in the workplace. The more exposure they have to various products, the more effective job placement is.

(2) The community at large was very interested in learning specific computer-related skills without being locked into a structured curriculum program. This indicated that flexible training programs that accommodated individuals, groups of all sizes, and special interest groups would be received well.

(3) Customers of other computer training programs in the area were not always satisfied with the introduction and training support they were receiving upon purchasing new equipment. Much of this training was being done by sales or technical personnel and did not relate well to the problems of the end user.

(4) The public needed an environment free of sales pressure to become "computer literate." This involved assisting people who would eventually find themselves in a sales environment trying to ask the correct questions and make the right purchasing decision.

These observations led to a written proposal submitted to college administration for an Office Automation Resource Center. The plan included offering a unique facility to the community that brought technology, resources and educational opportunity together in an attractive environment.

STRATEGY FOR IMPLEMENTATION

After the proposal was approved on the institutional level, efforts began to evaluate the trends of office automation in the community. This was accomplished through on-site visits with equipment vendors. Information gathered during these visits became valuable in determining what the resource center should contain in order to be adequate at meeting the needs mentioned earlier.

Data collected from each vendor included products carried by them, which products sold more frequently, how training was handled by their company, and general feedback on how they felt a resource center of this type could benefit their business. Once these initial contacts were made, careful evaluation of the collected data resulted in certain products emerging as leaders in sales in each of the categories of equipment and software. The major categories being considered were personal computers, printers, electronic typewriters, word processors, copiers, facsimile and furniture. Software categories included word processing, spreadsheet, data base, communications and graphics.

From the collected data, a list was compiled outlining what model offices should contain in order to cover the wide spectrum of office automation and also feature what was representative of the local business community. Vendor sources for each of the products were identified, as well as a secondary source for each product. Careful

consideration was given in order to keep technology from being duplicated.

Meanwhile a design specialist from a local vendor agreed to offer her services to make the facility attractive and workable. The final floor plan included an open work area containing six model offices, a reception area, reference section, training room for larger groups, and a conference room. The plan also included a lounge for preparing refreshments and several staff offices. Herman Miller Ethospace was chosen as the furnishings to be used in all areas of the Center. The college made a financial commitment to renovate the space allocated for the Center and provide all furniture. A special discount was negotiated for the furniture since the Center serves as a showcase for the product. All other equipment and software are vendor sponsored.

Vendors were again approached with the resource center concept--only this time they were given drawings of the proposed center, a contract outlining the details of displaying products, and suggestions on what they might feature in the complex.

The arrangement allows the college to have full use of the products on display for training and demonstration purposes. Sponsoring vendors are responsible for training Center staff to use the products. Commitments are made for six months, with the option to renew with the consent of both parties. The vendors, in turn, may use the facility as a showroom.

MARKETING THE CENTER

The Center had its grand opening in July of this year. Announcements were sent to area business people and the general public was informed by advertising in local newspapers.

The ongoing marketing plan for the Center includes publishing a bi-monthly newsletter that briefly describes changes that are taking place and upcoming events. A series of workshops and seminars are also offered ranging from general public interests to vertical market topics. Flexibility and unique opportunities for learning are the keys to keeping the Center from becoming stale. Many of the seminars are designed with managers as the primary targeted audience. A series of productivity workshops as well as evaluation seminars aimed at helping the consumer make sound purchasing decisions are examples.

Interest in the Center has been expressed by a wide variety of clients ranging from the small independent business owner to Chief Executive Officers of large corporations. Each client is requested to complete a questionnaire during visits. This information is used to supply vendors with feedback and to determine the changing interests and needs of the local community. One of the more extensive projects of the Center has been a joint effort with the local Chamber of Commerce to facilitate the automation of their offices. A task force comprised of college and Chamber staff conducted a needs assessment to determine the primary areas to benefit from automation. Once the needs were identified and prioritized, a generic list of recommended equipment was presented to local vendors for bidding purposes. Training on the equipment and software is done through the college's various training programs.

The opening of the Center proved to be a very successful showing of leading hardware and software vendors. Each

vendor is given recognition in printed materials about the Center and a special plaque indicating their contributions is located near their display. Among the items on display are:

AMTEL messenger center

APPLE MacIntosh SE

AT&T personal computers, telephones, typewriter, scanner, printers

CANON micrographics

IBM typewriters, personal computers, printers

INTIMUS paper shredder

KONICA ROYAL desktop copier

MINOLTA full color copier

MOTOROLA alpha numeric pagers

NEC alpha numeric pages

NORCOM dictation systems

PANASONIC typewriters, telephones

SAVIN desktop color copier

SHARP facsimile

UNITED TELESPECTRUM cellular phones

WANG OIS system

XEROX telecopier facsimile, typewriters

Among the software being featured are various word processing, spreadsheet, data base management, computer-aided design, desktop publishing, accounting, medical and legal office assistant, and graphics packages.

Vendors are encouraged to use the Center as a showroom and to leave promotional literature among their displays. Sales are absolutely forbidden within the Center.

BENEFITS TO THE COMMUNITY

The overall benefit to the college is an opportunity to become a leader in the field of office automation. Students are given the advantage of being allowed to use the Center as a resource during their studies. All advanced Secretarial Science students are required to successfully complete a working practicum in the facility before graduating with an associate degree. Plans are underway for students of the Paralegal program to complete projects using the legal accounting software available in the Center.

The Center is located in the downtown campus of the college. The area itself is located in both a socially and economically depressed area of the city. The renovation of an abandoned high school has resulted in a beautiful campus convenient to inner-city residents who before had difficulty transporting themselves to one of the other two campuses. Since the commitment of Trident Technical College to develop this campus, much progress has been made in redevelopment of the surrounding area.

This project is an exciting and innovative approach to computer training. The enthusiasm is evidenced by participating vendors, educators, and clients alike. The rewards for the college as well as the participating community are numerous, some yet to be discovered.

7/87

PERSONAL COMPUTER-BASED PROJECT MANAGEMENT USING VIEWPOINT

Edwin E. Blanks, CSP, Professor, Virginia Commonwealth University,
Richmond, Virginia 23284

ABSTRACT

This paper addresses the topic area of Management Information Systems using a project management software package on a personal computer. The recent flurry of excitement around the new workbench tools makes this an exciting subject to the Information Systems students. This systems planning course utilizes a well designed project management software program called **Viewpoint.** Viewpoint consists of five major components and is menu driven with very powerful special features. Using a personal computer software approach rather than the mainframe, students have the opportunity to plan and control a major systems project with 150 activities. This paper is proposed for the track on Innovative Teaching Methods II, CIS elective area, MIS and Information Resource Planning.

Computer-based project management systems run the spectrum from straightforward, menu-driven programs for personal computers to complex, highly versatile programs for the minis and mainframes. Personal computer-based project management software is popular today because it is not expensive and it is very easy to operate.

There is a wide array of products available in the market and it is a difficult task to select the best product. The first step is to understand several of the basics of project management. Project management is the scheduling, planning, and tracking of a group of activities that function together in achieving a specific goal. Projects generally must meet agreed upon requirements and limitations on time, cost, personnel, and resources. Project management is composed of two major processes, namely, project planning and project control. Project planning is actually planning and scheduling work before it has begun. Project control is the process of statusing and replanning work while it is in progress.
Before deciding on a specific project management software package, other important criteria should also be examined. Issues related to security, response time, project complexity, standard reports, and network diagramming must be addressed. With over 100 project management software packages available for micro computers, making the right decision can be difficult.

With this preliminary work completed, the Information Systems Department at Virginia Commonwealth University in Richmond, Virginia selected the package known as **Viewpoint.** Viewpoint is a product of Computer Aided Management in San Rafael, California. This package combines user-oriented design and advanced project management features to make the analytical capabilities of project management software usable and easy to learn.

The software consists of many valuable features, including a relational database and spreadsheet like format for table editing and formula evaluation. Several of the features designed for ease of use are:

- Menu driven (See Figure 1)

- Pop-up forms
- Context sensitive help
- Dynamic use of colors
- Documentation with tutorial
- Optional mouse support

To use Viewpoint you point with the cursor and fill out blanks on the forms shown on the monitor. Please note there are no paper forms to be completed as input to this software. However, standard printed reports are available through the Report Writer which also has capabilities for tailoring reports to your specific needs.

Menu choices are explained in a constant mini-help window. To activate a choice or to find out more about something, you point to it using the cursor. Also if you are filling in a form on the monitor and standardized multiple choices exist, a choice table will appear. When you point to an item in the table it will move it into the form. If, for some reason, completing a transaction becomes confusing, pressing the escape key will undo what has just been entered.

Viewpoint is simply a tool to aid the student in performing project management functions more accurately and quickly than could be handled manually. Performing project management functions manually dictates voluminous amounts of paper. Using Viewpoint the student eliminates the paper and substitutes for it a time lined CRT screen known as the Planning Screen. Activities and events are specified on the screen and relationships are represented between them.

The concept of project planning using Viewpoint consists of five sequential steps which are:

(1) Identify goals (important target dates)

(2) Break down the project into separate activities

(3) Connect these activities together in time - thus generating a logical network

(4) Assign resources and refine time and cost estimates

(5) Balance costs, resources, and outcomes.

It is always important before covering a tremendous amount of material on any personal computer (PC) package to be certain the student has a foundation of the basic terminology. Project management is no different than many other subjects. It has its unique jargon. Some of these terms are defined as you begin to tackle that appropriate subject matter on the PC. As an example, some project management terms are:

Activity	Something that consumes resources - Viewpoint shows this as a horizontal bar
Event	Point in time, a date - Viewpoint shows this as an arrowhead
Critical Path	The path of the workflow that directly effects the finish date of a project - Viewpoint shows this as a red line
Float	This is thought of as slippage, that is shifting an activity without effecting the start or finish dates - Viewpoint will color these shifts.

There are many other terms which are introduced at appropriate points as the student plans, builds, and controls the project.

The Viewpoint system characteristics and capabilities covered during the project management class are:

- Generation of CPM and Gantt Charts (See Figure 2)
- Progress reporting
- Automatic loop detection
- Use of flexible calendar
- Resource and Cost leveling and constraining

A software package like Viewpoint enables the student to schedule the activities in a project and allocate the required human and material resources. This program allows the student to also have the ability to send data to other programs (export) and receive data from programs (import). Once the plan is produced the student can track the progress as changes in resources and requirements are provided to the student.

This is a semester course and the student works on two individual projects and one team project. This allows each student to be responsible for his/her own management of a project along with the responsibility for functioning as a project team member.

While the availability of personal computer hardware might present a problem in some institutions, here at VCU we have an adequate number of PCs to support this class. The Viewpoint package requires an IBM PC-XT or compatible with 512k bytes of main memory, color monitor, and optional mouse. A variety of printers are supported and dot matrix such as Epson FX and MX are the most popular.

Despite the recent flurry of excitement about PC-based project management software, many of the techniques are time tested. Providing students the opportunity to evaluate projects based on resource relationships and dependencies based on task relationships was very difficult to accomplish using manual methods. The use of a powerful tool such as Viewpoint allows the student the opportunity to visualize immediately task schedules, networks, and relationships. Student productivity is enhanced while they minimize the time and money spent in completing a project.

References

1. Claude W. Burrill and Leon W. Ellsworth, Modern Project Management, Tenafly, N. J., Data Processing Handbook Series, Burrill-Ellsworth Publishing Associates.
2. John Kador, Looking for Project Management Software, Information Center, Volume II, No. 10, October 1986, pp. 54, 56, 57-60.
3. Viewpoint, Computer Aided Management, Inc., 1986.

ViewPoint RoadMap

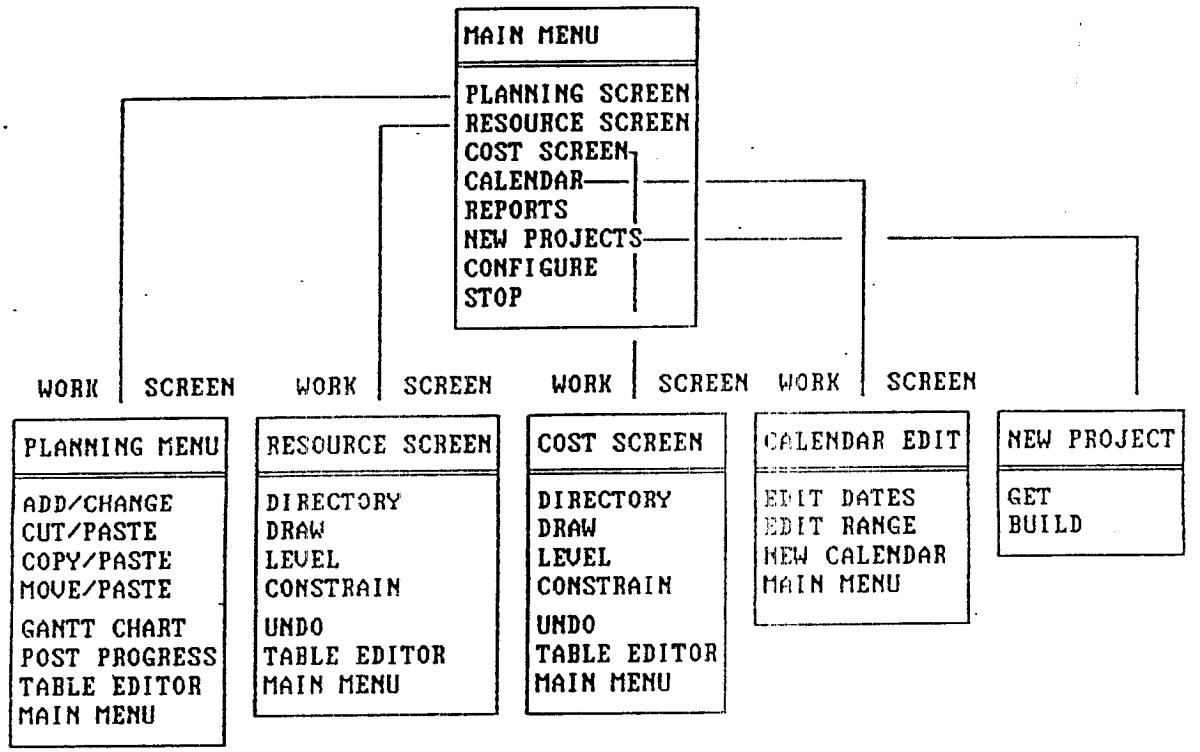


TABLE EDITORS

Planning Screen
- Schedule
- Progress
- Resource
- Cost
- Activity Codes

Resource Screen

Cost Screen

Figure 1

UNDERSTANDING AND TEACHING THE INFORMATION SYSTEM IMPLEMENTATION PROCESS

J. C. Vail
Assistant Professor
College of Business Administration
Loyola University
New Orleans, Louisiana

Defining implementation as the bridge between system development and steady-state operation, this paper presents a three-part hypothesis for researching as well as teaching I.S. implementation. Separate diagnostic methodologies establish criteria for defining a successful implementation effort, identifying the factors which affect the probability of success, and classifying research findings. Adaptation and use of this material in the undergraduate and graduate curricula are described.

There is a rapidly growing research base concerned with the relatively narrow subject of implementing computer-based information systems (1,2). Interestingly, however, these findings receive little attention in some widely used texts (3,4,5).* At the Loyola (New Orleans) College of Business Administration, we have embarked on a research program to try to distill and supplement the findings in this area, and to develop ways of conveying useful insights at both the graduate and undergraduate levels.

One encounters no shortage of empirical "horror stories" in this field. The problem seems rooted in the observable fact that people in line management (and even in DP management) do not implement systems very often. And almost by definition, they do not repetitively implement the same systems in the same environment. Thus there is little natural accumulation of learning applicable to the next time; if a successful implementation is the goal, heuristic processes are not much help.

IMPLEMENTATION DEFINED

We define I.S. implementation as the bridge between system development (including design and construction, and/or procurement) and steady-state operation. This definition applies to the introduction of wholly new systems as well as to the conversion of existing computer-based systems. Further, this definition is completely independent of size; the concepts presented here are as applicable to the first introduction of a single-station PC in a small business setting as they are to a major system conversion in a main-frame environment.

EMERGING TRENDS AND INFLUENCES

The study of, and, more particularly,

* A notable exception is Leigh and Burgess (6).

the need for teaching, I.S. implementation is affected by two principal factors. The first reality is that implementation of information systems is no longer only the DP manager's problem. The expansion of end-user involvement continues; information systems are being linked to suppliers and customers; and systems development is increasingly an integral part of corporate strategy and competitive advantage.

The second (equally obvious) reality is that I.S. technology is changing rapidly. The economics of computing and telecommunications are presenting new opportunities to both large and small users, spawning entirely new applications and wholly different approaches in areas like desktop publishing, computer-integrated manufacturing, and education itself (7).

A HYPOTHESIS IN THREE PARTS

At Loyola we have formulated the following propositions:

(1) that current findings in this field, generally based on larger systems, may not be entirely applicable to smaller systems or newer uses of I.S. technology

(2) that technological developments present new alternatives for the design of the implementation approach

(3) that there is a largely unfilled need to translate research results in this area into a usable and understandable framework for the general manager.

In pursuit of such a framework, we have defined and are developing the following diagnostic methodologies.

DIAGNOSTIC METHODOLOGY I: DEFINING SUCCESS

Even in a rigorous academic context, we try to be quite clear about our orientation -- we are interested in real-world success. The concept of success, however, needs to be carefully differentiated. We define a successful I.S. project as one which (1) provides the anticipated functionality, (2) provides the anticipated economic and other benefits, (3) meets technical performance specifications, and (4) interfaces correctly with other systems.

On the other hand, we define a successful I.S. project implementation as one which (1) comes in on cost, (2) comes in on schedule, and (3) elicits user involvement and support.

Obviously it is possible to have a successful implementation of an unsuccessful project, and vice versa.

DIAGNOSTIC METHODOLOGIES II: CATEGORIZING RELEVANT FACTORS

We have chosen to draw from the Cash, McFarlan, and McKenney taxonomy in identifying, and adapting to the special topic of implementation, factors which affect the probability of success. These factors fall into two broad groupings: characteristics of the project itself, and characteristics of management (both people and process).

Project characteristics include:

(1) Company-relative technology. This is a short-hand phrase for describing the degree to which the technology (whether hardware, operating systems, applications, or telecommunications) is new to the particular company, or subunit thereof, into which it is being introduced. The probability of both project success and

implementation success varies inversely with the degree of "newness."

(2) Size of project. Project "size" is a function of hardware configuration, software complexity, time to develop, time to implement, number of organizational units involved, number of individuals involved, and other measures. The larger the project, the lower the probability of success.

(3) Stability of project. Applied to the development cycle, the concept of project stability refers to the degree to which project scope and requirements specifications change in midstream. Applied for our purposes to the implementation process, project stability encompasses both that which is being introduced and the environment into which it is being introduced. To the degree that either is unstable, the probability of success declines.

Cash, et al have developed extensive questionnaires to measure and quantify company-relative technology, project size, and project stability (7). We are developing and preparing to field-test similar questionnaires focused on the implementation process.

Management characteristics include:

(1) commitment and talent

(2) external and internal integration efforts

(3) planning tools in use

(4) project control mechanisms in use.

These characteristics are harder to quantify but obviously no less important than project characteristics in determining the success of an implementation effort. Indeed, our preliminary findings indicate that independently of all other factors, user involvement and management commitment alone often make the difference between success and disaster (2).

DIAGNOSTIC METHODOLOGIES III: CLASSIFYING FINDINGS

However elegant one's observations in the area of I.S. implementation, we believe it important to be cautious before supposing that perfectly valid findings from one implementation study are of any value whatever in situation n + 1. In other words, the honored maxim of replicability is especially elusive here.

To attack this problem explicitly, we have developed the matrix at Figure 1. Implementation findings are conceptually plotted along two axes: the "uniqueness" dimension, and the "permanence" dimension.

The uniqueness dimension is a way of forcing the analyst to evaluate the placement of a particular finding on a scale between "generic" and "unique." We believe there are relatively few findings in this field which can be safely assumed to apply to dissimilar situations. (The user involvement and management commitment examples cited above are clearly two.) The analyst should go further than this simple classification, of course; his burden is to identify the project and environmental factors which, if present in the next situation, would make an assumption of applicability valid. But the first order of wisdom is the recognition that few findings in his research are universal.

The permanence dimension in Figure 1 suggests a further classification of findings. Independently of whether generic or unique, a finding may be more or less enduring over time. One reason for this perishability of

findings is technological advance; another (or a subset) may be advances, including AI, in the man-machine interface.

TEACHING CONCERNS AT THE UNDERGRADUATE LEVEL

In the undergraduate M.I.S. course at Loyola, we explain the above concepts, with numerous examples, primarily as a way of enhancing awareness of the likely impact of an implementation process on the user. Although the employee in a junior general management position may not be in a position to select an implementation strategy or to direct the effort, our premise is that as a user the recent graduate can, with appropriate insights, materially contribute to a successful outcome.

TEACHING CONCERNS AT THE GRADUATE LEVEL

Our graduate systems course, in contrast, delves far more deeply into the topic of I.S. implementation. We cover the literature in some detail, with emphasis on alternative implementation strategies, selection of an appropriate strategy, and management control of the implementation process. We try to cover the same situations and issues from the perspective of the senior general manager, the I.S. manager, and the user. At this level we find that virtually all our students have had experiences--often painful--with introductions or conversions of computer-based systems. The discussion is invariably rich and lively.

To supplement the literature and our own work, we conduct an ongoing case development program focused on implementation experiences in local concerns. Cases developed to date are used principally in our graduate course; other cases more appropriate to the undergraduate level are planned.

CONCLUSION

We are convinced that at both undergraduate and graduate levels there is a need to deal explicitly with the topic of I.S. implementation. The methodologies outlined above seem to provide a useful framework for this purpose.

REFERENCES

1. F. Warren McFarlan (ed.). The Information Systems Research Challenge. Boston: Harvard Business School Press, 1984.

2. H. Wayne Hunt. "Management Information System Implementation Literature: Annotated Bibliography." (Unpublished ms, Loyola University, 1986.) I am indebted to Mr. Hunt, a graduate student, for much of the search work we use and benefit from at Loyola.

3. James O. Hicks, Jr. Information Systems in Business. St. Paul, MN: West Publishing Co., 1986.

4. Donald H. Sanders. Computers Today. New York: McGraw-Hill Book Co., 1985.

5. David M. Kroenke and Kathleen A. Dolan. Business Computer Systems. 3rd ed. Santa Cruz, CA: Mitchell Publishing Co., 1987.

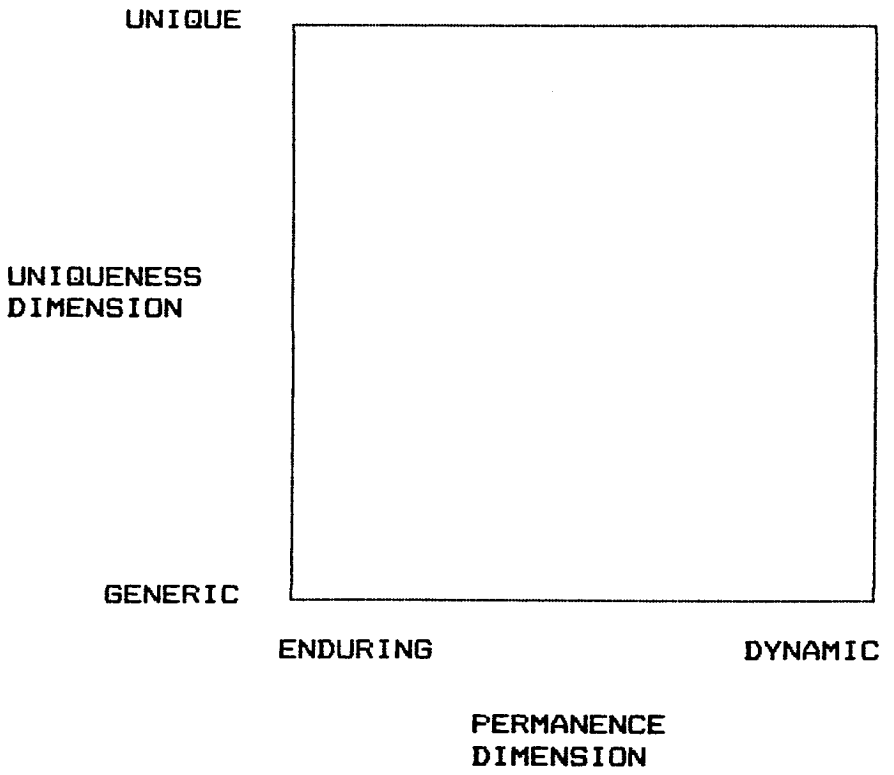
6. William E. Leigh and Clifford Burgess. Distributed Intelligence. Cincinnati: South-Western Publishing Co., 1987.

7. James I. Cash, F. Warren McFarlan, and James L. McKenney. Corporate Information Systems Management: Text and Cases. Homewood, IL: Richard D. Irwin, Inc., 1983.

Exhibit 1

DIAGNOSTIC METHODOLOGIES -- III

CLASSIFYING FINDINGS:



Teaching Database Management Systems Using a Case Study Project Implemented by dBASE III

By

William J. Dorin

Instructor, Purdue University Calumet

Hammond, Indiana

Track: Innovative Teaching Methods

Abstract

A case study project is used to a teach database management systems course. The problem in the case study is implemented by the students, using dBASE III Plus. This software package allows the students to see a direct relationship between the data modeling and the normalization techniques taught in the course. The easy-to-use and user friendly dBASE III Plus software allows the students to program a substantial part of the system within a limited time frame.

"Management is the creation and maintenance of useful and productive order." (1)

INTRODUCTION

As an organization's data needs grow, traditional file processing becomes ineffective. To manage the vast amounts of data efficiently a database management system (DBMS) is needed by the organization to manage the data. A DBMS is a "software package through which users interact with a database." (2) Effective manipulating and processing is necessary to provide information to different levels of management in an accurate and timely fashion for decision making.

The purpose of this paper is to discuss teaching a database management systems course using dBASE III Plus to implement a case study problem.

THE COURSE

In the DPMA model curriculum, course CIS 86-6 is Files and Databases. At Purdue University Calumet in Hammond, Indiana, we offer course CIS 353 Database Management Systems. The purpose of this course is to introduce students to logical and physical database structures, database design with data modeling and normalization, design objectives, the role of the database administrator, and commercial database management systems, including data dictionaries. The hierarchical, network, and relational database models are discussed in relating the design to the various commercial databases.

THE CASE STUDY PROJECT

Case studies are descriptions and simulations of real world problems. (3) Case studies provide the student with a problem that can be solved

using a computer. The problem is described as the difference between desired and actual outputs of the current system. The case studies used for CIS 353 describe an organization in its entirety with a focus on the sub-units that are affected by the problem.

The students' assignment is to design a system using the data modeling techniques discussed in lecture and the textbook. The design must take into consideration the entire organization, including any sub-units that will utilize the database. The students are divided into project groups of three or four members depending on the size of the class.

The problem solution design is to be completed in phases, and the scope of the project is demanding but not impossible to finish within a nine-or ten-week period. Included in the scope are the data modeling, building the data dictionary, deciding user views, and programming specific aspects of the systems. These programming tasks include reports, display screens, and file maintenance.

IMPLEMENTING THE SYSTEM

RELATIONAL DATA MODEL

The first phase of implementing the system design is data modeling. Before students can begin modeling, they must know the difference between an entity and an attribute. This terminology is defined, and emphasis is made at this point that modeling examines logical relationships and not physical data.

The main problem encountered in teaching students data modeling is that during the first two years at Purdue University Calumet, traditional file processing is taught. Now they must learn to think of logical relationships and shared data files. A number of examples are used to show data in its unnormalized form and to explain how to proceed to the third normal form. Students are then given exercises to try on their own.

These exercises teach students to identify attributes that are functionally dependent on other attributes. Through these exercises, students learn to identify the different anomalies: update, inconsistent data, additions, and deletions. In addition, the concepts of primary, secondary, and foreign keys are discussed in the normalization lectures and exercises.

Once each entity has been defined, the relationships between entities are determined. These relationships are graphically represented through the use of Bachman Diagrams or enterprise schema design model diagrams. To reinforce these skills, students are given a number of exercises for practice to learn the normalization techniques.

To apply these techniques to the case study problem, a partial class period is used to help students brainstorm about the data requirements for the entire organization. Students are encouraged to offer suggestions, which are then listed on the board. Nothing is noted as correct or incorrect immediately. After all possible items have been suggested, The definitions of entity and attribute are repeated, and the students are asked to separate entities and attributes from the items.

MAPPING TO A DATABASE MANAGEMENT SYSTEM

Once this process is complete, students begin to map the design onto the database. Because of the physical restraints of dBASE III Plus, students create separate

database files for each relationship. Each file has its own structure or schema which allows students to see a direct correlation to the normalization process and to the design of the database. In addition, students use dBASE III Plus to build a data dictionary.

ADVANTAGES TO DBASE III PLUS

A number of advantages exist for using dBASE III Plus in this course. First, it is one of the most popular personal computer database packages available. It is likely that students could encounter the software on the job.

Second, compared to Knowledge Man and R:Base 5000, dBASE III Plus allows a logical join of files. A logical join means that two files may be joined temporarily in memory without creating a new database. This join is a logical relationship between two files using the foreign key concept taught in normalization. In the logical join, the two files are opened and accessible at the same time, allowing updates to fields in either or both files, and a new physical file does not have to be created.

DISADVANTAGES TO DBASE III PLUS

One major disadvantage to dBASE III Plus is it does not provide for use of tables. R:Base 5000 allows for tables to be built that are accessible by column and row. However, it must be pointed out that these tables are data files, and not the subscripted variables used by arrays in other high level languages. (4) Neither DBMS allows the use of arrays with the command language syntax.

In addition, dBASE III Plus limits the number of open files at one time. R:Base 5000 allows up to 40 files with a maximum of 400 fields per file. dBASE III Plus is limited to 15 files and 128 fields per file.

However, in terms of the project, the students are not at a disadvantage using dBASE III Plus because the scope of the project does not exceed the file and field limitations. The students rarely have more than ten files open at one time, and no more than two files may be logically joined in internal memory at one time.

THE PROJECT SOLUTION

To implement a solution to the project, students must code a partial system using dBASE III Plus. This system must be menu driven, and must include file maintenance, inquiry screens, and reports that utilize the logical or physical joining of two or more data files.

For example, one case study described a hospital environment. In order to keep the system and the assignment within manageable size, the normalization and data modeling resulted in the following minimum entities: patient, physician, inventory, room, and cost center. However, the enterprise schema design model showed relations between these and other organization files such as billing, accounts payable, and accounts receivable.

In using dBASE III Plus, these entities translated to separate physical files. The student were required to do file maintenance of the patient file, a screen display which showed a physician and all the patients assigned to that physician, a daily revenue report, and a patient invoice showing items used and cost with total amount due.

The file maintenance program included data validation. Therefore, maintenance and display screens were created using format files. The

report generator of dBASE III Plus could not be used because the revenue and billing had to be run from combined files.

The entire system was presented by each project group. The instructor tested the file maintenance data validation by trying to insert incorrect data into fields. Reports were run, and student disks were turned in to verify that data is not hard coded for reports.

SUMMARY

A case study that not only presents a problem, but also describes an organization is used to teach CIS 353 Database Management Systems at Purdue Calumet. Problems are used that can be implemented using dBASE III Plus. dBASE III Plus is used for this course because it is currently the most popular database software package. It offers few advantages over R:BASE 5000, but dBASE III Plus' popularity suggests it is more likely to be encountered on the job by students.

The course teaches DBMS concepts that are easily applied to dBASE III Plus. For example, the data modeling and normalization process translates to dBASE III Plus files. Also, the concepts of data independence, data dictionaries, and database administration can be demonstrated with dBASE III Plus.

REFERENCES

(1) Bradely, J. (1987). Introduction to Data Base Management in Business. New York: Holt, Rinehart, and Winston.

(3) Green, L. (1986). A Phased Approach to the Case Study in Systems Analysis and Design Courses. Proceedings Fifth Annual Information Systems Education Conference. pp. 109-112.

(2) Pratt, P., Adamski, J. (1987). Database Systems: Management and Design. Boston: Boyd & Fraser Publishing Co.

(4) Poor A. (1987, January). Database Power Puts on an Easy Interface. PC Magazine. pp. 109-117.

WORKING WITH BUSINESS AND INDUSTRY ON SYSTEMS PROJECTS:

PITFALLS AND SOLUTIONS

Dr. Robert L. DeMichiell
School of Business
Fairfield University

Classroom systems development projects are undertaken for a myriad of reasons, but in the case of its requirement as a Senior Project, there are several important issues in their design and implementation which must be addressed completely and accurately at the outset. The contract between student and instructor must be concise and clear, the relationships among client, instructor, and student must be documented, and the project must force the student to apply the full range of skills gained in prior CIS and business work. The building of effective systems for completion in the semester time frame requires organization and attention to progress if a quality product is to be obtained. The approach taken here addresses the many pitfalls inherent in the process and provides practical guidelines for avoiding them.

PRELIMINARY COMMENTS

The transition from the classroom to the business world requires a departure from the typical task-oriented academic assignment so prevalent during most learning experiences in formal education. There are some commonalties, such as "deliverables" and team project orientation. However, the approach taken in this course does not place emphasis on the accomplishment of tasks predetermined by the instructor. The assumptions, constraints, project implementation schedule, and milestone definitions are the responsibility of the student. For many students this shift in responsibility and accountability from instructor to student can be challenging, exciting, frustrating, confusing, and rewarding.

The methodology for presentation of such a course on applications systems projects, which must include some involvement by business and industry in order to be representative of actual working conditions, can vary considerably with individual instructors. There are few texts devoted specifically to teaching this course. There are numerous texts, journal articles, and other educational materials on the systems development life cycle process with a myriad of systems analysis and design techniques, of programming approaches, and of implementation strategies. These topics should have been taken prior to the Senior Project, where an application is brought to bear on a problem they select, define, and complete.

Therefore, several aspects of the problem must be discussed at this time: theory vs. practice, students vs. working professionals, realistic expectations, real deliverables and commitments, client relationships, progress and project control, dealing with disasters, and finally, reports and evaluations from the educational institution and from industry. It is in this context that the presentation is cast and is complemented with numerous examples derived from experience with several hundred information systems student projects over the last fifteen years. Some guidelines are provided in an effort to avoid the pitfalls of

working with business and industry in such a necessary and practical capstone course.

SYSTEMS PROJECT "START-UP"

The nature of this course, a capstone senior course dedicated solely to one major systems applications project undertaken by each student, requires that the student understand from the outset that the following different set of rules apply:

- The work is individual and independent;
- Formal class meetings are infrequent and are complemented with several individual meetings with the instructor;
- The instructor provides guidelines and primarily becomes more reactive as the course proceeds;
- After some initial assistance by the instructor in the form of representative and possible projects, the student assumes full responsibility to find, evaluate, define, and complete the selected project;
- As the project develops, a contract between instructor and student surfaces and finalizes the deliverables, schedule, expectation levels (student, instructor, and now, the client);
- Client involvement is examined extensively for commitment, constraints, resources, and cooperativeness;
- Documentation of all project work commences during the first class day and continues throughout the project according to an outline predetermined by the student; and,
- Oral and written status reports are presented throughout the course, provide for some slippage of time, and do not provide for major modifications in design once the project is well underway.

Although such requirements may appear in other courses, the major problem for most students is the initiation stage. They don't see the upcoming scenario very clearly...they aren't told precisely what to do...how to find the project...what the report will look like...how much time will be involved ...where to go in the library for illustrative examples. The client in business and industry has many of these answers, but sometimes has different objectives. Systems projects may or may not include software design and coding; the course may de-emphasize this aspect, and the client very well may want just programming. The process of estimating the scope of projects, especially in the formulation stages, requires experience and presents a problem to the student, who generally has not had the opportunity to acquire it.

A major pitfall in the project definition stage is an over-extension of commitment by students. They think they can do more work than is possible during a semester. Their estimation of project scope, time to complete, and actual work usually is quite optimistic. The approach taken here is to discuss that aspect very thoroughly with the focus on completion of deliverables...as delineated in the contract...as promised to the client..with documentation...and with assured quality. It is at this time that political feasibility concerns are raised to complement the typical operational, economic, and technical feasibility studies.

The process of systems applications development is examined thoroughly with full consideration given to the previously mentioned pitfalls. Even though it has been covered in other courses, some framework must be identified so

that each student is able to:

- Scope the project in terms of analysis, design, and implementation;
- Mesh the project scope with the constraints of resources available and of time for completion; and,
- Identify the similarities and differences of his/her project to other projects. This procedure forces proactive planning for realistic objectives. The emphasis on speaking and writing skills, on modeling, on listening, and on human factors is prevalent not only for any software design, but also for all stages of the project.

Organizational issues probably will evolve as the student works with business and industry.

The approach here is to minimize the pitfalls by delineation of these specific problem areas and prevent them in project initiation, rather than correct them later. Also, importance is given at this time to the client's perspectives on return on investment, on degree of risk and uncertainty in project completion, and on productivity.

Is this a "crunch mode"project, or is it something that has been on the back burner for the past year waiting for some additional resource to address it? Is it just a learning experience, or will it be an operational system? How dependent is it on the current systems team, or end-user managers, or will it be done regardless of personalities?

THE PROJECT WORK

Once the several intensive questioning/answering sessions have been conducted in formal class settings and individual meetings, the project can move from the realm of possible and probable topics to the selected one. The project notebook has been organized to collect the data according to a student-defined outline, and the precise methodology has been documented. There is a shift in emphasis to management concerns. If teams are used, then the elements of project control must be expanded to include teamwork, leadership, and organization. There is a need to define task accountability, completion schedule, and systems relationships. Network diagrams would have to accompany this approach.

Student progress would be monitored very closely at the undergraduate level. All students are not self-motivators. They would tend to let the project work slip since it is a flexible enterprise compared to other more structured time demands. A continuous check should be made of predicted versus actual task completions. A pitfall here is the impression that if the schedule is met, everything is all right. Other relevant questions to ask at the milestones are: What is the quality of the work at this point? How is the documentation coming? To what depth have you explored the project at the various stages...and is this consistent with the student-instructor contract? Is it consistent with promises to the client?

As the project proceeds, one of several conclusions may be evident:

- Quality is assured, the project is on schedule, documentation is adequate;
- One or more of the above items may be incorrect, but the project can continue with proper adjustments; or,
- The project must be discontinued because any adjustment would be too late, or because the client has made major changes of intention (dropped it, gave it much lower priority, or made major design changes).

Disaster planning to some degree is inevitable. The theoretical model of systems development almost always is incomplete or inaccurate. Unanticipated events take place, client end-users and/or computer technical support people change jobs and new personalities enter the scene, or the student contracts mononucleosis. Magnification of the problem takes place because of the semester time frame. Therefore, contingency plans must be made to deal with a reasonable solution. In the case of the senior project, which generally is a core requirement, the consequences may be quite severe...no major...no graduation.

One answer is simply to change the project requirements, assuming that satisfactory progress has been witnessed thus far. Salvage the work completed and perhaps simulate the rest. What are the next steps and what probably would result if the project had continued? Any reduction of the scope of the project without such compensatory work would not be fair to the other students; additional personnel on the project is out of the question in this academia-business venture. The criteria for grading can be modified to cover such extenuating circumstances.

In addition to progress reports throughout the semester, there are final oral and written reports due near the conclusion of the course. The oral report is to comment on:

- Objectives, methodology
- Accomplishments to date, and those yet to be done;
- Problems encountered, expected, solved, unsolved; and,
- Documentation status.

The final written report is the final examination and must contain, but is not limited to, the following items:

- All notes and charts, handwritten, summarized, typed;
- An outline for documentation, including all initial reports;
- A logical arrangement of project materials, such as chronology and functions, etc.;
- Analysis, summary, commentary, appendices; and,
- All forms used in every aspect of the project.

The project has been completed, the work has been documented and reported, and it is compared to the promised deliverables. Progress is evaluated, the client is contacted for their input, and the grade is established. Students have worked harder than anticipated, but have acquired some experience necessary for entry into the job market; clients have received free systems work; and informations systems theory becomes practice. If the pitfalls are addressed early in the course, then proper solutions evolve and the reasonable expectations are realized.

The approach taken here fosters student creativity, instructor guidance, and client participation, all of which act to produce a synergistic effect. The methodology certainly is not novel, except that the procedure is standardized and represents the culmination of several years of experience in undergraduate and graduate project work with business and industry. From the point of view of academicians and corporate professionals, this approach exemplifies the need for careful planning to ensure that the process is indeed accelerated and concurrently, effective.

EXPERT SYSTEMS' IMPACTS ON MANAGEMENT INFORMATION SYSTEMS COURSES

Chi-Chung (David) Yen, Miami University
Hung-Lian Tang, Bowling Green State University

ABSTRACT

Since the introduction of electronic data processing systems (EDP) in the early 1960s, computer based information systems have gone through tremendous developments in capability and design. In recent years, one advanced type of an information system, expert systems (ES), made its appearance and is evolving at a steadily increasing pace, not only in concept development, but also in practical applications. This introduction of ES concepts and their successful applications creates two interesting situations. In one way it creates a new research area in the field of business information systems, in the other way it results in some challenges to the course content of the existing management information systems' (MIS) courses. The main purpose of this paper is to discuss the ES impacts to the MIS courses such as system analysis and design, data base, computer graphics, application programming, and general system theory (including MIS or decision support systems).

INTRODUCTION

Since the 1970s, the development of expert systems (ES) has accelerated and applications have developed in a variety of areas such as management, production, scheduling, and portfolio analysis. This growth is in spite of the slow improvement in the introductory stage of its development. Currently, every person in every business or computer related discipline is talking about this powerful product. Further, many new software packages have made their appearance with the name of artificial intelligence (AI) or expert systems (ES) even though these products just take some ES ideas (e.g., using rule-based technique). It is apparent that ES are so popular that no one can resist their influence. This is especially true in the field of management information systems (MIS). Therefore, more and more educational organizations are starting to offer AI and ES courses to make their students more competitive in the job market. In addition, AI or ES related knowledge is being mentioned more often in some particular MIS courses. This paper tries to discuss the ES impacts to the existing MIS courses such as system analysis and design, data base, computer graphics, decision support systems or management information systems, and business application programming courses in terms of ES concepts, ES techniques, ES tools/languages, and ES applications.

BACKGROUND OF THE EXPERT SYSTEMS

An ES is formerly defined by Feigenbaum in 1982 as follows:

> An "expert system" is an intelligent computer program that uses knowledge and inference procedure to solve problems that are difficult enough to require significant human expertise for their solutions. The knowledge necessary to perform at such level, plus the inference procedure used, can be thought of as a model of the expertise of the best practitioners of the field.

In this section, this author tries to discuss ES impacts on the existing MIS courses in terms of the following areas:

(1) System analysis and design (including system development and implementation)
(2) Data base management systems
(3) Management Information Systems (including decision support systems)
(4) Business application programming

The Area of Systems Analysis and Design

Traditionally, there is a large number of systems analysis and design methods in this area. Generally speaking, these methods can be divided into two major categories. The first category is often related to the management information systems discipline while the second one is associated with the decision support systems (DSS) discipline. In the MIS category, these methods include traditional system development life cycle concept, James Martin's information engineering method, Tom Demarco's structured analysis method, Yourdon and Constantine's structured design method, Ken Orr's output-oriented design method, Michael Jackson's systems development method, Hamilton and Zeldin's high order software (HOS) method, and system prototyping method. The DSS category includes such famous methods as Carlson's representation-based approach, Keen's adaptive design approach, Courbon's evolution approach, Schonberger's contingency approach, and Sprague and Carlson's quick-hit approach.

With the emergence of ES, we now encounter the following question -- "Do we need to cover some ES analysis and design techniques, to some extent, since ES are one major type of the business information systems?" If the answer is yes, then at least some ES design techniques and related terms must be mentioned briefly to complete the knowledge in this area regardless if the students will pursue any further study in the AI/ES area.

Further, what is an ES analysis and design method? This is a very hard and interesting question and there are no unified answers so far. The major reason for the lack of a consensus is because every individual ES application, to some extents, uses a unique design methodology. This is partly because the development of ES is so rapid that there is no time for ES practitioners and researchers to draw a unified framework of various ES design methodologies. Based on this author's point of view, ES design framework can be roughly distinguished into two directions: searching-oriented and reasoning-oriented. In the searching oriented system design, ES techniques such as factoring into metaproblems, abstraction solution space, heuristics search, and trial and error (hill-climbing) are commonly used. On the other hand, such techniques as justification building, assumption building, inference with symbolic constraints, and generate and test are widely applied to deal with reasoning-oriented system design.

In addition, ES tools/languages may also influence this area. Instead of using third generation languages (e.g., FORTRAN, COBOL, etc.) and fourth generation languages (e.g., SQL, MANTIS, FOCUS, etc.), ES prefer to use fifth generation languages (e.g., LISP, PROLOG) and specially designed languages suitable for particular applications (e.g., OPS5, HEARSAY-III, and ROSIE). Consequently,

this preference for different languages adds complexity to this area, especially to the system development and implementation.

The Area of Data Base Management Systems (DBMS)

The emergence of ES may create some future impacts in the area of DBMS, specifically in three related disciplines:

(a) Semantic modelling in data base
(b) Object-oriented data base design
(c) Information retrieval and query answering interfaces

Sematic modelling can use the ES techniques to understand certain simple atomic data values, and hence, better understand the query and thus respond a little more intelligently to users' requests or interactions. For example, suppose we have two data base files. One deals with shipments while the other deals with parts. There is one field, "quantity," in the shipments file while there exists one attribute, "weights," in the parts file. Of course, everyone understands that, though both attributes have numeric values, the two values are semantically different because of the different units of measure associated with each. It would be nice to differentiate these two attributes semantically while answering a query. ES can make semantic modelling more powerful and more effective in the near future. Consequently, it will have another new area in data base management systems which instructors in this area may need to teach.

Secondly, object-oriented data base is emerging to deal with some complex applications which do not map well into any of the existing data model we covered so far. Namely, these existing data models are: (1) entity relationship model, (2) network model, (3) hierarchical model, (4) relational model (includes relational algebra and relational calculus), and (5) inverted lists model. So, why has the objected-oriented data base model emerged? For example, users may want to work on a house plan which is made up of a list of components, say doors, windows, fireplace, bricks, different sizes of lumber, and carpets, etc. As we can see that the relationships among these data items are very complex, and cannot be solved by the above five conventional static models. Therefore, we must use some kind of expertise to find out the interactions between different items. In addition, some reasoning and generate-and-testing techniques of ES need to be used to make object-oriented data base models more refined and sophisticated. There is no doubt that in the near future, object-oriented data bases, with the help of ES techniques, will play a much more important and active role in this area.

In addition, it is possible that the future end users will no longer need to enter their query in certain format/style or in some restricted software commands provided by the vendors. Rather, they will key in English sentences or paragraphs and let the system recognize the text, restructure the text, translate the text into some specific executable commands, execute them and then present the results. In order to achieve this task, the language understanding capability and the language synthesis capability of ES are definitely needed. Maybe in the foreseeable future, the area of informational retrieval and query-answering interface will have some dramatic and major changes.

The Area of Information Systems

Actually in this subsection, this author intends to discuss the ES's impacts to several types of major information systems such as management information systems (MIS), decision support systems (DSS), and electronic data processing systems (EDP) as a whole.

Traditionally in this area, instructors will cover such ideas as (1) history and background of the information systems, (2) concepts and techniques of MIS, DSS,

and EDP, (3) MIS strategy and planning, (4) resource controlling and scheduling, (5) the nature, similarities and differences among these three types of systems, (6) centralization vs. decentralization, (7) DSS generators and models, and (8) the distributed environment. However, we need to introduce something new because of the appearance of ES. At least some basic and preliminary understandings about ES is needed to prepare the students for further advanced study in the AI/ES area. More explicitly, some extra topics which are relevant to the general system theory such as (1) ES terminology (e.g., rule base, knowledge engineering, and inference engine), (2) ES concept (e.g., factoring, abstraction, heuristics search, and reasoning), and (3) ES techniques (foreward chain, backward chain, and describe and match) are essential. Furthermore, the following two topics, the differences and similarities between ES and conventional business information systems, and when/where to design the right type of systems to deal with the decision making and problem solving are extremely useful to our students.

The Area of Business Application Programming

Compared to the previous four topics, this topic should be an easy and simple one. The major impact that ES creates on this area is in ES tools/languages.

Recall the first subsection "The Area of Systems Analysis and Design," we know that two major types of tools/languages that ES prefer are fifth generation languages (5GL) and specially designed languages. Specially designed languages actually do not affect this area at all, partly because teaching them will not benefit the student in a business school but it is also against the principles of an MIS curriculum (e.g., the future jobs of our graduates are either as a system analyst or an information specialist, not a computer programmer), and partly because these specially designed languages have a strong tendency for the knowledge of compiler/parser construction and language structure which our students need not know unless they are interested in it. Therefore, the core of the ES impacts on this area is actually in 5GL. Perhaps in the near future, in addition to giving the students some preparation in 4GL, procedure languages, and non-procedure languages, we need to introduce 5GL to some extents.

CONCLUSIONS

Expert systems are one of the most exciting developments in the history of the computer. While most artificial intelligence techniques remain in the research laboratories, the expert systems are emerging from the lab into the real world. Today, the United States, Japan, England and the European Economic Community are all in the process of launching major research programs to develop and implement expert systems. This paper tries to discuss the contents of several major important MIS disciplines, and hence, provide some exposure to the new contents in these MIS disciplines. However, this author does not imply that the knowledge of AI or ES can be taught isolately and individually in several separated courses.

References available upon request.

TEACHING COMPUTER ARCHITECTURE AS A BASIC PART OF THE INFORMATION SYSTEMS CURRICULUM

Irv Englander, Ph.D.
Computer Information Systems
Bentley College
Waltham, Massachusetts 02254
(617)891-2585

ABSTRACT

Computer Architecture forms a basic technological foundation for the study of Information Systems. The Bentley College implementation of this course considers the computer in terms of three major components: data and its representation, the hardware facility, and the software tools and support provided by the Operating System. Programming assignments are used to reinforce the fundamental concepts. The emphasis is on the relationship of the computer's components to the typical user's activities. This paper describes our paradigm for the course.

COURSE OVERVIEW

The Computer Information Systems Department at Bentley College offers degrees at the Bachelors' and Masters' levels, using a curriculum modelled after the ACM Information Systems Curriculum Recommendations [1].

The Computer Systems Principles course is an implementation of course IS1 in the model curriculum. It is offered as a one semester requirement at the sophomore undergraduate level, with prerequisites of an introductory data processing course and a semester of Pascal programming. Students entering the course have been exposed to both microcomputer and multi-user interactive environments. Typically, the students are intelligent and well motivated, but lack a strong formal background in mathematics.

There is also a similar required graduate level course in the corresponding curriculum position.

GOALS FOR THE COURSE

The stated goal for this course is to provide a fundamental understanding of the concepts and terminology associated with computer hardware and software architecture as a basic technical foundation for the study of Information Systems.

Underlying the stated goal is a more specific set of conceptual goals:

*To give the student a realistic view of the computer as a finite resource; to understand its abilities and limitations in terms of such factors as speed and storage capacities.

*To reinforce the concept that computers are simple, albeit fast and powerful, data manipulation and computational devices; that so-called "thinking" by computers is, in fact, clever use of computation and data manipulation within software.

*To expand the student's ability to make effective use of computer tools and software resources.

*To help the student to understand the concepts that influence and affect computer program construction and performance, by understanding the underlying operations that take place.

*Finally, to remove some of the "mystique" surrounding the computer; to acknowledge the computer as a box filled with hardware and software. We sometimes refer to this goal as "humanizing the computer by dehumanizing it".

COURSE IMPLEMENTATION

The focus of the course is on a detailed, though not highly technical, understanding of the operation of the various hardware and software components that make up a computer system, as well as the relationship of programs and data to the system.

The course is broken into three modules, consisting of data representation, hardware architecture, and the operating software environment. The graduate level course differs from the undergraduate course mostly in the time alloted to each module; at the graduate level, data representation is covered very quickly. This allows us to discuss the Operating System software with significantly more depth than in the undergraduate course.

Data Representation Module

The first part of the semester is devoted to a discussion of data representation. Topics include counting and arithmetic in different number bases, conversion from one number base to another, methods of storing and manipulating negative numbers, representation and manipulation of integer and real numbers, and the representation of character strings and boolean variables.

The material explores the relationship between word size and numerical storage and computation limitations. It explains why overflow can occur with integer arithmetic. It suggests the extra computation required with real numbers. It shows the difference between numbers stored in a numerical form and those stored as strings of characters. It shows the difference between ASCII (or EBCDIC) and binary data files.

Related Pascal assignments reinforce these concepts: students learn the difference between the data typed into a keyboard from that represented internal to the computer by writing a program to input a number as a character string and convert it to integer. They convert numbers from a binary representation to decimal. They learn the difference between real and integer number representation by converting real numbers to a sign-and-magnitude excess-50 decimal floating point format.

Hardware Architecture Module

The second module of the course focuses on operation of the various hardware components that make up a computer system. Roughly 35% of the course is devoted to study of the hardware architecture. Topics for this module include:

*The components that comprise the CPU, including a description at the block diagram level, as well as an introduction to register operations, the movement of data between registers, and how memory access works.

*The makeup of a typical, fundamental instruction set.

*An explanation of how the computer processes instructions, including study of the Fetch-Execute cycle.

*The extensions to the fundamental instruction set that exist in different computers: different addressing modes, multiple registers, stacks, etc.

*How to write simple programs in machine language.

*How I/O is performed in a computer. Our specific topics include programmed

I/O, the concept of busses, interrupts, and Direct Memory Access.

*The relationship between common program structures in high level languages (loop, IF-THEN-ELSE, subroutines, array manipulation) and the underlying machine language implementation.

For this module, we rely on an extended version of a model known as the Little Man Computer. The original model was developed for course use at Massachusetts Institute of Technology by Dr. Stuart Madnick [2]. This model provides a close analogy to the Von Neumann CPU architecture, and can be tailored to demonstrate many of the different aspects and features of CPU hardware.

The original model consists of a single register adding machine, a hundred mail boxes used for program and data, an in basket, an out basket, a "location" [program] counter, and a Little Man. The instruction set has nine basic instructions (LOAD, STORE, ADD, SUBTRACT, IN, OUT, COFFEE BREAK, SKIP ON CONDITION, GOTO), and supports direct addressing, with a three digit decimal word.

We have extended the model to include discussion of different instruction addressing modes (immediate, indirect, indexed, relative, subroutine), multiple address instructions, different methods of I/O, the concept of bootstrapping, and, later in the course, assembly language and the workings of assemblers and loaders. A printed description of the LMC model will be available from the author at the conference.

Students are assigned to write a simple program in LMC machine language (to input three numbers, and present the output in sorted order). For this purpose, we have written an LMC machine language interpreter that uses commands similar to those of Basic. The LMC machine language interpreter is written in Pascal, and will also be available at the conference or by mail from the author.

The Operating System and Software Tools Module

The final segment of the course focuses on the operating system and software tools provided in a typical system. Two important concepts guide our discussion:

*That the computer requires software to operate. Thus, the Operating System provides program continuity in the sense that there is software to keep the machine running when the user is not actually executing a program. We use the LMC model to explain this concept.

*That the Operating System is made up of programs, similar to a user's programs, but supplied with the computer; that to many of the Operating System's programs, the user's program represents data.

We introduce this module with a detailed discussion of the steps which take place during the compilation, linked loading, and execution of a program. We have discovered that these procedures tend to be a mystery to our students; understanding what is occuring within the computer clarifies the steps and clears up a lot of confusion.

This discussion allows us to consider such issues as program relocation, the operation of assemblers, the concept of libraries and linking of subroutines, as well as more subtle ideas such as Operating System control, scheduling, file structures and directories, and memory management and use.
At the graduate level, we extend the third module to include an in depth discussion of Operating Systems, including detailed discussions of process scheduling, resource

allocation, virtual memory techniques, deadlocking, and data protection.
The last two programming assignments lead the students through the Pascal design of a two-pass assembler for the Little Man Computer model.

EVALUATION OF OUR METHOD

A superficial but measurable test of the course's effectiveness is the depth of student understanding of computer architecture that results, as measured by examinations in the course material. Exams in this course are designed to test the students' ability to apply the material, rather than to reproduce it. The results indicate that the students have an excellent understanding of computer hardware and software.

At the graduate level we also assign a term paper, asking the students to describe in some detail the architecture for a computer of their choice, working from the technical manuals for the computer. The students are expected to anlyze the block diagram, to discuss the interconnection between the various components, to consider the available addressing modes, to comment on the instruction set, and to discuss any special features of the machine, such as memory management or virtual addressing. The students have expressed pleasure at their acquired ability to perform this task. We have been pleased with the results.

More useful, perhaps, are the numerous informal written and oral comments from students at the completion of the course and during following semesters. In addition to these informal comments, we also request written general course evaluations of all our graduating seniors. This course rates high in the curriculum. Numerous students have specifically referred to the Little Man Computer model as an important tool in their education. The same enthusiastic comments are received from our graduate students. The nature of the comments leads us to believe that the conceptual goals for the course have been met.

Finally, our paradigm for this course has also been adopted with success recently in a similar program at Boston University [3].

CONCLUSIONS

After nearly five years of experience with this course, we believe strongly that the introduction of computer architecture at an early stage of the curriculum eases the difficulties that Computer Information Systems students often face in understanding and using the computer. The Little Man Computer model appears to be an enjoyable and effective way to introduce and explain the important concepts of hardware architecture. Joined together with discussions on computer software and data representation concepts, the course results in a high level of comprehension by the students. We believe that our goals are effectively achieved.

REFERENCES

[1] ed. Nunamaker,J.F.,Jr., Couger,J.D., Davis,G.B., "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs", Comm. of the ACM, Vol.25, No. 11, Nov. 82, p.781+

[2] Personal communication with Dr. Stuart Madnick, Sloan School of Management, MIT.

[3] Personal communication with Mr. Richard Briotta, Department of Management Information Systems, Boston University.

ARTIFICIAL INTELLIGENCE TECHNIQUES FOR BUSINESS: A COURSE PROPOSAL FOR THE 1986 DPMA MODEL CURRICULUM

Scott C. McIntyre, Richard Discenza, and Lexis F. Higgins
College of Business and Administration, University of Colorado

ABSTRACT

The 1986 DPMA Model Curriculum describes two courses which address Artificial Intelligence (AI): "Artificial Intelligence in Decision Making," (CIS/86-12) and "Decision Support and Expert Systems" (CIS/86-11). These courses are oriented strongly towards special topics in AI, specifically Management Science topics. In this paper the authors present arguments for a broader approach and describe a course which is currently being presented to students and management at the University of Colorado, Colorado Springs. Knowledge and skill levels for each facet of the course are presented.

This research was supported in part by grants from Xerox Artificial Intelligence Systems and EXSYS, Inc.

ARTIFICIAL INTELLIGENCE IN PRACTICE

Artificial Intelligence (AI) is no longer an esoteric subject confined to laboratories and understood only by initiates. Numerous predictions indicate that by the mid-1990s more than half of the computers sold will contain AI components and that the annual revenues for AI hardware, software and services will exceed $8 billion ,10.. Articles describing the impacts of AI upon science, business and society may be found in such widely read publications as Newsweek, US News and World Report and The Wall Street Journal. A national society devoted to AI in the United States, the American Association of Artificial Intelligence, currently has over 10,000 members from business, government and academia.

One of the most serious problems facing the many organizations that wish to begin AI initiatives is the lack of trained personnel. To illustrate, during the 1987 Texas Instruments Artificial Intelligence Satellite Symposium, beamed to over 1200 international locations, experts in the science and business of AI were asked how one could "get started" in AI. Rather than suggesting courses or books, these experts from academia and industry suggested that one buy some low level AI software and begin a process of self education. This suggestion did not explicitly rule out courses in colleges and universities; instead there was an implicit recognition that these resources are not widely available for the broad range of interests in AI.

Figure 1 describes the major research categories associated with AI. Knowledge systems codify human cognitive abilities and include expert systems (5), knowledge based decision support systems (1) (8) (9), learning systems (12), and expert data base systems (7). Natural language systems seek to provide the computer with the ability to communicate with human language, both verbal and written. A promising application is communication with data bases (13). Robotic systems link mechanical systems with knowledge systems to produce intelligent agents (4). Vision systems link visual input systems like cameras to knowledge systems to enable computers to both see

and understand what they are seeing (2).

Any list of AI applications is not only extremely large, but immediately out of date. A general introduction to the subject may be found in (11). Figure 2, based upon (6), presents an overview of AI applications. Current industrial applications are numerous and their growth continues to be exponential. Extensive research and development is currently underway in applying AI techniques to service industries such as banking, insurance and medical data processing. Tasks associated with these have high intellectual content. The impact of AI in this area is expected to be extensive based upon success with operative systems. Research into AI support for managerial decisions is difficult, but the payoff is expected to be extremely important. Knowledge based decision support systems are expected to help managers to deal with the vast amount of information and make consistent, effective and efficient decisions.

A COURSE FOR BUSINESS STUDENTS

Business schools educate individuals to become productive and effective in all aspects of business and government. Recognition of the importance of AI to these organizations is growing among business schools, resulting in a growing interest for offering AI courses and for discussing AI topics in a wide range of business courses. The 1986 DPMA Model Curriculum describes two courses which address (AI): "Artificial Intelligence in Decision Making" (CIS/86-12) and "Decision Support and Expert Systems" (CIS/86-11). These courses are specifically designed to address the influence of AI upon management science techniques and applications, and upon decision support tools for mangers. Given the current range of AI theory and application, the authors believe that these courses offer valuable special topics, but that they are much too narrow to address the breadth of students' needs for AI knowledge.

A course which covers the broad range of AI technology and applications and which contains a special orientation toward business is described in the remainder of this paper. This single semester course is taught at the University of Colorado, Colorado Springs and is offered to both graduates and undergraduates. Full time students and special students participate, including managers from government and industry.

Table 1 identifies the course outline and stipulates the recommended level of student comprehension. These levels are the Knowledge level (acquired through self-study, listening to lectures, working problems or analyzing cases) and the Skill level (acquired through completion of projects representative of real-world situations) (3).

The course has numerous objectives. Students should become familiar with the goals of AI at the Knowledge level. This helps to dispel myths and place AI in perspective. Technical foundations are gained at the Skill level through the use of class exercises and a major project. Knowledge representation and inferencing are at the heart of AI programming, whether for knowledge systems, natural language, robotics or vision.

The use of expert system shells and object oriented languages on micro computers is an inexpensive and effective way to achieve Skill levels in knowledge representation and inferencing techniques. These tools provide development systems which enable the students to construct simple but powerful AI programs in English. Many programs developed by students are no different in quality than those currently in use in industry today. Examples of such tools include EXSYS, GURU, ACTOR, Smalltalk AT and many others.

If economically possible, the extensive development tools offered by Lisp Machines can be employed by advanced students, although a Knowledge level is suggested for all. These machines employ advanced software technologies and are expected to have profound impact on AI and conventional software development in the future. Vendors of such machines include Xerox, Hewlett-Packard, Texas Instruments, Symbolics and Digital Equipment Corporation.

Lisp is currently the language of choice for the American AI community. Prolog is

extremely powerful, but limited in the kinds of problems it can address. The student is expected to achieve the Knowledge level in Lisp and, secondarily, Prolog. Advanced students achieve the Skill level, but mostly through private work with the instructor.

The proposed course is not a language course. Students are given projects and lab assistance so that extensive class time may be devoted to the rest of the course. In the remainder of the technical foundations section of the course, the student attains knowledge level in the theory of cognition, natural language, vision and robotics.

In the next segment of the course, students are exposed to a wide range of tools for constructing AI systems through lecture and demonstration. Emphasis is placed not only upon individual tools, but upon parameters for tool classification and applicability. Lectures are conducted in the purpose and applications of Expert Systems, other Knowledge Based Systems, Natural Language Systems, Vision Systems, and Robotic Systems. Computerized and video tape demonstrations enhance the lectures.

In the fourth section of the course, the student is exposed to the Knowledge System Development Lifecycle. This is typically discussed concurrently with the rest of the course, since the student is using this lifecycle to develop a semester long project. In the lecture material, the student learns a framework for knowledge system development, how that framework differs from other lifecycles, and what tools may be employed to construct the knowledge system.

In the final segment of the course, the student achieves knowledge level understanding of the business of AI. Through discussions of the topics listed in the figure, students are exposed to strategic, tactical and operational issues. Lectures are enhanced with visits from AI professionals in the local business community.

SUMMARY

The academic business community must fulfill the needs of industry in the area of Artificial Intelligence. To do so, it must provide one or more courses which familiarize business students with the technical and organizational issues faced by government and industry. Narrow, special interest courses provide valuable insights but limited scope. Inclusion of the described course in the curriculum of the University of Colorado at Colorado Springs has shown that the suggested approach offers students interesting and relevant experience to address the rapidly expanding role of AI in business and government.

REFERENCES

(1) Applegate, L. M., T. T. Chen, B. R. Konsynski, and J. F. Nunamaker, "Knowledge Management in Organizational Planning," Proceedings of the Twentieth Hawaii International Conference on System Sciences, v. 1, 1987.

(2) Charniak, Eugene and D. McDermott. Introduction to Artificial Intelligence (Ch. 3), Addison-Wesley, Reading, MA, 1985.

(3) Couger, J. D. and J. F. Nunamaker. "Delineation of Level of Coverage for Each Topic in the ACM Recommended Information Systems Degree Program", Proceedings of the Third International Conference on Information Systems, 1982, Ann Arbor, MI. (pp. 311-333)

(4) Craig, John J. Introduction to Robotics: Mechanics and Control, Addison-Wesley, Reading, MA, 1986.

(5) Harmon, Paul and David King. Artificial Intelligence in Business: Expert Systems, John Wiley and Sons, New York, 1985.

(6) Kaplan, S. Jerrold, "The Industrialization of Artificial Intelligence" The AI Magazine, Vol. 5, No. 2, 1984.

(7) Kerschberg, Larry (ed.). Expert Database Systems, Benjamin Cummings, Menlo Park, CA, 1986.

(8) McIntyre, S. C., B. R. Konsynski, and J.

F. Nunamaker, "Automating Planning Environments: Knowledge Integration and Model Scripting," Journal of Management Information Systems, 2, 4 (Spring 1986).

(9) McIntyre, S. C. and L. F. Higgins, "Knowledge Base Partitioning for Local Expertise: Experience in a Knowledge Based Marketing DSS," Proceedings of the Twentieth Hawaii International Conference on System Sciences, v. 1, 1987.

(10) Miller, R. K. Artificial Intelligence Applications for Business Management, SEAI Technical Publications, Madison, GA, 1985.

(11) Rauch-Hindin, Wendy B. Artificial Intelligence in Business, Science and Industry, (Vols. 1 and 2), Prentice Hall, Englewood Cliffs, NJ, 1985.

(12) Schank, Roger C. and Peter Childers. The Cognitive Computer (Chapter 8), Addison-Wesley, Reading, MA, 1984.

(13) Wallace, Mark. Communicating with Databases in Natural Language, John Wiley and Sons, New York, 1984.

Table 1. An Outline for the Business AI Course

	Topic	Knowledge	Skill
1.	Introduction to AI		
	- Overview of AI	X	
	- AI History	X	
	- AI Goals and Perspectives	X	
2.	Technical Foundations of AI		
	- Knowledge Representation		X
	- Inferencing		X
	- Learning in AI	X	
	- Planning in AI	X	
	- Natural Language	X	
	- Knowledge Based Systems		X
	- Object Oriented Systems		X
	- Computer Vision	X	
	- Robotics	X	
3.	AI Tools and Applications		
	- Tools for Knowledge Systems Construction		X
	- Applications of Expert Systems		X
	- Applications of Knowledge Systems	X	
	- Applications of Natural Language Systems	X	
	- Applications of Vision Systems	X	
	- Applications of Robots	X	
4.	The Knowledge Systems Development Lifecycle		
	- Planning for AI		X
	- Knowledge Engineering		X
	- System Construction		X
	- Integration		X
5.	The Business of AI		
	- Inter-Organizational Considerations	X	
	- Intra-Organizational Considerations	X	
	- The AI Entrepreneur	X	
	- Marketing AI	X	

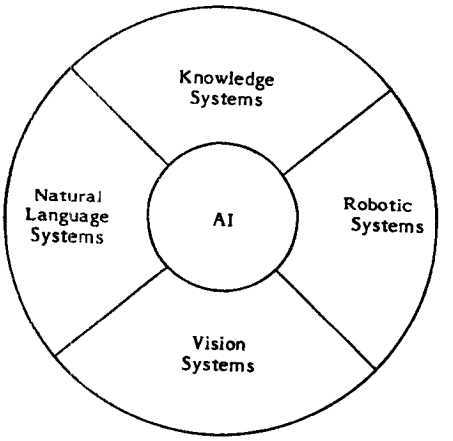


Figure 1. A Typology of AI and Its Current Subsets

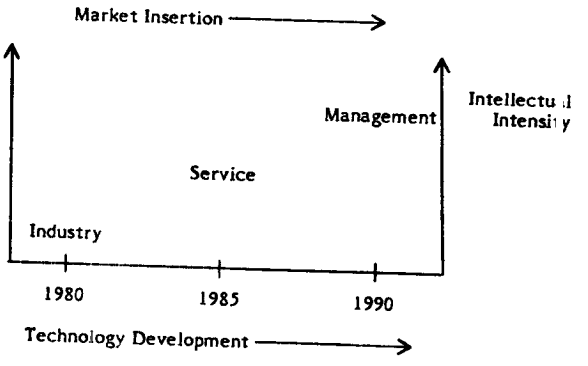


Figure 2. Trends of AI Applications and Development

Industry Perspective on System Development Productivity Aids

Lewis A. Myers, Jr., Visiting Associate Professor
Department of Business Analysis and Research
Texas A&M University
College Station, Texas 77843-1233
(409) 845-2389 BITNET: MYERS @ TAMCBA

E. Reed Doke, Associate Professor
Computer Information Systems Department
College of Business Administration
Southwest Missouri State University
Springfield, Missouri 65804-0095
(417) 836-4131 BITNET: D006 @ SMSVMA

ABSTRACT

This paper reports on the results of the first round analysis of responses to an industry wide survey. The purpose of the survey is to identify productivity aids currently in use and the relative importance of installed productivity aids. Six major categories of productivity aids are included in the study. These categories are: (1) management polices, (2) organizational structure, (3) development methodologies, (4) software tools, (5) hardware tools, and (6) ergonomic factors.

Introduction. This paper reports the results of the first round analysis of responses to an industry wide survey. A primary objective in this project was to determine what productivity aids are being used in industry and how important each of these aids are. A productivity aid is defined here as any tool or technique that results in more or better output for a given level of input during system development. The most expensive input during the system development process is the labor or manpower input. The output that we are concerned with in system development includes software and its related documentation. The production of software and its related documentation is dependent on all phases of the system development life cycle to some degree. The system development life cycle model is usually depicted as a continuous series of steps or phases. Snyder and Cox (10) conducted an extensive review of system development life cycle models that determined that the stages or phases are usually depicted as a continuous circle of events. When the last step or phase is completed the cycle begins again with the first step. In this paper we define the system development life cycle (SDLC) as having six identifiable steps or phases. These phases are: (1) the preliminary study, (2) analysis, (3) design, (4) development, (5) implementation, and (6) post-implementation. Productivity aids are separated into six

major categories in this paper. These categories are: (1) management policies, (2) organizational structure, (3) development methodologies, (4) software tools, (5) hardware tools, and (6) ergonomic factors. The survey instrument was developed to determine the importance of each phase of the system development life cycle.

Specific Productivity Aids. The taxonomy described by Myers (8) was used in this study to expand each of the six major categories of productivity aids. Each of the major categories includes several specific productivity aids. These are briefly described below along with a detailed outline of their subcategories.

1. Management Policies - Paul (9) stated that formal objectives must be established to improve production. Jamieson and Wyckoff (6) said poor productivity existed because the focus has been on programmers instead of on managers. Obviously, management policy can have a significant impact on software development productivity. The following are some specific sub-categories of managment policies as described in the Myers (8) study.

- 1.1 Flextime (flexible working hours)
- 1.2 Formal performance evaluation system
- 1.3 Formal productivity objectives
- 1.4 Telecommuting
- 1.5 Training program

2. Organizational Structure - Authur (1) suggested five factors that impact productivity: (1) methodology, (2) technology, (3) people, (4) organization, and (5) external variables. Jeffery (7) likewise indicated social and organizational factors can inhibit productivity improvements. Jamieson and Wyckoff (6) concluded that organizational and job structure are important factors impacting productivity progress. This paper groups organizational aids into the following sub-categories.

- 2.1 Database administrator
- 2.2 End user programming
- 2.3 Independent test group
- 2.4 Information center
- 2.5 Maintenance group
- 2.6. Professional programmer
- 2.7 Programming assistant
- 2.8 Quality assurance team

3. Development Methodologies have made major contributions to software development productivity. Arthur (1) and others have recognized the impact of methodologies in all phases of the SDLC. For example, Connell and Brice (3) suggest using prototyping throughout the life cycle to boost productivity and meet user requirements. The major subcategories of these methodologies are listed below.

- 3.1 Code inspection and walkthrough
- 3.2 Design inspection and walkthrough
- 3.3 Charting techniques
- 3.4 Prototyping
- 3.5 Pseudocode
- 3.6 Structured design and analysis
- 3.7 Structured programming

4. Software Tools - In this paper a software tool is one or more software modules that are used to decrease the total effort needed to develop and maintain a software system. Stated another way, a software tool is a local optimization technique that is applied to shorten the effort required to complete or to improve the quality of the output from one phase or a set of phases in the SDLC. Frequently individual software tools are combined to form integrated multifunction packages referred to as a programmer work bench or station. Software tools can be classified into one or more of the cate-

gories listed below.

4.1 Code production tools
- 4.1.1 Code generators
- 4.1.2 Menu generators
- 4.1.3 Nonprocedural languages
- 4.1.4 Reusable code

4.2 Documentation tools
- 4.2.1 Automated data dictionaries
- 4.2.2 Automated program documenters
- 4.2.3 Automated slide producers
- 4.2.4 Automated text processors
- 4.2.5 Automated typesetters
- 4.2.6 Graphics generators
- 4.2.7 Screen generators
- 4.2.8 Spelling checkers

4.3 Management tools
- 4.3.1 Cost accounting system
- 4.3.2 Cost and schedule estimating
- 4.3.3 Job scheduling
- 4.3.4 Project control - PERT/CPM/ Gantt
- 4.3.5 Production monitoring system

4.4 System design
- 4.4.1 Application generators
- 4.4.2 Computer aided design - interactive text/graphics
- 4.4.3 Database management system

5. Hardware tools are those hardware components that have the potential for improving productivity in software development. While the benefits of an online development environment have been clearly demonstrated, the potential of other tools may not yet be realized. The following list indicates some of the more obvious hardware tools.

- 5.1 Electronic mail
- 5.2 Teleconferencing
- 5.3 Online program development
- 5.4 Word processing

6. Ergonomics is concerned with the study of factors that affect the welfare, satisfaction, and performance of people working with man-made systems and equipment. Strain and fatigue can develop when programmers and analysts must work in an uncomfortable physical environment. Studies have shown increases in productivity when attention is paid to ergonomics. The National Institute of Occupational Safety and Health has stated that productivity can be enhanced by 25 percent just by paying attention to posture and vision (2). A Norwegian study revealed a 50 percent reduction in absenteeism when ergonomically designed furniture was provided to CRT operators (5). Subcategories to be considered in ergonomics are:

6.1 Terminals
- 6.1.1 Adjustable viewing positions
- 6.1.2 Detachable/relocatable keyboards
- 6.1.3 Nonglare screens
- 6.1.4 Screen phosphors
- 6.1.5 Terminal color scheme

6.2 Office furniture
- 6.2.1 Chairs with adjustable backrests and seat height
- 6.2.2 Desk height, adjustable ledge
- 6.2.3 Source document holders

6.3 Work environment
- 6.3.1 Lighting levels
- 6.3.2 Sound levels

Preliminary survey results. Following a pilot study that included a questionnaire mailout to 30 managers the main survey was conducted. The main survey consisted of a questionnaire mailout to an additional 470 managers. The survey instrument provided a detailed list of productivity aids and asked the manager to indicate which aids were currently in use in their organization. The survey instrument also asked the respondent to indicate how important each installed productivity aid was during each different phase of the SDLC. At this time responses have been received from 78 individual managers, a response rate 15.6%. Followup analysis

is being conducted to evaluate the reason for the low response rate. The preliminary finding is that the length of the questionnaire was the major factor behind the low response rate. The construction of the questionnaire required as many as 450 independent answers and that is likely the reason for the low response rate.

Some of the preliminary results of the study are summarized below:

TABLE 1: RESPONDENT PROFILE

Organization Size: Ranges from 24 employees to 85,000 employees.
Information System Professionals: Range from 1 to 1500 employees.
Industry Representation: 13 different industries are represented.
Job Responsibilities: 91% of all respondents held managerial positions.

Table 1 above provides summary information on the profile of respondent organizations and personnel. Table 2 below summarizes the most used and the least used productivity aid as indicated by the respondents.

TABLE 2: MOST USED AND LEAST USED PRODUCTIVITY AID BY CLASSIFICATION

1. MANAGEMENT POLICIES
   - MOST USED: 1.2 Formal performance evaluation system
   - LEAST USED: 1.4 Telecommuting
2. ORGANIZATIONAL STRUCTURE
   - MOST USED: 2.6 Professional programmers
   - LEAST USED: 2.3 Independent test group
3. DEVELOPMENT METHODOLOGIES
   - MOST USED: 3.7 Structured programming
   - LEAST USED: 3.5 pseudocode
4. SOFTWARE TOOLS
   - MOST USED: 4.4.3 Database management system
   - LEAST USED: 4.2.3 Automated slide producers
5. HARDWARE TOOLS
   - MOST USED: 5.3 Online program development
   - LEAST USED: 5.2 Teleconferencing
6. ERGONOMIC FACTORS
   - MOST USED: 6.2.1 Chairs with adjustable backrests and seat height
   - LEAST USED: 6.2.3 Source document holders

Some interesting facts have surfaced during this study. For example, of the 78 respondents only 63 or 80.8% indicated that they presently use online program development, a rather startling fact in the contemporary world of on-line real-time systems. A detailed tabulation of data describing specific sub-categories of productivity aids and their relative importance in the SDLC will be presented at the conference. This detailed tabulation will utilize the matrix approach that was first suggested by Doke and Myers (4).

Summary. This paper has reported some of the preliminary findings of a research project undertaken to identify productivity aids currently in use and the relative importance of these aids during the SDLC. Only through greater productivity can organizations continue to create and maintain the number of information systems necessary for survival in this dynamic world of automation.

References. Available upon request from the authors.

FIFTH GENERATION C.A.S.E. (COMPUTER ASSISTED SYSTEMS ENGINEERING)

Robert H. Dunikoski, Ph.D.
Blair Y. Stephenson, Ph.D.
University of Dallas
Irving, Texas 75062

ABSTRACT

Computer Assisted Systems Engineering (C.A.S.E.) systems have leaped recently to the forefront of attention in the systems development community. Vendors tout C.A.S.E. systems of all sizes, shapes and capabilities in industry publications and trade shows. In fact, the acronym "C.A.S.E." is beginning to rival "fourth generation language", "data base" and "artificial intelligence" as one of the most broadly misunderstood and misapplied terms in the MIS jargon set. The nomenclature C.A.S.E." is used to refer to everything from automated drafting tools, to code generators, to multi-function, integrated systems.

We propose two conceptual frameworks to assist in reducing confusion about whether a systems development tool is or is not a C.A.S.E. system and the type of C.A.S.E. represented. These frameworks should help clarify the scope and dimensions of current and future C.A.S.E. systems. The first conceptual framework is created by applying the concept of "generations" to the evolution of C.A.S.E. systems. The second conceptual framework is a taxonomy of the functional components which should exist in a fifth generation, "Ideal" C.A.S.E. SYSTEM.

C.A.S.E. GENERATIONS

Ken Orr, in his 1986 ISECON Keynote Address, outlined his ideas concerning generations of C.A.S.E. systems. A useful conceptual model can be developed by extending and elaborating on Orr's basic premise that C.A.S.E. systems have evolved in fairly distinct stages over the past several years. In general terms, C.A.S.E. system generations appear as follows:

GENERATION	CHARACTERISTICS
First	Auto. Drafting + W.P.
Second	Imbedded Methodology
Third	Integrated Modeling Tools
Fourth	Support Full Development + Multi-Dimensional Modeling
Fifth	A.I. + Natural Interface

FIRST GENERATION C.A.S.E.:

First Generation C.A.S.E. systems consist of simple automated drafting graphics tools plus rudimentary word processing. These graphics tools permit the computerized creation, storage and editing of various types of diagrams including flowcharts, data flow diagrams, Warnier-Orr diagrams, block diagrams, etc. Systems characteristics which cannot be described by graphics diagrams are described by some variation of word processing including narrative text, tables, pseudocode, etc. These graphics tools and word processing tools are not integrated, but operate separately from each other.

SECOND GENERATION C.A.S.E.:

Second Generation C.A.S.E. systems are extensions of the first generation, but they differ significantly in that they incorporate a specific systems development methodology; e.g., Yourdon, DeMarco, Gane and Sarsen, Warnier-Orr, James Martin, etc. In addition, these Second Generation C.A.S.E. systems usually exhibit more sophisticated graphics, data dictionaries and encyclopedic data bases. The graphics are typically derived from the foundation methodology (e.g., data flow diagrams) and incorporate a limited number of rules for constructing those specific diagram models. The data dictionary tools include certain fairly strict conventions for definitions, usually geared toward programming level data definitions. The dictionaries may

also include some rudimentary cross-checking abilities such as 'next higher', 'next lower', 'find aliases', etc. Often, these Second Generation C.A.S.E. systems will include a primitive encyclopedia data base which serves as a repository where the C.A.S.E. systems user can store any or all copies of each entity modeled in the system(e.g., each diagram, process description, etc.).

THIRD GENERATION C.A.S.E.:

Third Generation C.A.S.E. systems extend the imbedded methodology characteristics of the second generation, but they demonstrate initial attempts to integrate the various modeling facilities in the C.A.S.E. into a more unified whole. For example, a data flow or data store identified on a data flow diagram is automatically entered in the data dictionary. Changing a data structure name in the data dictionary automatically changes that data name everywhere it occurs in graphics models, data dictionary definitions, data models and process logic descriptions. Third Generation C.A.S.E. systems typically provide more sophisticated encyclopedias and data modeling tools. The C.A.S.E. encyclopedia is integrated with all other modeling facilities in the C.A.S.E.; automatically recording each iteration of entity modeled, and providing some limited cross function analysis (e.g., data conservation testing of transformations using data flow diagrams and data content definitions from the data dictionary). The data modeling tools include at least primitive hierarchical, network, or third normal form (object-oriented) data modeling capabilities.

The most obvious shortcoming of Third Generation C.A.S.E. systems is their inability to fully service all stages of the systems development and maintenance effort. This problem seems evident when viewing the three major groupings of typical third generation C.A.S.E. systems:

1. Analysis and Design C.A.S.E. Systems
2. Applications Generators
3. Fourth Generation Language Prototyping Tools

FOURTH GENERATION C.A.S.E.:

Fourth Generation C.A.S.E. system are distinct from Third Generation C.A.S.E. systems in three areas:

1. Fourth Generation C.A.S.E systems attempt to support the entire span of systems development from early concept formalization, to analysis and design modeling, the physical systems modeling, to application generation, to follow-on maintenance and enhancement. Higher order software's "Use. It" might be an example of this type C.A.S.E. system.
2. These systems incorporate at least primitive attempts to accomplish true multidimensional modeling, integrating various single-dimensions models based on data, function, events, logic, time, etc. These systems usually incorporate at least limited forms or expert systems to support both single-dimension and multiple-dimension modeling facilities.
3. Fourth Generation C.A.S.E. systems incorporate rudimentary network modeling tools to support modeling the desired system into a physical system environment; especially into environments characterized by multiple distributed processing nodes, data bases, and data communications facilities.

FIFTH GENERATION C.A.S.E.:

Fifth Generation C.A.S.E. systems extend fourth generation systems significantly through the extensive use of artificial intelligence. The A.I. portion of

these systems will use expert systems approaches and inference reasoning approaches to support all modeling, analysis and implementation activities of the C.A.S.E. system. The A.I. module will use an exhaustive encyclopedia which contains all historical and current knowledge about the prior system and the desired future system. In addition, the A.I. model will use an extensive collection of rule sets which include desired or accepted modeling methodologies, custom rule sets, and systems generated rule sets based upon user specified priorities and objectives. A Fifth Generation C.A.S.E. system could apply a set of very specific methodology rules using the expert systems mode; or create an optimized set of modeling rules based on user generated objectives and global (across methodologies) rules when placed in an inference reasoning mode.

Fifth Generation C.A.S.E. systems must also incorporate very advanced "proptotyping" capabilities; i.e., the ability to combine many single-dimension models into one multiple-dimension model. These systems must enable the user to begin creating full-scale systems prototype models based on very early design attempts in one or more single-dimension models. (This requirement appears to be intuitively obvious when examining the process of systems development or acquisition in other environments such as building a home or buying a car.) The fundamental idea is to generate a continuous iteration of multiple-dimension, systems prototypes from the concept definition phase of development through the actual operational implementation of the system. Thus, the final iteration of the prototype would be converted to an operational production system by the C.A.S.E system using user supplied specifications for the target physical environment (machine architecture operating system,, program language, data base structure, data communications protocols, etc.).

Another key distinction of Fifth Generation C.A.S.E. systems will be their man-machine interface facilities. In order for C.A.S.E. systems to reach their full potential impact on organizations, they must provide extreme ease of use interfaces for their human users. These human users will vary from the local user who knows almost nothing about systems to the very sophisticated systems professional. A.I. will be used extensively to support highly interactive communications between the user and the C.A.S.E. system. The system must assess the level of user capability and respond accordingly. This man-machine interface will probably be characterized by interactive query/response, tutorials, color graphics, voice input/output, touch screen derivatives, visually directed communications and two-way interactive video.

Fifth Generation C.A.S.E. systems will also incorporate powerful modeling concepts which are still in their early formative stages or on the leading edge of systems development thought. For example, third normal form data modeling (object-oriented data models), object-oriented graphics, state-transition and event analysis modeling, and systems design and creation from mathmatically provable logic constructs.

C.A.S.E. SYSTEMS FUNCTIONAL COMPONENTS

A Fifth Generation C.A.S.E. system must interact with three primary external entities; the design group (including users and analysts), management decision makers and the production system environment. (See Figure 1, "C.A.S.E. Context Diagram") The design group feeds a variety of data flows into the C.A.S.E. system including:

* Features and problems of the current system
* User requirements and objectives for the new system

* Network design requirements for the new system
* Physical implementation requirements for the new system
* Methodology and decision rules for systems analysis, design and construction
* Requests for analysis and evaluation of systems models and prototypes

Interactions between management and the C.A.S.E. system focus primarily on management review and approval processes. Management reviews design requirements, objectives and priorities; then executes policy decisions for approval or revisions. Management performs a similar review and approval decision process for progressive iteration of new system prototypes.

The interaction between the production systems and the C.A.S.E. system is fairly straightforward. The C.A.S.E. system reads the parameters of the production systems (e.g., operating system, DBMS, communication protocols, application language, etc.). The C.A.S.E. system converts the final approved prototype into a production ready application and passes it to the production system for operation.

Although, this explanation of the context of a C.A.S.E. system is greatly simplified, it should establish an overview perspective as to how a fifth generation C.A.S.E. system should fit into an organization.

C.A.S.E. SYSTEMS - LEVEL 0 PERSPECTIVE

When we drop down one level, we can see the major sub-system components of the "ideal" C.A.S.E. system. (See Figure 2, "C.A.S.E. Level 0 Diagram") First, the C.A.S.E. system contains a series of modeling facilities which are single-dimensional or limited dimension modeling tools. At a minimum, we would suggest that these include tools to model functions, data, logic and events/state transitions. These single-dimensional modeling facilities interface with the design group and with the "Intelligence" component of the C.A.S.E. system.

Second, the C.A.S.E. system will contain systems modeling facilities which will enable the designers to model the network aspects of the system and create multi-dimensional prototypes of the new system. These system modeling facilities will interface with the "intelligence" facility, the design group and, finally the production system environment.

Third, the C.A.S.E. system will contain an analysis/evaluation facility which will read iterations of models and prototypes from the encyclopedia and critically test these models against designated rule sets. These analyses will assist both the design group and management in making judgments concerning the progress, strengths and weaknesses, failures, trade-offs, etc. evident in old or new systems.

Fourth, the C.A.S.E. system will contain a very critical "intelligence" facility. This facility will include two major capabilities; an encyclopedia and a rules plus reasoning system. The encyclopedia maintains all historical and current information about current systems and new systems. This information would include all single and multi-dimensional models, details of physical dimensions and requirements, etc. The rules and reasoning system would contain all appropriate rule sets, statements of objectives, priorities, policy statements, constraints/limitations, etc. This component would operate in oth an expert systems mode as well as a reasoning mode. It would apply the user designated (or default) methodology-specific rule set(s) (i.e., Martin's Information Engineering); or a user-designed customized rule set; or a user-requested systems generated rule set. The rule set(s) may be applied to either single-dimensional models or to multi-dimensional systems models/prototypes; or to both. In

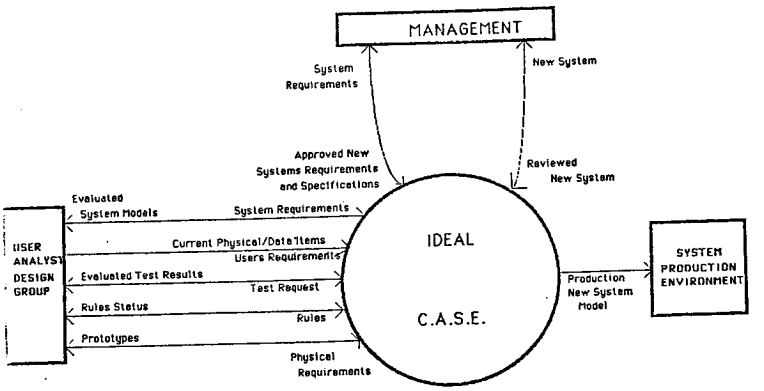
the reasoning mode, the intelligence facility could create the optimum rule set to apply based on user and management supplied objectives, priorities, constraints, parameters, etc. plus rule sets available from the rules base.

C.A.S.E. SYSTEMS - LEVEL 1 PERSPECTIVE

By dropping down one more level, the individual sub-components and their relationships become more apparent. (See Figure 3, "C.A.S.E., Level 1 Diagram") A modeling facilities (functions, data, logic and events/state transitions) and their interfaces become visible. The position and interfaces of the multi-dimensional network modeling and prototyping facilities also become better defined. The same is true for the analysis/evaluation facility and the rules update function. This Level 1 diagram presents the major distinct components or facilities of the fifth generation "Ideal" C.A.S.E. system. The section which follows will present the specific functions and interfaces of each of these individual components or facilities in both tables and diagram forms.

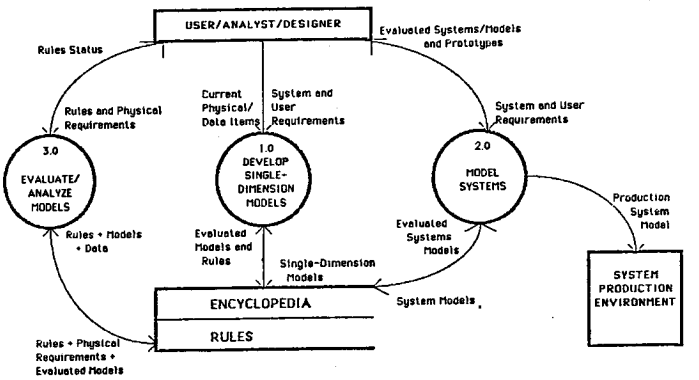
SUMMARY

This paper has attempted to make two contributions to the literature. First, the idea of generations of C.A.S.E. technology has been expanded and further defined. Second, a conceptual framework for the ideal C.A.S.E. system has been defined. Research is presently being conducted to evaluate the current C.A.S.E. systems in light of the proposed framework.



CONTEXT DIAGRAM

FIGURE 1



LEVEL 0 DIAGRAM

FIGURE 2

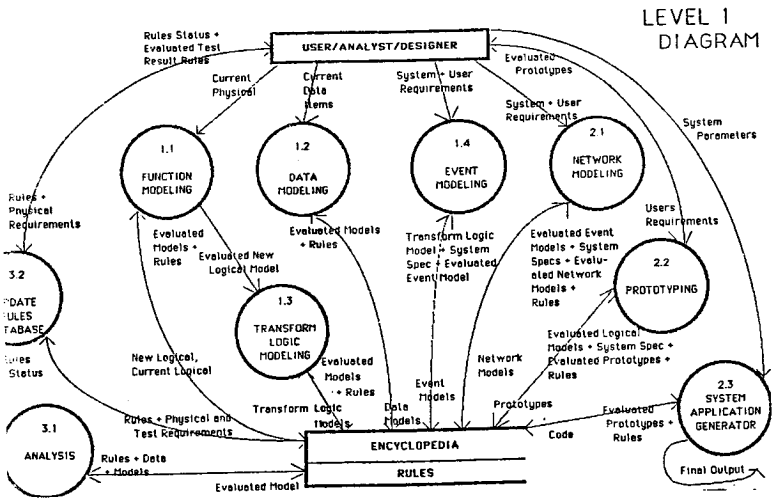


FIGURE 3

END USER EDUCATION SURVEY
Implications for Educators

by

Dr. David B. Armbruster and Mary L. Amundson-Miller
Department of Management Information Systems
University of Wisconsin-Eau Claire

Bret R. Ellis
Department of Information Systems and Computer Science
Brigham Young University-Hawaii Campus

ABSTRACT

The purpose of this study is to help us, as educators, determine what Information Systems skills and knowledge are important for a business student (non-Information System major) to obtain while attending college. The findings indicate that graduates have required considerable training, on-the-job, to be effective end-user computing professionals.

BACKGROUND

"In 1985 there were an estimated 20 million electronic work-stations (word processors, personal computers and terminals) in use in US business, one for every three white collar workers. While the ratio was one to seven in 1980, it may be one to one by the end of the decade." (1) "End-user computing is becoming more than a mainstream activity; it is the mainstream activity." (2) As educators in Information Systems at the University of Wisconsin-Eau Claire, we provide a core course for the School of Business entitled Information Systems in Business. Every student that majors or minors in Business Administration (BBA - Accounting, Finance, MIS, etc.) is required to take this course, which amounts to approximately 1,000 students per year. In most cases it is the only course in Information Systems that these students take. With class sizes averaging 38 students, this amounts to about 27 sections per year. In order to meet the demand, many of the 10 full-time faculty in the MIS department teach this course. Because of this, the course content varies from instructor to instructor, and agreement on a common course outline is hard to obtain.

As educators, we are faced with the dilemma of selecting the most important material to be presented in this survey course to prepare the Business students (future end-users) for their professional careers. Following AACSB guidelines, evaluating new introductory texts, and current literature in the area provides some direction. But, we find that covering the conceptual material that is considered important and giving the students working experience on the microcomputers tends to make for an extremely full semester. We wonder if we are trying to pack too much material into the three credit course, making it ineffectual in the preparation of an end-user for industry. An evaluation of the current and possible course content is needed to sift out the nice-to-know's from the

necessary-to-know's. We also wonder if it is possible to present the necessary material in one three-credit semester course.

A search of the current literature supported our belief that most School of Business majors would be future end-users, but it did little in the way of giving us specific information about exactly what knowledge and skills are needed by the typical end-user in industry. We concluded that the best way to decide what to present in this course, to future end-users, would be to survey Information Center managers and end-users in industry.

PROCEDURES

A total of 140 companies were selected to participate in this study because of their affiliation with one of the following professional information center organizations: Association of Information Center Professionals (AICP), Minnesota; Saint Louis Information Center Exchange (SLICE), Missouri and Illinois; and Wisconsin Information Center Users' Group, Wisconsin and Illinois.

The Information Center managers were asked to complete a survey and distribute surveys to three representative end-users in the company. For purposes of this study, end-users were defined as professionals who used a computer at least once a week to obtain information essential to their job function. Thirty-one Information Center managers, 24 percent, responded to the survey; and 75 representative end-users of a possible 420, or 18 percent, responded.

The survey consisted of demographic and general questions, but was mainly composed of three sections, each containing a table. Each table listed 20 Information Systems topical areas. The topic area was further divided into (a) conceptual knowledge of the topic and (b) working skill (the application of the conceptual knowledge).

Section 1 asked the respondents to evaluate the importance of each topic to the end-user. A Likert-type scale of unimportant to extremely important was used.

Section 2 asked the respondents to indicate where (high school, technical school, college, within their own department, information center, seminars/workshops, or other) they felt the end-users had acquired the largest percentage of their conceptual knowledge and skill in the topical areas.

In section 3, the Information Center managers and current end-users were asked where they felt the future end-users should obtain their knowledge or skill for each of the topical areas.

FINDINGS

On the average, the Information Centers in the sample population had been established for three years. Of the 75 end-users who responded, 49 percent worked in Sales/Marketing, Finance, Accounting, or Operations. Seventy-seven percent had earned a college degree (Bachelor, Master, Ph.D.). Approximately 49 percent indicated that their major field of study, while in college, was business related. The remainder of the findings is divided into three sections: (1) Working Skill, (2) Conceptual Knowledge, and (3) Comparison of Working Skill and Conceptual Knowledge.

WORKING SKILL

The findings indicate, as illustrated in Table I, that:

- over 50 percent of the Information Center managers believe that a working skill with microcomputer operating systems, statistics pack-

ages, systems analysis and design, spreadsheets, financial modeling, graphics, backup, security and control, data base structures, data management, nonprocedural languages, and file transfer techniques is important for an end-user.

- over 50 percent of the end-users believe that a working skill with microcomputer operating systems, systems analysis and design, spreadsheets, financial modeling, graphics, backup, security, and control, data base structures, data management, nonprocedural language, file transfer techniques, mainframe operating systems, data communications, and integrated software is important for an end-user.

- over 50 percent of both the Information Center managers and the end-users believe that a working skill with procedural languages, distributed processing, and local area networks were unimportant for end-users.

- of the topics ranked important, 40 percent of the managers indicated that a working skill in microcomputer operating systems should be acquired in formal education; whereas over 40 percent of the end-users indicated that a working skill in systems analysis and design and spreadsheets should be acquired in formal education.

TABLE I

WORKING SKILL

TOPIC RANKING AND EDUCATION'S ROLE IN PREPARING END-USERS

	MANAGERS:			END-USERS:		
TOPICAL AREAS	Ranking	Percent Acquired in School	Percent Believed Should Have Been Acquired in School	Ranking	Percent Acquired in School	Percent Believed Should Have Been Acquired in School
Microcomputer Operating Systems	I	3	40	I	8	37
Statistics Package	I	8	33	U	16	29
Systems Analysis and Design	I	6	31	I	16	45
Spreadsheets	I	0	30	I	3	42
Financial Modeling	I	10	30	I	8	30
Graphics	I	3	20	I	3	33
Backup, Security, and Control	I	0	17	I	5	28
Data Base Structures	I	7	17	I	12	24
Data Management	I	3	13	I	7	25
Nonprocedural Language	I	0	10	I	6	16
File Transfer Techniques	I	0	3	I	5	26
Procedural Language	U	22	45	U	48	54
Mainframe Operating Systems	U	0	13	I	8	33
Data Communications	U	0	10	I	6	25
Distributed Processing	U	7	10	U	9	20
Integrated Software	U	3	7	I	4	21
Local Area Networks	U	0	3	U	4	13

Table I itemizes the topics addressed in the survey. The rankings columns indicate whether over 50 percent of the respondents considered the working skill to be important (I) or unimportant (U).

- of the topics ranked important, over 30 percent of the managers indicated that a working skill in statistics packages, systems analysis and design, spreadsheets, and financial modeling should be acquired in formal education; whereas over 30 percent of the end-users indicated that a working skill in microcomputer operating systems, graphics, mainframe operating systems and financial modeling should be acquired in formal education.

CONCEPTUAL KNOWLEDGE

The findings indicate, as illustrated in Table II, that:

- over 50 percent of the Information Center managers believe that a conceptual knowledge of financial modeling, systems analysis and design, statistics packages, data base structures, word processing, spreadsheets, graphics, data management, microcomputer operating systems, integrated software, backup, security, and control, data communications, non-procedural language, and file trans-fer techniques is important for an end-user.

- over 50 percent of the end-users believe that conceptual knowledge of financial modeling, systems analysis and design, statistics packages, data base structures, word processing, spreadsheets, graphics, data management, microcomputer operating systems, integrated software, backup, security, and control, data communications, nonprocedural language, file transfer techniques, local area networks, and mainframe operating systems is important for an end-user.

TABLE II
CONCEPTUAL KNOWLEDGE
TOPIC RANKING AND EDUCATION'S ROLE IN PREPARING END-USERS

	MANAGERS:			END-USERS:		
TOPICAL AREAS	Ranking	Percent Acquired in School	Percent Believed Should Have Been Acquired in School	Ranking	Percent Acquired in School	Percent Believed Should Have Been Acquired in School
Financial Modeling	I	31	59	I	12	46
Systems Analysis and Design	I	24	59	I	26	57
Statistics Package	I	14	55	I	20	51
Data Base Structures	I	13	53	I	20	50
Word Processing	I	6	52	I	7	52
Spreadsheets	I	3	52	I	7	46
Graphics	I	0	52	I	4	39
Data Management	I	3	50	I	8	43
Microcomputer Operating Systems	I	0	50	I	8	50
Integrated Software	I	3	49	I	6	38
Backup, Security, and Control	I	3	47	I	15	42
Data Communications	I	3	43	I	10	44
Nonprocedural Language	I	3	42	I	9	35
File Transfer Techniques	I	0	32	I	7	44
Local Area Networks	U	3	49	I	8	30
Mainframe Operating Systems	U	0	48	I	14	39
Procedural Language	U	47	40	U	53	71
Distributed Processing	U	17	39	U	14	54

- over 50 percent of both the Information Center managers and the end-users believe that a conceptual knowledge of procedural languages and distributed processing is unimportant for an end-user.

- over 50 percent of the Information Center managers indicated that a conceptual knowledge of local area networks and mainframe operating systems was unimportant; whereas over 50 percent of the end-users felt the same knowledge was important.

- of the topics ranked important, over 50 percent of the managers indicated that a conceptual knowledge of financial modeling, systems analysis and design, statistics packages, data base structures, word processing, spreadsheets, graphics, data management, and microcomputer operating systems should be acquired in formal education; whereas over 50 percent of the end-users indicated that a conceptual knowledge of systems analysis and design, statistics packages, data base structures, word processing, and microcomputer operating systems should be acquired in formal education.

- of the topics ranked important, over 40 percent of the managers indicated that a conceptual knowledge of integrated software, backup, security, and control, data communications, and nonprocedural language should be acquired in formal education; whereas over 40 percent of the end-users indicated that a conceptual knowledge of financial modeling, spreadsheets, data management, backup, security, and control, data communications, and file transfer techniques should be acquired in formal education.

- of the topics ranked important, over 30 percent of the managers indicated that a conceptual knowledge of file transfer techniques should be acquired in formal education; whereas over 30 percent of the end-users indicated that a conceptual knowledge of graphics, integrated software, nonprocedural language, local area networks, and mainframe operating systems should be acquired in formal education.

COMPARISON OF WORKING SKILL AND CONCEPTUAL KNOWLEDGE

A comparison of the findings indicates, in general, that the percentage of managers and end-users that feel these topics should be taught in formal education is higher on Table II (Conceptual Knowledge) than on Table I (Working Skill).

IMPLICATIONS

(1) An arbitrary cutoff based on 40 percent of manager or end-user respondents, indicates that working skill in microcomputer operating systems, systems analysis and design, and spreadsheets should be presented by formal education.

(2) An arbitrary cutoff, based on 40 percent of manager or end-user respondents, indicates that conceptual knowledge in financial modeling, systems analysis and design, statistics packages, data base structures, word processing, spreadsheets, graphics, data management, microcomputer operating systems, integrated software, backup, security, and control, data communications, nonprocedural language, and file transfer techniques should be presented by formal education.

(3) The study indicates that the topic most learned through formal education is procedural language. It's interesting to note that both managers and end-users consider it unimportant. It would be interesting to know how many introductory information systems courses include units on procedural language.

(4) As was noted earlier, over 50 percent of the end-users in the

sample graduated outside of the School of Business; therefore it may be concluded that the need for conceptual knowledge and working skills in information systems is important for non-business students also.

(5) If a three-credit introductory course doesn't allow time for all the topics, as listed under implications one and two, other avenues for computer-based training should be explored.

(6) This study was intended to help us find topics that were considered valuable to make end-users more effective. More study can now go into determining what specifically could be covered within the recommended topical areas.

REFERENCES

(1) Arthur Andersen & Co., Trends in Information Technology: 1986, Chicago: January 1986, p. 10.

(2) Zink, R. A., "The Tilt to End-User Programming," Computerworld, Volume 19, Number 30, July 1984, pp. 10-14.

Using a College Structured Programming Service Course as a Model for Training Industry Professionals

by Helene P. Baouendi and Alka R. Harriger
Computer Technology Department
Purdue University
West Lafayette, IN 47907

ABSTRACT: Today's computer age requires most individuals to have some working knowledge of the computer as a tool for their jobs. Many industry professionals with expertise in non-computer related areas may now have to learn to use the computer. A course for this audience should be specifically adapted to their unique expertise area. For the last two years, the Computer Technology Department at Purdue University has offered specially designed courses to students taking various majors. The majority of assignments were directly related to the students' major fields of study. This paper will describe our approach for teaching structured programming to audiences with expertise in non-computer areas. Although most of the discussion centers on a college audience, our approach is being used successfully in a large company in our city.

INTRODUCTION

The computer revolution has reached such great proportions that most children (grades K-12) are learning to use the computer in many of their subject areas. Reinhold confirms this trend in "Electronic Learning's 1986 Annual Survey of the States." [1]

Many people feel that the youth of this country know more about the computer than their older counterparts. This has created a great demand for retraining industry professionals in the area of computing. But, as Reinhold quotes Jeanelle Leonard of the District of Columbia, "it is a deliberate and slow process to remove the fears of and train personnel in automation." [1] This statement could easily be applied to the non-computing major taking a computer course.

The School of Technology at Purdue University has over 3300 students enrolled in eight departments. In addition to the the main campus teaching activities, the school is also sponsoring similar programs at regional campuses and local industries throughout the state. Our primary goal is to offer a uniform curriculum statewide.

Most technology students are required to take an introductory computer service course. Depending upon their major, students would take either computer programming or computer literacy. In the past, the students took general service courses offered for non-majors at their local campus. The technology faculty felt this training was inadequate since:

- There was no uniform curriculum for non-computer majors.
- The student audience was varied; thus, the class examples and assignments were general in nature and did not apply to the fields of study of our technology students.

Two years ago, the Computer Technology Department (CPT) at Purdue University began offering service courses to students in our sister departments within the School of Technology. In agreement with Adair et.al., we felt

we had a "responsibility to provide appropriate instruction and support beyond the level of computer literacy." [2]

To be successful, we had to accomplish, as Karten put it, "the most important objective of user training: To prepare business professionals to gain business benefit from the use of computers." [3] We felt that the service course should satisfy the specific needs of the different departments. In addition to giving an introduction to the computing discipline, our students more importantly needed to learn the applications of computing in their own fields.

Although CPT's service role has grown, initially two groups of computer service courses were offered: computer literacy and structured programming. This paper discusses the initial development of these courses and focuses on our approach in designing assignments for the structured programming course. Without the constant interaction between CPT instructors and department representatives, this could not have been accomplished so successfully. Our approach in designing these service courses has been used successfully in its current form at the industry level.

INITIAL DEVELOPMENT

The CPT Instructional Development Initiatives Committee prepared a survey that was sent to all other departments in the school. The results revealed a common goal: the students should be able to use the computer as a tool in their fields of study. The approach for the curriculum was clearly divided between three types of courses: computer literacy, structured programming, and hardware organization.

Everyone interested in the computer literacy course wanted the coverage to include fundamental computer concepts and terminology with some minimal programming. They also wanted an emphasis placed on the use of computer software packages. Since all groups seemed to want Halaris and Sloan's definition of the second level of computer literacy, a single course was created. [4]

The hardware organization course was requested by a single group. It was to be a continuation of the introductory programming course. Both a conceptual understanding and assembly language programming were desired. Since all requirements came from one group, one course was created.

The people interested in the programming course wanted the same type of information covered, but with different languages and applications. Task force committees with representatives from CPT and our sister departments were created. Each group had to determine which programming language should be taught and recommend appropriate class assignments. Now that the courses have been implemented, these committees still meet in an informal way.

The students from the Electrical Engineering Department (EET) are taught structured programming concepts and applications using the BASIC-PLUS language available on the PDP 11/70, RSTS/E system. Computer Integrated Manufacturing Technology (CIMT) students learn structured programming through FORTRAN-77 on the same system.

Regardless of the language used, all of the programming courses place great emphasis on problem solving and algorithm design. The students receive much practice applying newly learned concepts through class examples and programming assignments. The assignments are designed so that they test computing concepts in the context of a problem in the students' fields of study. Thus, we maintain the students' interest as well as show the computer as a tool in their chosen fields.

EXERCISE AND PROJECT EXAMPLES

The students have two types of assignments in the programming course: exercises and projects. Exercises are fairly short programming or nonprogramming assignments. Projects are quite detailed programming assignments which are completed in two phases. In the course of a semester, each student completes 7-9 exercises and three projects.

The instructor determines the computing conceptual material to be tested by a particular assignment. After conferring with the appropriate departmental committee for sugges-

tions on application topics, the instructor creates the programming assignment. Ideally, the assignment would be one the student has been (or would be) exposed to in his/her major area classes. The following are some of the problems we have used and a discussion of their relationships to the specific fields.

EET majors use complex numbers a great deal in their major. For the classical quadratic-equation problem, the EET majors were asked to write a program to compute and print the real roots of the equation if they exist and the complex roots if no real roots exist.

For the classical least squares problem, EET students were given as input data for their program, the results of an experiment performed in the "Process Control" course. Some of the students had already taken this course and shared their experience with the other students.

Another project required the EET students to write a program to obtain the frequency response of an alternating current circuit using the polynomial ratio H(jw):

$$H(jw) = \frac{(a_0 - w^2 a_2) + jwa_1}{(b_0 - w^2 b_2) + jwb_1}$$

where the a_i's and b_i's are constants and $w = 2\pi f$ is the frequency variable.

This example was tested in the "Advanced Electrical Circuits" course. Experimental values were used for the a_i's and b_i's. Again, some students had already taken the "Advanced Electrical Circuits" class and were eager to share their knowledge with the others.

The CIMT students were given a programming assignment to compute the manufacturing lead time, average product time and average product rate for a manufacturing plant. This example was already seen by many of the students in their "Automated Manufacturing" class.

CIMT majors also wrote a program related to the machining process known as milling. The formulas and values for coefficients were found in the actual text they used in their "Automated Manufacturing" course.

By doing assignments related to their field of study, students showed more interest in their work. They also realized that their homework had immediate applications and would be useful in the "real world." The computer is viewed as a tool, so they see the utility of what they are learning.

THE TWO-PHASED PROJECT

The emphasis of the programming course is program designing, not coding. The students' project assignments are sufficiently complex that writing the program code without prior algorithm design is an infeasible task. We reinforce the design-before-code sequence by requiring submission of the projects in two parts: the design report, and the completed project.

The major part of the project is the design report. This report must be professionally done, with a title page and appropriate formatting. Very few of these students will be professional programmers, but most of them will use the computer as a tool and will use the help of computer specialists. They will need to communicate their needs to the computer specialist, so they must learn to clearly define the problem.

The five parts of the design report confirm the student's understanding of an assignment. The description/scope must clearly define the project. The student is reminded that the reader may know nothing about the assignment and subject; therefore, the description must include all formulas and units used.

The input and output sections are format specific. The student must show the source/destination of the data (CRT or file) and the exact format, including all placement of literals.

The test data section again reminds the student that the programmer is not a specialist in the student's field. Therefore, the input data to test the program as well as the anticipated results must be supplied. Data should cover not

only the general cases, but also the extreme limits.

The algorithm design section is the most important. Here, the student must break down the problem solution into a logical sequence. The steps are outlined using pseudocode.

All five parts are put together as a professional document. After the graded design document has been returned, the student needs to write the necessary structured code following the course guidelines. The program must follow the design and logic contained in the design report. If any changes are made, the report must also be revised. Finally, the revised report and program are resubmitted for a grade.

CONCLUSION

The computer age has created a great demand for programs which teach computing concepts to the industry professional. The structured programming course discussed in this paper provides a vehicle for computer instruction in the context of the students' fields of study. Although our discussion describes it as a college course, it has been successfully transported "as is" to a large company in our city. By using our practical methods, the students not only retained the computer programming principles, but also realized the immediate applications within their own fields.

REFERENCES

[1] Reinhold, Fran. "Computing in America: Electronic Learning's 1986 Annual Survey of the States." Vol. 6, No. 2, Electronic Learning, October 1986, pp. 26-30, 32-33, 69.

[2] Adair, James H. et.al. "Computer Science as a Service Department: What do we Offer Non-majors?" Vol. 18, No. 1, ACM SIGCSE Bulletin, February 1986, pp. 149-150.

[3] Karten, Naomi. "End User Demand Requires New Approach to Training." Vol. 24, No. 5, Data Management, May 1986, pp. 10-12, 19.

[4] Halaris, Antony and Lynda Sloan. "Towards a Definition of Computing Literacy for the Liberal Arts Environment." Vol. 17, No. 1, ACM SIGCSE Bulletin, March 1985, pp. 320-326.

A CORPORATE UNIVERSITY LEVEL COMPUTER SCIENCE PROGRAM

Gary L. Bringhurst
Robert P. Burton
Brigham Young University
Provo, Utah 84602

Ira E. Heiney
IBM Corporation
Austin, Texas 78758

Abstract

International Business Machines (IBM) Corporation has designed, developed and implemented a University Level Computer Science Program. Employees take intensive one week courses selected from the curriculum of twenty courses with the goal of achieving bachelor-level equivalence. No college credit is earned, but recognition is given within the company. The corporate goal of maintaining a progressive programming community is met. Faculty from respected universities teach the courses, bringing contemporary university perspective into the company.

Participants in the Program complete the "Computer Science Fundamentals" series of three courses the first year, and two core courses in each succeeding year. A participant is able to achieve bachelor-level equivalence in five years.

INTRODUCTION

A University Level Computer Science (ULCS) Program has been developed and implemented by International Business Machines (IBM) Corporation. The Program brings the latest computer science material taught in leading universities into IBM. It complements existing tuition refund and graduate work study programs by providing an alternative means of obtaining university level instruction for employees who cannot devote ten to fifteen weeks to attend a full university quarter or semester. It differs from other programs in that the student does not earn credits toward a degree as a result of completing courses.

The ULCS curriculum consists of twenty full week courses. Some courses are related hierarchically as indicated on the course schematic (figure 1). By taking the three, one week long courses in the "Computer Science Fundamentals" (CSF) series the first year and two one week courses in each of the succeeding four years, an employee can achieve the knowledge equivalent of a computer science bachelor's degree graduate in five years. Some of the curriculum is relevant for recent computer science graduates, depending on the content of courses they took while in college.

Each course is designed to cover a semester of material in five days. Some courses are combined into a series, e.g. "Computer Science Fundamentals." Each week of a series is separated from the succeeding weeks by four or five weeks to avoid the impact of employees being away from their departments for lengthy periods.

To assure appropriate instruction, each course is developed and taught by a team of university professors. This brings non-IBM viewpoints to the classroom, utilizes leading computer science educators and increases the level of interaction between IBM and universities. The team teaching approach makes possible extensive individual assistance, accommodating the needs and interests of both slower and faster students.

Courses are offered at an intense, attention consuming level. Both students and faculty must be free of the distractions of office, home and other interests. Courses offered off-site with students and instructors resident at a conference center are ideal. The schedule of activities (including

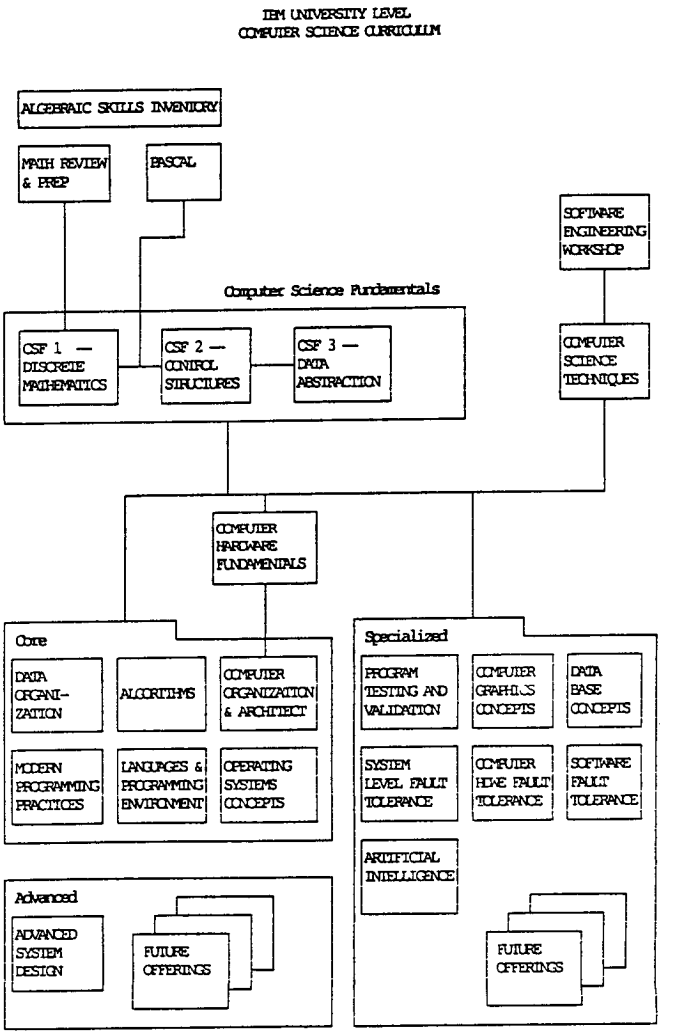


Figure 1

lectures, laboratories, homework problems/solutions, quizzes and recreation) occupies virtually every waking moment to maximize learning with minimum time away from the job. Each of the week long courses consists of lectures and homework Monday through Thursday, with evaluations and an examination on Friday. This teaching approach has been evaluated by an independent consultant, students, instructors and course managers, and has been found to produce significant, effective, applicable learning and retention.

The Program is of particular benefit to an employee who has been away from formal education five or more years and to the recent computer science graduate who did not take some of the courses in the ULCS curriculum while in college. Programmers and managers of programmers are natural candidates for these courses. Consideration is also given to engineers, planners, writers and other employees doing microcoding and/or other programming related work.

While the content of each course is significant and voluminous, content is not the sole objective of the curriculum. A major goal is to provide improved perspectives, approaches to the job and, especially, ways to think about the programming process. Much attention is given to improving the process and product early in the development cycle, avoiding more costly diagnosis and repair during later stages. The overall thrust is to add contemporary and prospective productivity and quality assurance tools to the employee's capabilities.

PROGRAM HISTORY

The ULCS Program began in 1980 with the development of the CSF series by a group of professors from Carnegie Mellon University. These professors also prepared a core curriculum. As a means of keeping the large programming community within IBM technically current, the Technical Education Department at IBM has orchestrated its development into its present form. A long standing tuition refund program is available to employees at IBM, but most employees continue to find it difficult to commit to a 10 to 15 week university course.

During 1985, 79 total classes representing 21 distinct courses were offered. A total of 71 classes are scheduled for 1986, representing 20 courses.

PROGRAM OBJECTIVES

The ULCS Program has several specific objectives. For the student, the objective is to provide an opportunity for professional growth and personal satisfaction, with the aim primarily toward the experienced programmer. For the company, the objective is to maintain a programming community that is technically up-to-date, competent and productive. Outside the corporation, the objective is to establish and maintain ties with universities, increasing the traditional level of interaction.

As a unit, the three courses in the introductory CSF series have four general objectives:

1. To read, understand and use basic concepts and related notation;
2. To read and understand the more common notation appearing in software engineering textbooks, literature, designs and specifications;
3. To understand basic proof techniques and their roles in software engineering and computer science; and
4. To understand the roles of abstraction, formalism and mathematical rigor as tools for computer science.

Two broader goals are implicit in these objectives:

1. To prepare students thoroughly for later courses in computer science and
2. To confirm a view of computer science as a profession (i.e., a view of programming as "engineering based on science" and a view of programmers as professionals with a relevant professional literature).

The affective element of this second broader goal is that students come to appreciate and value the science of programming.

ORGANIZATION

ADMINISTERING THE PROGRAM

Departments of Information Systems and Product Group (IS&PG) Technical Education are responsible for advanced education in fields such as manufacturing, engineering, planning and programming. The ULCS Program is administered by one of two IS&PG Technical Education departments offering educational opportunities to the programming community within IS&PG. The Programmer Education Department concentrates on entry level training and programming skills training. The University Programs Department focuses on curricula designed to raise the knowledge level of experienced programmers in current computer science theory and techniques.

The Manager of the ULCS Program supervises resident class managers at seven major development sites. Each resident class manager is responsible in his/her own location (and in some secondary sites) for scheduling classes, recruiting students, contracting faculty and general administration of the Program. Each class manager is the "owner" of a number of courses and works closely with the university faculty who develop and maintain the courses. This ensures that course material is current and transportable (i.e., that it can be taught by other faculty while maintaining consistency across all sites).

Users are billed internally at a student/day charge sufficient to recover costs. The user's willingness to pay serves as an additional control and a value measurement for the courses.

SELECTING THE COURSES AND COURSE CONTENTS

Several methods are used to select courses for the ULCS curriculum. The ACM and IEEE core recommendations have been used to select, design and organize courses. The curricula of leading universities have been used as models. Suggestions are obtained from class managers who oversee courses, from instructors who teach courses and from students who take courses. An advisory board composed of university faculty conducts regular reviews with the Manager of the ULCS Program.

A study has been performed comparing the ULCS Program with the CS programs of a dozen major universities. The general observation is that the ULCS Program provides a good educational background in computer science for IBM employees. Regular university CS programs offer a broader base of computer science material and exposure to specialized areas in computer science. The ULCS Program suffers from lack of depth due to time constraints. Broad-based mathematics material, such as numerical methods, logic, recursion, combinatorics and theory of computer science, which often is required in a university CS program, is not included in the ULCS Program.

Maintaining the ULCS curriculum is an iterative process. Any time a need for either a new course or modification of an existing course is determined, the curriculum is upgraded. Significant course changes are reviewed and approved by class managers and the curriculum review committee. Changes are implemented at all sites simultaneously, since the responsible class manager controls the master copy of

course materials. If a course has outlived its usefulness, it is retired. In this way the curriculum is kept in step with advances in computer science. The Program must remain timely in order to achieve its objectives.

SELECTING THE STUDENTS

As stated above, the main thrust of the ULCS Program is toward experienced programmers. Each course has prerequisite courses which generally are enforced, although they are relaxed in cases where the prospective student has equivalent prerequisite knowledge. Firm prerequisites are minimized to accommodate the logistics of providing a course, including student availability, the location of class, etc.

Some students have been allowed to take selected courses without satisfying the prerequisites (e.g. Artificial Intelligence and Operating Systems). In these cases, simple exposure to the subject matter has been the objective of the student's participation.

Most students have adequate backgrounds for the courses they take, but there are exceptions. The significant reported cases deal with students who do not have programming or engineering backgrounds. The most common remarks indicate a lack of programming experience, followed by a lack of knowledge of an assembly language. Other remarks indicate a need to enforce the self-study and particularly the CSF sequence before proceeding to core courses. The typical nonprogrammer student may be the developer of technical publications whose interest is in learning basic concepts and computer science terminology. This type of student does not expect to write programs.

Once a student completes the CSF series (which consists of a mathematics self-study course, CSF 1, a Pascal self-study course, CSF 2 and CSF 3), he/she receives information concerning follow-on courses. This information is given with the understanding that the student is responsible for obtaining management approval to attend selected courses or to work toward bachelor-level equivalence and beyond.

SELECTING INSTRUCTORS

Classes in the ULCS Program are taught by teams of two or three instructors. Each class has one lead instructor or two co-lead instructors, possibly with one or two assistants. The lead instructor or co-lead instructors primarily is/are responsible for presenting the course material. Assistant instructors may teach one or two lectures and conduct help sessions.

Instructors are selected from university faculty who teach or have taught in the subject area. For the first few classes, an instructor is hired to assist a course leader. Class managers and students evaluate the faculty after each course. Performance determines an assistant's subsequent opportunities as a leader or co-leader. An instructor is invited to teach at least twice. Stronger performers eventually become co-lead or lead instructors.

In preparing to teach, the instructor is encouraged to follow the course outline, but may make the course his/her "own" by personalizing handouts and transparencies.

SELECTING A LOCATION

A conference facility used for a class ideally should be 30 to 40 miles from the site of the employees enrolled in that class. A second, though inferior choice, is the IBM site where the students are employed.

The primary objective in choosing a location is to remove students from home, office, manager and telephone, and to facilitate the task of studying intensely. Participants experience significantly increased learning when they are off-site.

EXECUTION

THE WEEKLY PLAN

Before the start of the course week, students are expected to prepare by reviewing the class materials and working self-study problems. Generally, students arrive at the conference center on Sunday evening beginning the week of the class. At that time there is an orientation session and an occasion to relax before beginning the intensive week. Pre-tests are sometimes given at this point in order to measure what has been learned beforehand.

Class sessions are taught by university faculty with credentials in computer science and/or mathematics. The goal for the week is to cover the equivalent of material covered in a three semester hour university course. At the end of the week, students complete either an in-class or take-home final examination which is graded and returned. Individual scores are revealed by the class manager and faculty only to the student. Individual grades are not kept beyond the class period, nor are they reported to the student's manager. Attendance and participation is required for satisfactory completion of a course. Completion of a course is recorded permanently in an employee's personnel file.

THE DAILY PLAN

Students attend two sessions each day, morning and afternoon. During study time (often late into the evening) they complete assignments on the day's topics. The instructors are available for personal help during study time. A typical day consists of at least four hours of instruction, followed by homework problems that are to be worked during study time. The environment during study time is that of a laboratory, where the instructors work one-on-one or with small groups of students.

Students are encouraged to exercise between the afternoon class and dinner. This relieves pent-up stress and freshens the mind during a full day of study. The evening sessions are viewed positively, with the participants feeling that they can obtain extra, needed help.

Typical Daily Schedule	
7:00 am	Breakfast
8:00 am	Homework Review
9:00 am	Lecture
10:00 am	Lecture
11:00 am	Study
12:00 pm	Lunch
1:00 pm	Homework Review
2:00 pm	Lecture
3:00 pm	Lecture
4:00 pm	Study
5:00 pm	Exercise
6:00 pm	Dinner
7:00 pm	Study
8:00 pm	Study
9:00 pm	Study
10:00 pm	Study
11:00 pm	Retire

Figure 2

EVALUATION

Evaluation is an important step in the iterative process of "organization, execution, evaluation." Through periodic evaluation, the Program is kept

effective. An evaluation is performed after every class. Periodic surveys of the students and instructors are conducted. The faculty advisory committee reviews the curriculum and courses. From this review, courses are maintained, added, deleted and updated.

FOLLOW-THROUGH

After a student has completed the CSF series, a certificate is presented. Another certificate is presented upon completion of the six core classes. While a student cannot earn university credit through this Program, each completed course is important within the company. Completion of a course is viewed as an enhancement of an employee's general professional credentials.

The individual retains what he/she has learned in a course via private study, use of his/her newly acquired knowledge on the job and subsequent courses. Reports of a product being revised or improved after completion of a course are common.

OBSERVATIONS

STUDENT REACTIONS

ULCS Program participants evaluate the Program to be between "good" and "excellent." Instructor skill and performance rank high. The texts and materials are considered good, exposing the students to views outside IBM (one of the stated objectives of the Program). Taking courses for no credit is not a concern for most students. The typical student feels that raising his/her level of knowledge within five years to that of a computer science bachelor's degree recipient is sufficient in and of itself.

The majority of those who complete one or more of the ULCS courses state that they feel "more motivated," "more informed," "more quality conscious" and "more productive." Students come away from the courses with fresh ideas. Some of the topics covered, such as software design, testing and verification, are directly applicable to everyday work, in addition to being professionally educational.

INSTRUCTOR REACTIONS

The question "Is the ULCS experience equivalent to a typical university experience?" touches the sensitivities of several instructors. None consider it to be equivalent, except possibly in terms of "coverage" of subject areas. It is considered to provide an excellent overview of the field, but in general lacks depth because of time and laboratory constraints.

Faculty who teach in the Program feel that it is a better experience for them than attending conferences. They observe other professors teaching, see other course material and subsequently incorporate some of it in their own university courses. An instructor can teach a ULCS course several times a year if he/she is the developer. Lots of exercise and refinement can take place. An instructor associates with other teaching professionals and with a different kind of student. The development of the courses also differs from development in a university environment: during development, courses are walked through in detail with other faculty experienced in the subject matter.

REACTIONS OF MANAGERS

The reactions of managers are positive. They view the Program as valuable. Many managers take the courses themselves and return feeling they understand computer science better and communicate better with CS graduates. It also gives them the vocabulary and understanding needed to read the professional literature.

It is difficult to point to specific skills and concepts managers actually return to use after a class, but the consensus is that their ability to do the total job has been enhanced.

Losing an employee at full salary with all expenses paid for the one week courses has caused no reported hardships. The benefits of having an employee attend courses and return with new sets of tools greatly outweighs the expenses and temporary absences.

COMPANY REACTION

The ULCS Program reflects the company's commitment to employee education. This educational offering reflects the longstanding practice of full employment at IBM. The need for more employees to have current computer science knowledge is increasing. The only way to meet this need and maintain full employment is to upgrade the educational base of programmer employees. This is of particular importance during periods of reduced outside hiring.

ANALYSIS

The objective of involving employees who have been away from formal education five or more years is met. The courses are not as relevant to recent computer science graduates. Nevertheless, most Program participants are involved in some aspect of programming.

Instructors agree almost universally that the Program participants are highly motivated. Some exceptions have been reported. Nevertheless, an atmosphere of "professional pride" uncommon in the university environment is evident.

Test results vary for each course offering and group of students, but the majority of students do well on the examinations, scoring in the upper 80% range. After class surveys indicate that most students feel they have learned the material.

SUMMARY

The ULCS Program provides a meaningful educational opportunity in computer science for IBM employees. The ULCS Program is specific in its content and direction, while university CS programs provide a broader science background in addition to the material in computer science. Due to the intense nature of the ULCS courses, students learn in less time. However, the students have less time to digest the course material and to do homework than students taking regular university CS courses. Furthermore, some lengthy homework problems or projects often assigned in regular university CS courses are not suitable for ULCS courses because of time constraints. Consequently, topics are treated in less depth in ULCS courses than in regular university CS courses.

To stay current with the rapidly evolving discipline of computer science, students take new or revised ULCS courses. A university bachelor's degree program is a one-time experience, designed to prepare a student to enter the work force. The ULCS Program is an ongoing experience, designed to develop and maintain employee skills.

The goals of the Program are being met. In particular, students are expanding their professional horizons, IBM is maintaining a contemporary programming community, and traditional ties between IBM and universities are being strengthened.

FUTURE OF THE PROGRAM

The ULCS Program has been received well by management at IBM; it is in the process of becoming a corporate wide program. It will continue for the foreseeable future, though not in its present form, as it will be updated constantly by modifying, adding and deleting courses. Whatever is needed will be done to keep the Program on the leading edge of computer science.

A FIRST COURSE FOR RETRAINING CORPORATE EMPLOYEES FOR CAREERS IN COMPUTER PROGRAMMING

Kevin T. Cragun
Robert P. Burton
Brigham Young University
Provo, Utah 84602

and

Val S. Judd
IBM Corporation
Austin, Texas 78758

Abstract

International Business Machines (IBM) Corporation has designed, developed and implemented a first course for retraining its employees for careers in computer programming. Programming Fundamentals is a fifteen week course. Course objectives were identified by university computer science professors and IBM personnel. The course units were developed by university professors and are taught by IBM technical education departments. The Programming Fundamentals course fulfills the objectives of CS1 and part of CS2 as described by the Association for Computing Machinery.

Once candidates for retraining as programmers have satisfied selection criteria, they must complete two math self-study courses which are prerequisite to the Programming Fundamentals course. A large percentage of the participants complete the course successfully. Most course graduates are placed in programming positions.

COURSE HISTORY AND DEVELOPMENT

One of the basic principles of IBM is respect for the individual. IBM traditionally has provided opportunities for its employees to better themselves as opportunities and business needs allow. In addition, IBM equips employees with new skills as the need for one job skill diminishes while the demand for another job skill increases. This retraining allows IBM to provide new career opportunities for qualified employees and to balance the employee skill mix. Programming Fundamentals was developed as a course that can be used to retrain qualified employees as programmers when the programming skill is in high demand.

COURSE ROOTS

In the 1960's, IBM established a programmer training course to prepare employees to use the widely-marketed IBM S/360 computer. The course, known as Basic Programmer Training (BPT) or Basic Programmer Education (BPE), provided programming instruction for employees with college degrees that was, at the time, not available in most universities. The BPT course was structured around the IBM S/360's architecture and included instruction in IBM S/360 assembly language as well as other high level languages.

Programming emerged as a recognized profession in the 1970s; several universities began to offer computer science curricula. At that time, IBM hired computer science graduates. BPT was used as a course to allow qualified employees to change careers. Also during the early 70s, BPT was used to retrain employees for programming positions to balance skills within the company. During the 1970s as IBM introduced into the marketplace computers with different architectures, the BPT course was modified to train those employees who would be doing programming for an architecture that was different from the IBM S/360 architecture. When these modifications occurred, attempts were made to include new programming techniques.

By the early 80s, the Basic Programmer Training course was obsolete for organizations in IBM that had to develop software for multiple architectures. The course only superficially covered critical computer science topics such as step-wise refinement and structured programming. The course duration was eight weeks and did not cover satisfactorily such critical topics as documentation practices, testing or maintenance. In addition, it did not utilize effectively English-like design languages.

NEW COURSE OBJECTIVES

By the middle of 1983 an internal technical education department was responsible for programming technical education for several IBM divisions that developed diverse architectures. The management of this technical education department made a decision to develop an entirely new programmer training curriculum. To lay the groundwork for course development, a panel was formed. It was composed of six university computer science professors, five of whom were instructors for IBM in an internal university level computer science program, and seven IBM technical employees. Each IBM employee represented a different programming area such as systems engineering, information systems, programming assurance, application programming and system development programming. The panel

was set in place to accomplish the following objectives:

a) to define a course that would prepare employees for programming careers in the shortest time possible

b) to provide employees with the skills to communicate with other employees who had earned computer science degrees

c) to provide employees the background to continue with environmental training courses (courses that provide training for an employee's current job assignment)

d) to provide employees with the background to continue taking computer science courses either within IBM or at a local university.

Using a panel composed of both university professors and IBM programmers, a course could be developed that would blend leading edge programming concepts being taught and researched by the universities with the IBM methods of programming, including testing strategies, documentation standards etc.

During an intensive three day workshop in December, 1983, the panel accomplished four basic steps in the course development cycle:

1) need analysis - determining the need for the course

2) task analysis - selecting the material to be covered in th2 course

3) module objectives for each task, and
4) a flow of the modules into a course that met the course objectives.

NEED ANALYSIS

The need analysis requirement was satisfied since the Programming Fundamentals course would be replacing the BPT course for which the need was established.

TASK ANALYSIS

The task analysis was completed during the workshop. This was accomplished by allowing the IBM employees and the university professors to interact using the brain storming method of problem solving. Once the tasks that were to be learned were defined and agreed to, each task was assigned a level of learning -- again from interaction between the two groups.

OBJECTIVES

With the tasks defined and a level of learning assigned to each task, the panel was at a point where behavioral objectives could be created for each task. Again this was accomplished by interaction between the two groups.

COURSE FLOW

Once the task analysis was completed and the objectives had been defined, the panel grouped together tasks that were logically related. The related tasks formed modules for the course. Objectives were defined for each module in the course. The modules then were organized into a course flow that was acceptable and agreeable to both IBM and the university professors.

At this point the panel was disbanded and the next phase of development began.

MANAGEMENT REVIEWS

In order to insure that the level of training was acceptable to those who potentially would supervise the graduates, visits were made to several IBM programming laboratories. During these visits the objectives of the course, the content of the course and the flow of the course was presented to management. Any changes were discussed, understood and agreed to by both the course developers and the management reviewer.

COURSE DEVELOPMENT

Once all of the management reviews had been completed, four university computer science professors were selected by the IBM technical education department to develop the course; by choice, all four professors had taught in the internal university level computer science courses. Therefore they would know how to interface the Programming Fundamentals course with this existing curriculum. The identified teaching units were divided among the four developers.

During the subsequent nine months, from May 1984 to February 1985, the course was designed and developed in detail. The course developers met regularly to review progress and to maintain consistency. In the middle of the course and again at the end of the course the IBM technical employees were assembled and assigned the task of reviewing with the course developers the content of the course. All course materials were completed during the first three months of 1985. By March, 1985, the course was ready for a pilot class.

COURSE IMPLEMENTATION

PILOT TESTING

The first 15-week pilot class was taught from March through July, 1985. It was taught by IBM educators and the four course developers. Fifteen employees participated together with four prospective instructors employed by the technical education department. The IBM instructors attended the class to become familiar with the course and to prepare to teach the course in the second pilot class offered from July through November, 1985.

Simultaneous with the pilot class, two university co-op employees reviewed, upgraded, and rewrote portions of the course. One of these employees had experience instructing at the secondary and college level and was therefore able to produce the materials at the correct level for the employees who would ultimately take the course. The second was an advanced computer science undergraduate student. This team created and refined daily course objectives, daily outlines with time limits, notes for the instructors, foils, programming assignments, exams and references to texts. This activity was carried out real-time while the first pilot was being taught.

The second pilot was taught using the revised materials. Another rework of the course was completed to insure that all rough edges had been removed.

CANDIDATE SELECTION

Although each IBM site is autonomous, the means by which an employee is made aware of the Programming Fundamentals course is consistent. The employee either hears about the course when a site makes a general announcement or when his manager notifies him individually. At the initiation of either the manager or the employee, the manager may nominate an employee to take the course. Since an unprepared individual may apply to take the course or a manager may encourage an employee who does not have a programming aptitude, the candidates typically are screened as described below.

Prospective candidates are given an audiovisual presentation to acquaint them with the responsibilities and lifestyle of a computer programmer. Those who retain interest in a programming career file a more formal application with their managers. The personnel department then administers an aptitude test to the applicants. A selection committee reviews the applicants' personnel records and aptitude tests. Programming development managers then interview the applicants. Based on all of the data collected, a final selection of candidates is made.

AUDIOVISUAL PRESENTATION

The audiovisual presentation helps prospective candidates reach an informed decision about a computer programming career. The presentation was created after interviewing more than one hundred programmers who indicated what applicants should know about a programming career. Various responsibilities of programmers as well as the rewards and frustrations that accompany a programming career are discussed in the presentation.

APTITUDE TEST

The aptitude test was developed by IBM and is proprietary.

SELECTION INTERVIEW

The selection interview identifies applicants who have skills in problem solving. It also determines if an applicant has a nonacademic programming background or educational background sufficient to be successful in the course.

CANDIDATE PROFILES

Candidates selected for the Programming Fundamentals course predominantly are nontechnical employees. They often hold bachelors' or associate degrees from universities or have been exposed to programming. Representative participants in the pilot classes included a programming librarian, a secretary and a general machinist.

COURSE DESCRIPTION

PREREQUISITIES

Prior to beginning the Programming Fundamentals course, a candidate must complete self-study courses. This preparation ensures that all participants enter the Programming Fundamentals course with enough background to perform satisfactorily. Two self-study courses are mandatory and graded. They include an algebra refresher and an introduction to logical expressions.

COMPARISON WITH ACM RECOMMENDED CURRICULUM

The Programming Fundamentals curriculum is designed to include all of the recommended material from CS1 [1,2] and some of the material from CS2 [1,3]. CS1 includes top-down design, step-wise refinement, control structures, procedures, functions, parameter passing, internal data representation and recursion. Topics from CS2 include file handling, pointers, linked lists, sorting and searching. The course also provides information on and exercises with scalar and structured data types, testing strategies, the project life cycle, and computer organizations and architectures. It is sensitive to maintenance and IBM's documentation standards. The course provides team projects to acquaint the participants with working in groups. The team projects include design inspections and code reviews.

Rather than learn problem solving through either abstract instruction or practice, the course combines these two methods of learning. Classroom instruction satisfies the requirements for abstract learning. Pascal, a high-level, block-structured language is used for programming assignments to encourage the participants to learn good programming style.

COURSE FORMAT

The course is held at an IBM site where each participant is provided with an IBM PC. Each week of class consists of fifteen to twenty hours of lecture and twenty to twenty-five hours of exercises and programming. Tutoring is provided by instructors for those individuals who need additional assistance with concepts or programming assignments. The participants are evaluated through the administration of weekly tests; each participant must maintain an average of at least 70% and must never score below 60% on any test.

COURSE ACCEPTANCE

The course has been accepted widely within IBM. Currently it is currently being taught at eight of the domestic research and development sites across the United States. In addition, it is being used by R&D laboratories in Canada and England. It became popular so fast that the developing organization abandoned the idea of teaching it and became the support organization for those who are teaching it. This support includes train-the-trainer classes, keeping the materials updated, providing technical assistance and direction, assisting locations with problems, etc.

COURSE INTEGRATION

The Programming Fundamentals course is designed to serve as an introductory course for those who elect to continue on to IBM's university level computer science curriculum. However, an IBM employee generally is expected to work for eighteen to twenty-four months after graduating from the Programming Fundamentals course before enrolling in the university level computer science curriculum. An employee also must take an additional mathematics self-study course that acts as a bridge between the Programming Fundamentals course and the university level courses. After three to six years and ten courses in the university level computer science

curriculum, the employee is considered to have technical parity to a computer science bachelor's degree recipient (see figure 1).

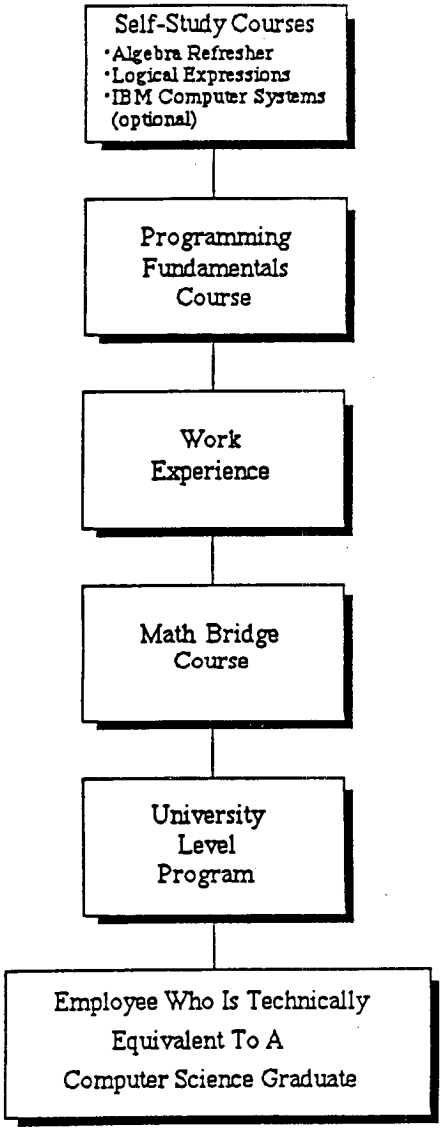


Figure 1

GRADUATE EVALUATIONS

Evaluations of the performance of Programming Fundamentals course graduates yield positive results. Of eighteen graduates evaluated (in mid-1986), sixteen use their new skills in present assignments (see figure 2). Ten of the eighteen participants have enrolled in a self-study course to learn an additional programming language; the Programming Fundamentals course has provided a background sufficient for the graduates to learn new programming languages without formal classwork. Five have registered for formal classes as well as self-study programs and two have enrolled in formal classes (see figure 3). Eight of ten managers interviewed indicate that they prefer Programming Fundamental graduates to BPT graduates. Nine of ten managers indicate they would hire another graduate if given the opportunity.

These evaluation numbers are small because they represent only the two pilot classes taught in 1985. A new study is now under way from a much larger sample. Data is not yet available.

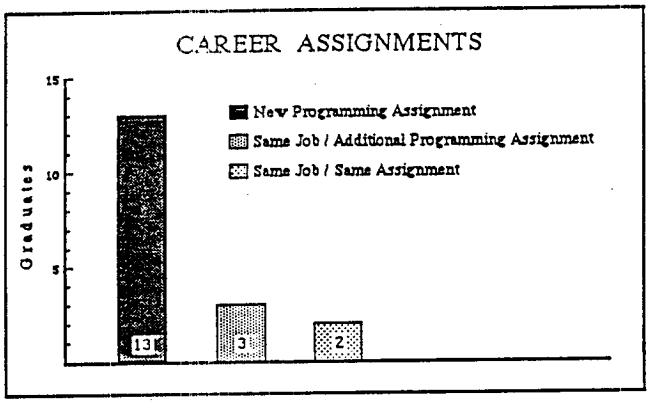


Figure 2

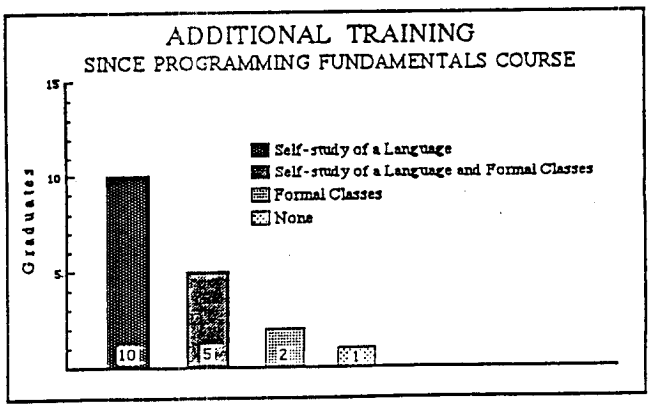


Figure 3

CONCLUSION

Fluctuations in the economy, IBM's desire to allow employees to acquire needed skills, and the Programming Fundamentals curriculum have offered many IBM employees the opportunity to change careers paths. The course provides employees with the skills to begin a programming career and the background to continue their professional development. This course for retraining corporate employees for careers in computer programming has demonstrated both intrinsic and measurable economic value to the corporation.

REFERENCES

[1] ACM Curriculum Committee on Computer Science, Curriculum '78: Recommendations for the Undergraduate Program in Computer Science, CACM 22, 3 (March 1978), pp. 147-166.

[2] ACM Curriculum Committee on Computer Science, Recommended Curriculum for CS1, 1984, CACM 27, 10 (October 1984), pp.998-1001.

[3] ACM Curriculum Committee on Computer Science, Recommended Curriculum for CS2, 1984, CACM 28, 8 (August 1985), pp. 815-818.

TEACHING PROTOTYPING: USING A LIMITED APPROACH FOCUSING ON USER INTERFACE DESIGN TO GET THE BANG WITHOUT THE BUCK

Neil W. Jacobs and Gregory L. Neal
Northern Arizona University

ABSTRACT

This paper addresses an alternative approach to introducing prototyping into the curriculum while avoiding major initial commitments or expenditures. The approach involves a limited exposure to prototyping which focuses on the user interface (UI), i.e., input and output displays. It is suited for teaching the fundamentals of user interface prototyping for systems using screen based input and output. The required investment in software and training is much less than full function prototyping. This approach may be desirable for schools seeking to introduce prototyping with limited resources or those wishing to postpone the decision on a 4GL until a language standard emerges.

INTRODUCTION

Software developments in the area of fourth generation languages (4GLs) coupled with increased user information literacy has placed new emphasis on prototyping, i.e., building working models, as a means to conquer persistent problems with the conventional system development process. Prototyping with a 4GL can increase programmer productivity (6), help reduce the backlog of user needs, cause users to be more active participants in the development process (10), provide an effective alternative approach to designing systems--especially the user interface portion of the system (14), or allow users to develop more of their own systems.

The key aspects of prototyping with 4GLs that leads to these benefits are rapid delivery and user involvement. Use of 4GLs allows rapid delivery (and quick revision) of a functioning system (3, 7, 9, 13). This translates into greater user involvement in the system development process (15) as a result of the timely feedback and the build it, try it, fix it mode of prototyping.

Prototyping helps users through the abstract process of application development by giving them a clear, hands-on understanding of the system design through interaction with the system's inputs, outputs, and operation (4). This leads to active participation in the design process and helps users develop confidence in and a sense of ownership of the system design. In part, these benefits are derived from exposure to tangible input and output displays (3).

This improvement in the quality of designer-user communication regarding system design is a major advantage of the prototype approach which has implications not only for system development in the real world but also for the CIS curriculum. CIS professionals and educators are calling for the inclusion

of fourth generation languages and prototyping methodologies in the curriculum (11). As Blaisdell (1) notes, this presents issues regarding: curriculum placement, which 4GL to use, and whether to direct the course toward CIS professionals, end users, or somewhere in between.

Significant front end costs are involved in setting up to teach prototyping using a 4GL (1). A 4GL must be selected from the many choices available--a significant decision considering the lack of a de facto standard and the initial investment required. This investment includes the 4GL software, faculty training, and the curriculum or course revisions required to teach the 4GL so that students are able to prototype with the selected 4GL.

Some situations may call for a different approach which provides students exposure to prototyping without the 4GL commitment and associated investment. A limited application of prototyping which only addresses the user interface is the alternative approach presented in this paper.

Teaching students how to design the user interface, and giving them the opportunity for interactive sessions with users on designs the students prepare, would provide them exposure to a crucial aspect of prototype design. If, as Stahl (15) claims, the key to successful prototyping is to produce an efficient system which will be easy for users to learn and use, then user interface prototyping is an appropriate place to begin a less comprehensive approach to prototyping. Because of the extensive user interaction prototyping entails, information systems professionals have greater need for interpersonal skills as Naumann and Jenkins have noted. Further, as Konsynski (8) states, 4GLs change the emphasis from the process of creating computer solutions to a focus on the business problem faced by the user. Therefore, by getting students actively involved in designing the user interface and interacting with users, prototyping and the necessary concomitant interpersonal skills and problem orientation can be concurrently developed.

The remainder of this paper provides an explanation of a limited approach to prototyping focusing on user interface prototyping and explores how the limited approach might be implemented.

A LIMITED APPROACH: USER INTERFACE PROTOTYPING

User interface prototyping is the building of a working model of the system that the user will see and use. Included are the design of input screens, data input edits, output screens, and system-user interactions that occur as the system is used. Examples of the system-user interactions are menus, functions executable from any screen, help mechanisms, and the paths the user would follow to move from one screen to another.

User interface prototyping is a portion of full function prototyping. UI prototyping encompasses all the user interactions with the system except for the actual functioning of the system. Screen use, data entry, transfer between screens in the performance of the user's job are all simulated; however, the program logic for computation, data storage, or data retrieval are not.

The design issues which are specifically addressed in user interface prototyping can be divided into four categories, general system design issues, user concerns, screen input/output issues, and help system issues.

GENERAL SYSTEM DESIGN ISSUES

General design issues address data requirements, error handling, and information output.

(1) Information Outputs - What information is needed for the user to perform their job?

(2) Data Requirements - What data must be collected to develop the information?

(3) Error Handling - What should the system consider to be an error?

USER CONCERNS

User concerns include flexibility, user sophistication, job knowledge, and sequencing and controlling screen use.

(1) Flexibility - The interface must support different levels of user's computer expertise and keyboard proficiency. Frequency of use is also important since extensive help provisions such as long sequences of descriptive menus may be appropriate for the infrequent user but laborious and unnecessary to the frequent user.

(2) User Sophistication - User sophistication may be defined as the ability for the user to handle complex decision situations. The interface designed for engineers could be considerably more complex than that for a retail clerk entering invoice information.

(3) Job Knowledge - Job knowledge is important since the user may not understand the purpose of the system and how it improves job functionality. If the user has little knowledge of how the system enhances the job function the interface must carefully guide user inputs by detailed prompts.

(4) Screen Sequencing and Control - The interface should be designed so that screen sequencing follows the user's problem approach and not the program's processing logic.

SCREEN I/O

Issues regarding screen input and output include display attributes, data entry techniques, and formatting.

(1) Display Attributes - The use of color, reverse video, and highlighting has great impact on ease of system use. Attributes should be selected which allow the user to easily visualize the next action or required response.

(2) Data Entry - With interactive interfaces the developer can enhance the ease of entry by providing pop-up or insert windows with relevant information for data input. If errors are made during data entry then messages showing the error can be displayed accompanied by screen descriptions of possible responses.

(3) Format - What should be on the screen and where. The format and location of data entry plays an important part in user understanding.

HELP SYSTEM

Help system related issues address on screen program control aids, what on screen help to have available, and the need to have help tailored to the context of the task being performed.

(1) On Screen Program Control Aids - Diagrams and menu specifications should be built to enable the user to easily follow the operation

of the system. With messages or other indicators like the display of commands executed by function keys, the interface can help the user understand how the system operates.

(2) Availability - The designer must deal with the question of when does the user need additional help and in what form should the help be provided.

(3) Context Sensitivity - If help is provided in direct relation to the current task being performed by the user, it has the greatest benefit. The interface should be constructed to allow the right help at the right time.

HOW USER INTERFACE PROTOTYPING COULD BE TAUGHT

Some prerequisite knowledge should be acquired before attempting user interface prototyping. This includes knowledge of a number of systems analysis concepts, i.e., user factors influencing design, principles of screen generation and data editing and error handling, and techniques for the design of the paths users will be provided between screens. Software prerequisites include a tool program, such as Dan Bricklin's Demo (2), to facilitate the preparation of specific user interfaces for input, output, and operation of the system. Related knowledge requirements include an understanding of business functions and related problems, as well as fundamentals of interpersonal relations. An upper division CIS major who had completed courses in organizational behavior, management information systems, and systems analysis would have the basic prerequisites.

Since user interface prototyping involves skill learning, i.e., the implementation of previously learned concepts and practice in their use, an experiential approach is most appropriate. We recommend use of a microcomputer laboratory workshop in conjunction with a structured experience which encompasses gaining an understanding of a user's needs, developing a prototype of the user interface, interaction with the user on the interface design, and subsequent revision of the prototype to the user's satisfaction. An annotated outline of this approach is shown in Appendix A.

With such an approach students can be introduced to one of the crucial aspects of prototyping and given the opportunity to develop important prototyping skills with much less institutional investment than would be required for full function prototyping using a 4GL. This could be used as an interim approach for institutions not yet ready to commit to a 4GL, or as a less comprehensive introduction to prototyping than use of a 4GL would entail.

REFERENCES

1. Blaisdell, James H. "Teaching fourth generation languages," Proceedings, Fifty Annual Information Systems Education Conference, Atlanta, Georgia, October, 1986, pp. 297-301.

2. Bricklin, Dan. Demo, Software Garden, Inc., 1986.

3. Campbell, Donald F. "Reducing the applications backlog with 4GLs," Journal of Information Systems Management, Fall 1985, pp. 8-13.

4. Connell, John and Linda Brice, "Rapid prototyping," Datamation, August 15, 1984, pp. 93-100.

5. Freedman, David H. "Programming without tears," High Technology, April 1986.

6. Green, J. "Productivity in the fourth generation: Six case studies," Journal of Management Information Systems, 1(3), Winter 1984-85, pp. 49-63.

7. Klingler, Daniel E., "Rapid prototyping revisited," Datamation, October 15, 1986, pp. 131-132.

8. Konsynski, B.R., "Advances in Information System Design," Journal of Management Information Systems, 1, no. 3 (Winter 1984-85), pp. 5-32.

9. Martin, James, Fourth-Generation languages, Volume I, Prentice-Hall, Englewood Cliffs, N.J., 1985.

10. Naumann, J. D., & Jenkins, A. M. "Prototyping: A methodology for the design and development of application systems," MIS Quarterly, September 1982, pp. 29-44.

11. Prickett, Jan. "Fourth-Generation languages and the 'CIS' curriculum: The analysis, design, and development of computer information systems," Proceedings, Fifty Annual Information Systems Education Conference, Atlanta, Georgia, October, 1986, pp. 23-25.

12. CIS '86: The DPMA model curriculum for undergraduate computer information systems, Data Processing Management Association, Park Ridge, Illinois, 1985.

13. Schussel, George, "Fourth generation productivity tools--A shopping guide for software consumers," Journal of Data Management, October, 1984, pp. 42-46.

14. Sprague, Ralph & McNurlin, Barbara, Information Systems Management in Practice, Prentice-Hall, Englewood Cliffs, N.J., 1986

15. Stahl, Bob, "The trouble with application generators," Datamation, April 1, 1986, pp. 93-94.

Appendix A
Outline of Approach to Teaching User Interface Prototyping

Step	Resources Required	Comments
Concept review	Instructor Materials on user interface design Structured experience	Brief review of prerequisite knowledge and introduction to structured experience
Screen building workshop	Instructor User interface design software Microcomputer lab	Introduce software, walk students through sample interface design, and have them design at at least one interface for the structured experience
Self-directed student lab work	Microcomputers User interface design software	Students complete design of user interface screens on their own
Advanced interface design concepts workshop	Instructor Microcomputer Lab	Introduce UI design software use for data editing, error handling, and movement between screens
Self-directed student completion of prototype	Microcomputer User interface design software	Students enhance interface design using advanced concepts to produce completed prototype
User interaction	Microcomputer Role players (users)	Designer students interact with individuals playing the role of users to identify revisions required
Revision and recycle user interaction	Microcomputer Role players (users)	Students revise interface design, and interact with user role players until design is accepted

A COMPARATIVE ANALYSIS OF THE MAINTAINABILITY OF CODE PRODUCED BY APPLICATION GENERATORS

Charles W. Golden, DBA
Auburn University at Montgomery

Robert C. Lake, DBA
Auburn University at Montgomery

This paper describes an analysis of source code produced by two popular application generators which use the dBASE programming language. Certain software characteristics are measured and evaluated to determine the maintainability of the generated code.

INTRODUCTION

The term "application generator" (AG) refers to software which can transform a set of problem specifications into program(s) that can be used to satisfy information needs. The resulting program(s) can be directly executable code, or source code in a procedural language such as COBOL, Assembler, or dBASE that must be compiled or interpreted for use. Such generators have been used for a number of years in the mainframe and minicomputer environments, but have also recently become available to microcomputer users. Our focus here is on the type of AG that produces source code, for our interest is in examining the *maintainability* of the generated code.

The earliest generators produced COBOL source code, but lately generators have been introduced to be used in conjunction with database management systems such as dBASE III. For various reasons, these new software tools are appealing to two distinct classes of microcomputer users. The first class contains professional application developers. For this group, the combination of a relational database management system (such as dBASE or PARADOX) and a powerful application generator approximates the capabilities of a fourth generation language (4GL) allowing them to use the now popular and efficient "prototyping" methodologies in their development activities.

The second class of AG users consists of end users that use the generators to develop application systems for themselves or close colleagues. They rely on the AG and DBMS to do the "heavy" coding needed for the basic, but complex, input/output and file management operations. It is the end user group that is of primary interest here (but many of our findings also have implications for professional developers).

Most of the microcomputer based application generators now on the market are designed to produce only the code necessary to implement the standard file maintenance and reporting activities common to most data processing systems. Unless the generator is exceptionally sophisticated, then, an application requiring complex logic must have at least part of the code "custom-made" and inserted by hand. It is obvious that if the code produced by the generator is not easily comprehendible, the end user programmer -- and, probably, many professionals -- will have difficulty making any necessary alterations.

The purpose of this paper is to first discuss the characteristics that make program code complex and difficult to

maintain, as well as to present some metrics which can be used to specify the degree of complexity. Finally, we will make a comparison of the code produced by two popular application generators using these complexity metrics.

PROPERTIES OF MAINTAINABLE CODE

There is a considerable body of research dealing with measuring the characteristics of programs -- "software metrics" -- and how these measurements describe the understandability, and therefore the maintainability, of programs.

Perhaps the most obvious property of generated code that is detrimental to maintainability is the *complexity* of the code. "...Complexity can be gauged in three general ways: the longer a program or module is, the more complex it is; the more decisions a program or module contains, the more complex it is; and the deeper that logic is nested in a program or module, the more complex it is."[1] This leads us to the three corollary rules discussed below.

LESS CODE IS BETTER

A code module several pages long is, first of all, psychologically intimidating. How many lines does it take to introduce significant complexity? Weinberg's studies[2] indicate that a module with over thirty lines is difficult to understand. IBM[3] recommends 50 lines as a workable number. We are not so willing to establish a rigid limit, preferring rather to use the heuristic that less is better. We do agree, though, that if a module can be limited to a single page the chore of tracing through the logic is less formidable than if several pages are involved. This would mean a limit of 66 lines based on a page size of 8 1/2 X 11 inches.

FEWER DECISIONS ARE BETTER

McCabe[4] proposes a complexity measure based on graph theory which he labels the *cyclomatic number*. Essentially, this is the number of paths through a module, and can be determined by counting the number of compares plus one. According to McCabe, the cyclomatic number should not exceed 10.

For our purposes, we have simply counted the number of IF and CASE statements (since the case statement is a substitute for the IF statement). We have not adhered to a McCabe's critical number of 10, preferring, again, merely to declare a module with more decisions as less desirable than one with fewer decisions.

LESS NESTING IS BETTER

Martin and McClure[5] indicate that the psychological complexity of a program increases as the nesting of IF statements increases. We have extended this a step further and recorded the combined nesting of IF and DO WHILE statements. As in the case of the earlier metrics, we prefer to say that a module with less nesting is more desirable than one with more.

EVALUATION OF APPLICATION

Similar applications were generated with two popular AGs -- *Genifer* (version 1.01) from Bytel, and *Quick Code Plus* (version 1.0) from Fox and Geller. Both are similarly advertised as meeting the needs of end users as well as professional developers.

APPLICATION DESCRIPTION

In order to make a proper comparison of the two generators, they must be evaluated across the same application. This section contains a description of the application that was used, a budget control system (BCS). The specific information required for the budget control system is as follows:

Account Number	Check Number
Account Description	Amount Paid
Budgeted Amount	Expense Description

Date Paid Payee Tax

In order to reduce the disk space required, the file was put into third normal form. The data structures, key fields, and linking field are shown below.

STRUCTURE	KEY	LINK
BCS_BUDGT		
ACCT_NUM	ACCT_NUM	
ACCT_DESC		
BUDGET_AMT		
BCS_EXP		
DATE_PAID	DATE_PAID	
CHECK_NUM		
PAYEE		
ITEM_DESC		
EXP_AMOUN		
TAX		
ACCT_NUM		ACCT_NUM

The basic logic required to make the application work is not complex. On the other hand, it is not trivial in that it requires the linking of two files. This exercise deals only with the file maintenance routines and ignores the report part of the application.

EVALUATION OF RESULTS

Quick Code Plus and *Genifer* were both able to generate 100% of the code required for the application and no insertion of code by hand was required. *Quick Code Plus* built 51 modules as compared to only 26 for *Genifer*. A comparison of lines of code is shown in Table 1.

It can be seen from Table 1 that *Quick Code Plus* generated significantly larger modules. If we view as undesirable those modules with more than 66 lines or one page of code, then *Quick Code* produced 16 modules (or 32% of the total modules it generated) that would be considered too long. *Genifer*, on the other hand, produced only 1 module (4%) that exceeded the one page limit.

Table 1
Lines of Generated Code

Number of Lines	*QC+* # Modules	%	*Genifer* # Modules	%
5- 24	18	35	17	46
25- 44	9	18	4	15
45- 64	8	16	3	12
65- 84	1	2	1	4
85-104	4	8	0	0
105-124	2	4	0	0
125-144	2	4	0	0
144-164	1	2	0	0
145-164	1	2	0	0
165-184	2	4	0	0
185 +	4	8	1	4
	51	100	26	100

Worst Case:
Quick Code - 438 (2)
Genifer - 212 (1)

Table 2 compares the number of (IF/CASE) paths included in the modules produced by each generator. Again, *Genifer* produced less complex code (measured in terms of the number of paths per module).

Table 2
Number of Paths

Number of Paths	*QC+* # Modules	%	*Genifer* # Modules	%
0- 5	32	63	22	85
6-10	6	12	2	8
11-15	3	6	0	0
16-20	5	10	2	8
21-25	2	4	0	0
26-30	0	0	0	0
31-35	0	0	0	0
36-40	1	2	0	0
41-45	2	4	0	0
	51	100	26	100

Worst Case:
Quick Code - 41
Genifer - 20

The final metric that was calculated was nesting level. As shown in Table 3, the maximum nesting level for *Quick Code* modules was eight, and for *Genifer*, three. This does not necessarily imply that Quick Code produces modules with excessive complexity. However, if "less nesting is better," then clearly *Genifer* produced modules with a more acceptable complexity level.

Table 3
Nesting Level

Maximum Nesting Level	*QC+* # Modules	%	*Genifer* # Modules	%
0	13	25	4	15
1	14	27	11	42
2	11	22	6	23
3	6	12	4	15
4	1	2	1	4
5	4	8	0	0
6	0	0	0	0
7	0	0	0	0
8	2	4	0	0
	51	100	26	100

IMPLICATIONS OF THE RESEARCH

Application systems that *never* require changes are quite rare. Professional developers have learned that users can always specify what is wrong with an existing system much better than they can describe what an imaginary system must do. This means that, by necessity, the development process is iterative. An initial application is developed, the user specifies the changes to be made; the developer makes the required alterations-- and the process continues until the user is satisfied that the information requirements are fulfilled.

If the application generator is incapable of regenerating the application system to include the requested changes (unfortunately, this system "tweaking" or fine-tuning often cannot be done with AGs), the new code must be created and inserted by the programmer. But if the automatically generated part of the code is complex in terms of size, number of paths, and/or nesting levels, the task of accomplishing the alterations becomes difficult and development time increases along with the user's dissatisfaction.

Our experience is that AGs are also frequently used by novices as a learning aid. Aspiring programmers may be well versed in the syntax of a language, but they become good programmers by looking over the shoulder of an expert programmer to learn the tips and tricks that come with experience. Many, if not most, end users do not have an expert programmer around to observe. The next best thing is to be able to study the work of an expert and many use the generator as a substitute for that missing guru. If the output of the generator is non-complex, then the learning process is enhanced -- and a good example is set; simplicity, after all, is a hallmark of the master programmer.

1. Schneyer, R., Modern Structured Programming, Mitchell Publishing, Inc., 1984, Santa Cruz, California, p. 151.

2. Weinberg, Gerald, The Psychology of Computer Programming, Van Nostrand Reinhold, New York, 1971.

3. Baker, F., "Chief Programmer Team Management of Production Programming," IBM Systems Journal, 11:1.

4. McCabe, Tom, "A Complexity Measure," IEEE Transactions on Software Engineering, SE-R, No. 4, December, 1976.

5. Martin, James and Carma McClure, Structured Techniques for Computing, Prentice-Hall, Inc., Englewood Cliffs, NJ, p. 49.

PERCEIVED IMPORTANCE OF TASKS IN THE SYSTEMS DEVELOPMENT PROCESS: THE IMPLICATIONS ON TEACHING SYSTEMS DEVELOPMENT

James R. King, Jr., Baylor University
G. W. Willis, Baylor University

ABSTRACT

One of the missions of Information Systems Departments in colleges and universities is to provide majors with exposure to the tasks involved in the system development life cycle. Unfortunately, what is taught in the classroom does not always parallel what is needed by industry. In an effort to determine what tasks are perceived as important to industry, a survey pertaining to the system development life cycle was conducted using Fortune 500 organizations. This paper relates the findings of the survey and how they should influence the teaching of system development tasks in information systems programs in colleges and universities.

Introduction

With the integration of "structured system development life cycles" and "structured development techniques and methodologies" into data processing departments of large organizations, it has become increasingly apparent that employees hiring into such firms need to have knowledge of these life cycles, techniques, and methodologies. The burden of providing this knowledge to the inexperienced major of Information Systems falls upon university Information Systems Departments and Programs. To provide this knowledge, the tasks involved in system development should be included in the curriculum.

The inclusion of these tasks in curricula requires that information systems faculty have an understanding of how important each task in the system development process is to industry and placing more emphasis on those that are more important and less on those that are not. As an information providing tool, a survey of Fortune 500 companies was conducted to determine the relative perceived importance of tasks in the system development process. The results indicate which tasks are perceived as most important to have knowledge of and which can be given lighter treatment in information systems education. In order to better understand what the survey was about, a brief overview of the system development process follows.

The System Development Process

The system development process can be broken down into subprocesses, each dealing with one or more aspects of the system development life cycle. This decomposition will be dependent upon one's viewpoint and experience. Since we are less concerned with which phase the tasks were placed and more concerned with the perceived importance of these tasks, there is no attempt to justify task placement within the phases. Instead, a modification of King's (2) traditional system development life cycle has been chosen to give a frame of reference. Then, tasks were gathered and refined from current systems analysis, system design, system development, and software engineering textbooks [Senn, (7); Whitten, Bentley, and Ho, (9); Peters, (4); Wetherbe, (8); Pressman, (6); Powers, Adams, and Mills, (5); and Adams, Powers, and Owles, (1)] and previous research [Ligon and Willis, (3)]. Each task was then placed into one of eight phases of the system development process. Figure 1 reflects the categorization of these tasks. The following section will explain how the tasks were studied.

Research Methodololgy

In order to study the relative perceived importance of system development tasks, a questionnaire was designed, tested, and sent to the "Director of Information Systems" of the Fortune 500 companies. Over 100 responses were received with ninety-nine being usable for the study giving a 20% response rate. The population selected was taken from the list of Fortune 500 companies in the March 15, 1985 issue of Datamation. The addresses of the 500 companies were found in Dunn and Bradstreet's Reference Book of Corporation Management. It is felt that the high response rate from extremely large organizations indicates a high current interest in the topics covered by the survey.

The survey instrument was designed to include a system development life cycle model based on the phases in King (2). Under each phase section on the survey, a number of related system development tasks or goals were provided and the respondent was asked to circle a response that indicated their perceived level of importance of the stated task or goal. The list of phases and related tasks is given in Figure 1. The following list shows the possible choices for the importance level for each question in terms of the respondent's level of agreement with the need of the stated task:

Strongly agree.....5
Agree.....4
Neutral.....3
Disagree.....2
Strongly disagree.....1

As the questionnaires were received, the data was entered into a computerized data base. The SAS Statistical Analysis Package was used to generate a frequency distribution of the responses for each of the tasks. A weighted average score was calculated for each frequency distribution and this weighted average score became the relative perceived importance score for each respective task. A summary of the scores is given by phase and task number (relating to the numbers in Figure 1) in Graphs 1 through 8. An interpretation of these graphs is given in the next section.

Results

A visual inspection of Graphs 1 through 8 serves as a beginning point in analyzing the results of the survey. These graphs show the relative importance of tasks within each phase. Graph 1 indicates that the most important tasks within the feasibility study phase are "to identify objectives", "establish a need for the system", "identify constraints", and "help users to understand the system". Graph 2 indicates that "user involvement", "communication between users and analysts", "defining the scope of the new system", and "generating a written statement of requirements" are the most important tasks within the requirements analysis phase. In Graph 3, only "considering alternatives" appears to be more important than the other tasks in the system specification phase. In the system design phase (Graph 4), the importance of "system controls" and "user/system interfaces" is apparent. There appear to be a number of very important factors in the coding and programming phase (Graph 5). These include the "use of work plans", "integration of program modules", "maintainability", "modular approaches", and "structured techniques". It is interesting that "prototyping" and "user-friendly languages" do not have high relative important scores. The most important factor in testing (Graph 6) appears to be "an evaluation of the results". Most interesting is the relative lack of importance of "data analysis and design", "pilot studies", and "prototyping" to the testing phase. In the documentation phase (Graph 7), the importance of "including documentation in every phase" is indicated as well and the importance of "documentation being clear and complete" is also shown. In the final phase, implementation (Graph

8), "training" appears to be the most important factor followed closely by the "involvement of operational management" in the process. As an indication of the importance of the groups of tasks (phases), an additional graph (Graph 9) has been included. This graph shows the average relative importance score for each phase in the study. As can be seen, the tasks in the feasibility study and the requirements analysis phases are perceived as most important while the tasks in the system specification and system design phases are much less important. The following section will provide some generalizations and conclusions based on these results.

Conclusions

Given the results of the survey, some generalizations and conclusions may be made concerning the teaching of these tasks in an information systems curriculum. Given Graph 9, it appears that much emphasis should be placed on the early phases of the system development life cycle. The average relative importance of the feasibility study and requirements analysis phases indicate that organizations need people that are good analysts. They need individuals that can fully think through and analyze a situation and come up with a conceptual solution to that situation. Within the feasibility study phase, educators need to show students how to evaluate whether a system is needed and emphasize the importance of defining the objectives of the system development process once the need for the system has been clearly established. Students must also be shown how to develop within a given environment which would include constraints, users, benefits, and costs. In the requirements analysis phase, it is very clear that professionals need the ability to integrate the user into the development process. This emphasis should be placed upon the students' ability to be a communicator of technical ideas in nontechnical terms. Also, he will need to be a moderator, innovator, and supporter for users. Thus, the student must pick up interpersonal and business skills as well as technical computing skills.

The survey also indicates a moderate level of importance for the coding/programming, testing, documentation, and implementation phases of the system development life cycle. A corresponding amount of educational time should be spent on the final phases of the life cycle. In programming classes, the technical concepts of structured programming and modularity must be emphasized. From experience, most students have a fundamental concept of these terms, but they do not understand them well enough to implement the concepts with any degree of consistency. Students should also be taught different methods for testing their systems and precisely what they are doing in testing. Testing should be more than an exercise in debugging programs. The results of the survey indicate very clearly how much emphasis firms place on well-developed, clear, and complete documentation that has been developed during the course of the entire system development life cycle. These ideas, as well as the concept of standardization, should be stressed throughout the information systems curriculum. For the implementation phase, interpersonal skills must once again be stressed. The most important criteria for this phase in the survey show that the system developer be able to work with the end-users at all levels. Some exposure to a project that has end-users as the final evaluator would be helpful in seeing how the communication skills of the developer would be used.

The two phases that received the lowest relative perceived importance scores were system specification and system design. This would indicate that firms place less importance on specifying the solution than they do on conceptualizing the problem and solution. This is important since many programs spend a considerable amount of time on the teaching of specific design and specifi-

cation tools and languages. Also indicated is the need for more teaching of automated specification and design tools. Once again, there seems to be a need for the integration of the end-user in the development process. Also, the specifications and design must be done with the end-user in mind.

In summary, it can be seen that the Fortune 500 firms do perceive a difference in the importance of the phases of the systems development life cycle. This means that information systems educators must be cognizant of these perceptions and modify their teaching methods to rectify any omissions in their programs of study. They must place a respective amount of time on studying the phases according to the average relative perceived importance levels of the phases. In addition, within each phase they must put particular emphasis on the tasks that had the highest scores. If educators make an attempt to provide students with a working knowledge of the most important tasks and a lesser knowledge of the lesser important tasks, students will be better prepared to enter the profession.

References

1. Adams, David., Powers, Michael J. and Owles, V. Arthur. Computer Information Systems Development: Design and Implementation. Cincinnati, Ohio: South-Western Publishing Co., 1985.

2. King, David. Current Practices in Software Development. New York, New York: Yourdon Press, 1984.

3. Ligon, Helen and Willis, G.W. "Systems Development Life Cycle - Delphi Revisited," Proceedings of the Southwest Region of American Institute for Decision Science, March, 1985.

4. Peters, Lawrence J. Software Design: Methods and Techniques. New York, NY: Yourdon Press, 1981.

5. Powers, Michael J., Adams, David R. and Mills Harlan D. Computer Information Systems Development: Analysis and Design. Cincinnati, Ohio: South-Western Publishing Co., 1984.

6. Pressman, Roger S. Software Engineering: A Practitioner's Approach, 2nd edition. New York, New York: McGraw-Hill, Inc., 1987.

7. Senn, James A. Analysis and Design of Information Systems. New York, New York: McGraw-Hill, Inc., 1984.

8. Wetherbe, James C. Systems Analysis and Design: Traditional, Structured, and Advanced Concepts and Techniques, 2nd edition. St. Paul, Minnesota: West Publishing Co., 1984.

9. Whitten, Jeffrey L., Bentley, Lonnie D. and Ho, Thomas I.M. Systems Analysis and Design Methods. St. Louis, Missouri: Times Mirror/Mosby College Publishing, 1986.

Figures and graphs may be obtained by writing the first listed author.

FACILITATED TEAM TECHNIQUES APPLIED TO REQUIREMENTS SPECIFICATION

Diane L. Lockwood
Seattle University

Facilitated team techniques (FTT) consist of a series of structured meetings designed to extract high-quality business system specifications from users and analysts within a compressed time frame. The advantages of FTT over interviews, some key features of FTT, benefits, and the conditions where FTT are appropriate will be discussed.

INTRODUCTION

I am going to introduce this paper by describing, in generic terms, what systems analysts do to gather and analyze data in support of a requirements specifications for business systems. Typically, we will proceed to gather data from the usual sources (e.g., forms and documents, procedures manuals, observation, work rate and volume sampling), but most importantly we will obtain information from the people directly involved in the business activity usually by conducting personal interviews. After concluding the initial round of interviews, we recognize that the individuals involved are frequently in disagreement with one another on some element of system scope, problem definition, process or functional descriptions, or input/output requirements, to name a few. Occasionally we even recognize that these disagreements are honest ones based on different people's perceptions of the way things are and the way they should be. Nevertheless, we have to come to a timely resolution regarding these issues if we are ever going to get the system implemented within the mandated time frame. So we go back again to interview people to try and resolve disagreements and inconsistencies. After still more "back and forth" interviews, we finally resolve the remaining loose ends in the best way we can given our collective knowledge of the system and technical expertise. However, several issues we thought were previously resolved among interviewees became the subject of future change meetings, additional interviews, and downstream maintenance of the system.

In the scenario presented above, one should take particular notice of two points. First, each user has a different viewpoint and this is important information because changes to the system or a new system must attempt to satisfy as many users as possible. Second, when inconsistencies arise in requirements specification, it is typically the analyst, not the users themselves, who are forced to resolve differences. Consequently, there will always be some people dissatisfied with the system.

Achieving good communications between users and developers has been a longstanding goal, but how can systems professionals deal with the nagging problem of getting top management and users to really pay attention to system details during the crucial early phase of requirements specification? Furthermore, how can honest disagreements in requirements specifications be resolved in a timely manner and with

a consensus of the people involved? Several techniques such as prototyping, the use of data flow diagrams during interviews, and presentation graphics in analysts workbenches have been offered to help bridge communication gaps, but most of these are still oriented toward a single-user requirements specification environment (i.e., one user and an analyst versus a "team" of users). This does not mean that these techniques are not applicable in a team environment, but rather that their use in practice is usually confined to a single-user data gathering method. In such situations, the problem of resolving multiple user views in a requirements specification still remains.

Facilitated team techniques (FTT) are an alternative (not a replacement) to conventional data gathering and analysis methods. The primary purpose of FTT is to develop high quality systems faster through the use of team sessions designed to achieve consensus among users in requirements specifications for large-scale application systems. What are some key features of FTT? What are the benefits? Under what conditions are FTT appropriate? What are some caveats to be aware of with FTT?

WHAT ARE FACILITATED TEAM TECHNIQUES?

Facilitated team techniques consist of a series of structured meetings designed to extract high-quality business system specifications from users and analysts in a compressed time frame (=< 2 weeks). There are a small number of commercially available FTT which vary in their focus (e.g., analysis and/or design phases), underlying theoretical and methodological framework (e.g., data and/or process orientation), level of detail, specific procedures employed, and documentation requirements. Most FTT, however, have four key features: (1) structure, (2) workshop format, (3) an impartial session leader, and (4) consistent documentation.

Meetings with teams of users and analysts to define requirements is not new in itself, but the degree of structure associated with formal FTT is what keeps a team meeting from becoming just another uncontrolled brainstorming session. In order to be considered a structured approach, FTT must have: a) a well defined objective, b) a well defined scope, c) a well defined deliverable, d) a structured agenda capable of being replicated, e) a proven system development method, f) a carefully chosen, representative Team of participants, and g) a neutral, unbiased session leader.(1) The size of the team depends upon the scope of the system, which in turn depends on a clearly defined set of objectives. However, the ideal group size is from 12 to 15 people. The makeup of the team is typically 10% executives, 20% managers, and 70% operational level people. Systems analysts and data administrators do not typically contribute directly to the requirements definition and analysis document during FTT (they are observers in team sessions), but they are needed to collect information for subsequent phases of design and implementation and to identify requirements which may cause difficulty later on.

A workshop format, a second key feature of FTT, helps attendees concentrate on exchanging ideas and avoiding politics. (2) Team members are guided through a well-ordered sequence of workshop tasks such as Brainstorming, Analysis of the brainstormed information according to a prespecified set of questions, Organizing (logical groupings) and Documentation (1). Wall-size charts are typically posted during FTT sessions to enhance the requirements specification process and to

facilitate adherence to the workshop agenda.

An impartial session leader, a third key feature of FTT, must at least be perceived by team members to be neutral (i.e., have no vested interest in the outcome of the requirements specification). For this reason, an outside systems consultant (more accurately called a "facilitator") will often be used to facilitate a FTT session. A primary role of a facilitator is to guide the team through all steps of the requirements specification process and documentation procedures in a structured, timely, and collaborative manner. Facilitators are challenged to perform in four distinct arenas: a) systems theory and development methods, b) project management and reporting, c) training groups in concepts, and d) managing group processes. (1) They should remain neutral on content issues.

Consistent documentation is also a key feature of FTT. Business functions and information needs are recorded usually (but not always) by team members themselves on pre-formatted documentation forms during the FTT workshop. The use of preformated forms helps to insure consistency in documentation standards. Content of FTT documentation and deliverables will vary depending on the specific commercially available technique being used. In many cases, the completed forms become the requirements specification to be used as input into the subsequent phase of system design. The resulting requirements specification should also become the basis for ongoing project monitoring.

WHEN SHOULD FTT BE USED?

Conditions that are conducive to FTT include size of system and level of detail, extent of diversity of opinions among users, geographic or functional isolation of users, extent of systems integration required, existence of a need to avoid over-dependence on project leaders or systems analysts, and the time available to produce a requirements specification. Paper length limitations prevent the explication of these conditions here, but they will be discussed during the actual presentation. In general, a conventional approach like interviews should be used where the project is small and there is agreement among key users regarding the objectives and priorities of the system.

BENEFITS OF FTT

Primary benefits of FTT are increased productivity, learning, and commitment.(1) The data and decisions needed to complete a requirements specification are collected and documented usually within a two week or less time frame. This remarkably high level of productivity during system development realizes financial and other resource gains.

From the beginning of a FTT session, team members will be sharing their different perspectives and learning about different aspects of their company and its system(s). It is probable that some of them will meet and talk for the first time. This learning environment enhances communication about the interrelatedness of the functions within their system and between the system and its external interfaces. A corporation's employees possess the best knowledge about what a system is supposed to do, but they do not always know how to communicate this in accordance with accepted systems development practices. Team sessions therefore also provide a useful vehicle to educate users about the format, rigor, and level of detail needed for requirements specification. Finally, team members develop a strong sense of ownership of the

requirements specification which fosters commitment among users and the development staff. There is some empirical support for the argument that user involvement increases the perceived usefulness of information systems. (3)

IMPORTANT CAVEATS TO FTT

If FTT are incorrectly or poorly implemented, then the end results are often worse than if they had never been used. The same could be said of any system development technique. Examples of incorrect or poor implementation could include the use of an unskilled facilitator (a major problem), lack of a sound theoretical systems development foundation, inappropriate team selection, inadequate planning and top management support, and incomplete documentation, to name a few.

The most demanding feature of FTT is that they require a great deal of sustained energy and heavy time commitment from all team members during consecutive, full-day sessions for a period of weeks. Indeed, team sessions have often been characterized by participants as "educational, but grueling" and rightfully so because systems are complex and demanding in detail. FTT sessions also take participants away from other daily work responsibilities for the duration of the sessions. It takes an understanding and committed top management to approve full-time participation of a number of their key personnel. How to obtain top management commitment to FTT is the subject of much debate. A company must begin to seriously analyze the costs associated with conducting numerous one-to-one interviews that result in incomplete or inconsistent specifications. What are the costs associated with downstream propagation of specification errors because users are never forced to understand and resolve differences among themselves initially in a team environment? Stated positively, what would it be worth to a company from a competitive viewpoint to deliver a high quality requirements specification in 80% of the elapsed time that it takes using traditional interviewing and other data gathering techniques?

CONCLUSION

Practicing systems professionals suggest that it is far easier to convince budget-minded executives to invest in tangible hardware and packaged software, than it is to invest in the "people" costs associated with FTT. Nevertheless, when compared to traditional methods, FTT allow more information to be gathered faster and more accurately from the people who possess the most knowledge about what a system is supposed to do -- a corporation's employees.

REFERENCES

1. Burner, H. Blair. "Facilitated Team Techniques," WISDM Newsletter, vol. 2, February 28, 1987.

2. Canning, Richard (Ed.). "Developing High Quality Systems Faster," EDP Analyzer, vol. 24, no. 6, June 1986.

3. Franz, Charles R. and Daniel Robey. "Organizational Context, User Involvement, and the Usefulness of Information Systems." Decision Sciences, vol. 17, no. 3, 1986.

4. Rush, Gary. "A fast way to define system requirements," Computer World, October 7, 1985, pp. In-Depth 11-16.

SYSTEMS PERFORMANCE EVALUATION: A COURSE FOR WHICH THE TIME HAS COME

Judith L. Solano
Division of Computer and Information Sciences
University of North Florida, Jacksonville, Florida 32216
(904) 646-2985

ABSTRACT

The Division of Computer and Information Sciences at the University of North Florida has implemented an advanced undergraduate, beginning graduate course in systems performance evaluation. The course is designed to introduce students to the tools and techniques used in the evaluation of the performance of computer systems, as well as to the decision making processes involved in systems tuning and capacity planning. Using data from existing systems, students actually conduct performance evaluation studies in partial fulfillment of the requirements of the course. This is a course for which the time has come. All organizations that make use of computer systems must take performance factors into consideration. Students should be familiar with the tools and techniques that will enable them to assist their organizations in avoiding or eliminating the waste of costly computer resources.

INTRODUCTION

The Division of Computer and Information Sciences at the University of North Florida has implemented an advanced undergraduate, beginning graduate course in systems performance evaluation. The course is designed to introduce students to the tools and techniques used in the evaluation of the performance of computer systems, as well as to the decision making processes involved in systems tuning and capacity planning.

All organizations that make use of computer systems must take performance factors into consideration when dealing with problems of procurement, configuration design, system tuning, upgrading, accounting and pricing, scheduling and operations management, and short and long term planning. A system that does not perform as expected, or whose cost/performance ratio can be decreased, causes a waste of the organization's resources which should be avoided or eliminated.

The study of systems performance evaluation is not a new phenomenon. However, its practical application to everyday problems is not currently particularly widespread. This is caused in part by few professionals in the field with the necessary knowledge and understanding of performance evaluation tools and techni-

ques. The goal of the University of North Florida's course is to provide an audience of students with knowledge about the latest developments in performance measurement technology. The expectation is that the students will be prepared to establish and maintain an effective performance monitoring function in the organizations in which they will work.

COURSE OUTLINE

The systems performance evaluation course is organized in accordance with the phases and steps of a performance improvement method. Using Ferrari, Serazzi, and Zeigner's text, Measurement and Tuning of Computer Systems, the content of the course is divided into a treatment of the objectives definition, workload characterization, instrumentation selection, experiment design and execution, and results interpretation phases of the method. In addition, considerable amount of time is devoted to both system and program tuning, as well as capacity planning activities. The course concludes with a discussion of economic considerations, in order that students will finish the course with an appreciation of the economic risks involved in making an investment in performance evaluation studies.

Frequently performance evaluation studies are undertaken in response to complaints from users, who suspect that the system must be capable of performing far better than it seems to be. For example, users of interactive systems are particularly sensitive to response time and are rarely satisfied with the response time that they are experiencing. Studies undertaken in response to such complaints have a tendency not to systematically evaluate the system. They rarely include a definitive statement of objectives and tend not to adhere to basic principles of experimental design and analysis. The systems programmers or performance analysts are likely to arrive at obvious conclusions based on experiences and intuition. Their study will involve a search for performance data that will support their conclusions.

The systems performance evaluation course begins with an extensive discussion of a performance improvement method. The intent is to insure that students will understand from the beginning of the course that there is more to undertaking a performance evaluation study than just playing a hunch. Performance evaluation studies should be undertaken as systematically, and with as much attention to detail, as any other scientific experiment. Students learn that there is a prescribed set of tasks that must be addressed and a logical flow of control among those tasks. Students come to understand, prior to getting into the material of the course, that the organization of the material and its presentation will be based on the method. The method serves as the framework for the course.

The objectives definition phase of the method is that phase in which decisions are made about the type of evaluation that is to be conducted, the primary object of the study, and the evaluation techniques that are appropriate. The course stresses that evaluation studies are not limited to the performance problems that arise in existing, running systems. Performance evaluation studies can be classified into four groups: those involving the creation of a new system, the choice of a system, the improvement of an existing system, and the prediction of when a system's capacity will be exceeded. It is also noted that while computer systems, defined as consisting of hardware components and operating system software, tend to be the primary object of most studies, it is not at all uncommon to study the affects of the performance of application programs. Students are appraised that their decisions, in this phase, regarding the type of study to be performed and the object of the study, will influence decisions regarding evaluation techniques to be employed. Direct mea-

surement, simulation modeling, and analytic modeling represent the types of evaluation techniques that are appropriate, depending of course on the type of study and the object of the study.

At the conclusion of the objectives definition phase of the course, there is an understanding that this course will limit itself to a treatment of the class of evaluation studies designed to examine existing, running systems. Using direct measurement techniques, students will study both computer systems and application programs. The intent of these studies will be to gather information that will assist in the making of decisions that will result in the improvement of the system's efficiency and in the improvement of management's ability to predict future computer resource needs.

The workload characterization phase of the course is potentially the most difficult for the student, in terms of content, and the most dangerous for the instructor, in terms of presentation. Workload is a term used to represent all the input received and processed by a system. It is, for example, the programs, the data, and the commands that a system must contend with. A performance evaluation study can not be conducted independent of an understanding of the workload. A performance analyst must be able to specify the characteristics of the workload in quantitative terms. There are a variety of types of workloads that may be employed in a study, and herein lies the danger for the instructor and potentially the difficulty for the student.

This course in systems performance evaluation limits itself to a cursory overview of synthetic and artificial test workload models. A detailed discussion of these types of models would require a sophistication in statistical sampling, forecasting, and in simulation and analytic modeling that most undergraduate students do not have. To provide the type of background for some students and review for others that would insure that this material would be understood, would require the instructor to devote a considerable amount of time to this phase of the course. Without careful attention to the amount of time spent at this juncture, the instructor is in danger of finding himself without adequate time later in the course for equally important phases in the course.

To avoid the potential pitfalls of this phase, instruction in this course concentrates on real test workloads. Real workloads are those programs and data that are actually processed by the system. There is no attempt to create a representative model of the "real thing." The "real thing" is what is used. Students are taught to identify quantitative parameters that can be used to characterize the workload, such as CPU time, disk I/O operations, disk files used, lines printed, etc. There is discussion about the appropriate times when a measurement session should begin and end, which is the only decision that the performance analyst must make when dealing with a real test workload.

The principal activity of the instrument selection phase of the method is the selection of a measurement tool that will capture, quantify, and present data relative to the performance indices that are to be studied. Time is spent in the course, at this phase, ensuring that students have a good understanding of the various performance indices. Turnaround time, response time, and throughput are all examples of visible, and therefore, popular indices of processing efficiency. Students are cautioned to be aware that there are many other indices or variables that will have an effect on, for example, response time. CPU utilization, channel utilization, disk utilization are all examples of indices that provide information about the efficiency of usage of various system components, all of which will have an effect on response time. The choice of the appropriate measurement tool will

be dependent in large part on the kinds of indices that are to be measured. Students are taught the comparing and contrasting characteristics of software and hardware monitors, with an emphasis on the applicability of these tools to the measurement of various system variables.

Once the measurement tool has been selected, the remaining decision to be made, before conducting the performance evaluation study, is that of which measurement technique to employ. The experiment design and execution phase of the course is designed to introduce students to the principles of two measurement techniques commonly used when analyzing real test workloads, or those programs and data that actually are processed by an existing system. Event detection, a technique that involves capturing all changes in a particular system state and recording those changes in the order in which they occurred, is discussed with particular emphasis on its advantages and disadvantages. In addition, considerable time is devoted to the sampling technique, a technique that is often preferred to event detection.

Sampling also involves the capturing of changes in a particular system state, but in sampling one captures the changes at regular intervals. Sampling is a statistical technique that results in a much smaller amount of data. The advantages of sampling include less interference with the performance of the system during data collection, and a shorter and less complicated process of data analysis. Sampling, therefore, can potentially be a more accurate and less expensive technique.

For students to be able to understand how to select a sample that will accurately represent the system that they intend to measure, they require a thorough review of basic statistical principles. To this end, the course includes a discussion of the random sampling technique, with emphasis on such terms as sampling distribution, mean, standard deviation, confidence level, confidence interval, and confidence coefficient.

Conclusion of the discussion on measurement techniques marks about the midpoint of the systems performance evaluation course. Of the time remaining in the course, almost 60% is devoted to the last phase of the method, the results interpretation phase. While it is important that students have a thorough understanding of the preceding phases, it is likely that they will find that their ability to put this knowledge to use will be limited. Many of the activities that are performed in the preceding phases are performed by automated tools. Most of the decisions that must be made may well have been already made. Therefore, one of the more valuable skills that a student can derive from this course will be the ability to make some sense out of the large volume of data that is already being captured.

The results interpretation phase of the course is taught using samples of real performance evaluation output from three different installations. The data from the first installation represents of a large commercial mainframe environment. Data from the second installation is from a university mainframe dedicated to instruction and research activities. And, the third set of data is from a small multi-user minicomputer installation.

Students are taught, using the three different sets of output that are primarily in tabular form, where to look to find the data they will need for their analyses. They are taught to use simple and easy to use mathematical algorithms that will allow them to process the data and draw conclusions. And, they are reminded of the variety of graphic techniques that can be employed to represent the data so that it is easy to interpret and compare.

Systems performance evaluation should be an ongoing activity. When the organization's workload changes and/or grows, interpretation of the results of the performance evaluation will suggest that the system is not meeting the service level expectations of its users. An alternative, at this point, is to engage in activities designed to tune the system and thereby improve its performance.

The systems performance evaluation course includes a discussion of both system and program tuning activities. Using case studies in Ferrari, Serazzi, and Zeigner's text, several tuning problems are described. Students are encouraged to discuss potential solutions to the problems. The discussion emphasizes techniques for the reconfiguration of the systems in question and the balancing of the workload on the components of those systems. Consideration is also given to techniques for the optimization of programs and program mix.

On the assumption that tuning activities might prove to be, at best, short-term solutions, students are also introduced to workload forecasting and capacity planning. The course includes the presentation of a simple technique, using current utilization figures, for estimating when new acquisitions will be required. Also included is a discussion of a more detailed, systematic process for forecasting future workload and determining required capacity to handle that workload.

Most systems performance evaluation studies are undertaken with one objective in mind, that is to reduce the system's cost/performance ratio. The studies themselves can prove to be costly, if one considers the cost of the personnel involved in the study, the cost of measurement tools used, the cost of data collection and reduction, and finally the cost of doing the tuning or making the new acquisitions that might be suggested by the results of the study. It is imperative that students understand the risks that might be involved in making an investment in evaluation studies. The course concludes, therefore, with a discussion of economic considerations, emphasizing various techniques that can be used to do cost-benefit analyses.

COURSE PROJECTS

In partial fulfillment of the requirements of the systems performance evaluation course both undergraduate and graduate students are expected to complete two evaluation studies. The students are provided access to two weeks of performance evaluation data collected from an IBM 4341 and two weeks of data collected from an AT&T 3B2. The expectation is that they will write two reports. One report will provide an analysis of the data from the mainframe environment and the other will provide analysis of the data from the multi-user minicomputer environment.

Students are told that their reports should reflect the contents of the course material. The organization of the reports should model that of the performance evaluation method that they have learned. Each report should include a description of the objectives of the study and a quantitative characterization of the workload that was processed by the system being studied. The report should also contain a description of the measurement tool that was used to capture the performance evaluation data.

A two week sample of data represents a large volume of data for students to sift through and analyze. Often they elect to work with a smaller subset of the two week sample. Their reports must include a description of the sample that they have elected to use and a discussion of the relative accuracy of the sample.

The reports conclude with an interpretation of the results and with recommendations. Students identify existing performance problems and forecast the po-

tential for problems in the future. Their recommendations include descriptions of how either the system or the workload might be tuned, in an effort to provide a solution to existing performance problems. They also recommend a schedule for the acquisition of more capacity, should the data suggest that there is a potential for problems sometime in the future.

CONCLUSION

The University of North Florida's course in systems performance evaluation is designed to provide students with a firm foundation in a performance improvement method. The strength of the course lies in the course projects that the students are expected to complete. Using data from existing systems, they are able to conduct performance evaluation studies. It is expected that students will be able to take this knowledge with them into the workplace and use it to establish and maintain a performance monitoring function.

This course is one of only four of its kind being offered by public institutions of higher education in the State of Florida. It is a course for which the time has come. Any institution offering a program of study in computer and information science should consider offering such a course. Students should be familiar with the tools and techniques that will enable them to assist their organizations in avoiding or eliminating the waste of costly computer resources.

BIBLIOGRAPHY

1. Allen, A. O. Probability, Statistics, and Queueing Theory With Computer Science Applications. Academic Press, Inc. Orlando. (1978).

2. Ferrari, D. Computer Systems Performance Evaluation. Prentice-Hall, Inc. Englewood Cliffs. (1978).

3. Ferrari, D., Serazzi, G., and Zeigner, A. Measurement and Tuning of Computer Systems. Prentice-Hall, Inc. Englewood Cliffs. (1983).

4. Foxley, E. UNIX for Super-Users. Addison-Wesley Publishing Company. Wokingham. (1985).

5. Letmanyi, H. Guide on Workload Forecasting. U.S. Government Printing Office. Washington. (1985).

6. MacNair, E. A. and Sauer, C. H. Elements of Practical Performance Modeling. Prentice-Hall, Inc. Englewood Cliffs. (1985).

7. Strauss, M. J. Computer Capacity: A Production Control Approach. Van Nostrand Reinhold Company. New York. (1981).

INTERLOCKING IS WITH THE ORGANIZATION

Mary R. Lind

Ph.D. Student, UNC - Chapel Hill

Assistant Professor, N.C. A&T State University

ABSTRACT

The role of IS has changed from traditional computing activities to include the dissemination of CBIS technology into the organizational components. Effective IS management in the face of these new responsibilities will require that IS position itself to maximize its communication with the rest of the organization.

Many large organizations formed data processing departments in the 1960's to use computer technology to handle routine financial applications. By the mid-seventies this technology had spread into smaller organizations. Currently, as a result of the low cost and availability of computer technology, it is being used by organizations of all sizes.

During the 70's, organizations began moving the data processing department out of the financial component of the organization and often positioned them on the organizational chart as a staff support group. The name assigned to data processing in many organizations was changed to management information systems or information systems. (In this paper the term used to describe this function is information systems or IS). In the 80's, many organizations are moving the IS function out of staff positions and placing them in line positions (1).

This movement to a position of greater responsibility and accountability has occurred along with a change in the role of the IS function. As Zmud (10) indicated in his discussion of design alternatives for IS, the role of the IS function has changed from that of manufacturing to manufacturing and distribution. This means that IS no longer acts solely as a data processing manufacturer that receives data, places this data into appropriate formats, processes it with computers, prints the results, and delivers them to the appropriate parties. Computer based technology has changed so that the end-users of it no longer have to be 'experts'. These end-users may be located in any of the functional areas in the organization and may be at any level in the organizational hierarchy - ranging from strategic, tactical, operational, to non-management employees. Because of the importance of this technology in their day-to-day activities, these end-users are acting independently of the information systems group in deciding when and how to integrate computer based technology into their work activities (5).

Thus, IS's roles for the 1980's and 1990's have become that of a technology manufacturer and a technology disseminator. As technology manufacturer, IS manages the acquisition and operation of computer hardware and associated peripherals, performs large scale software development and maintenance activities, and provides a technological infrastructure. This infrastructure is the core ability of IS to support the organization's information processing activities by providing computer hardware, software, databases, and communication capability. In its role as technology disseminator, an important activity is the promotion of new computer based technologies and software into other organizational components. In addition to 'selling' potential end-users on the merits of this

technology, IS must educate these end-users both before and after the decision to adopt the technology. End-users are taught not only the technical nuances of the hardware and software but also how to effectively integrate the technology into their work activities. This includes teaching them the differences between data and information and how to use the computer as a tool to aid in both quantitative and qualitative decision making. Additionally, in this role IS must provide support to end-users after they begin to use the technologies. This can involve providing personal responses to technical questions, technical support in developing new applications, or merely keeping them informed of changes in the technology via newsletters. Besides facilitating the flow of technology to end-users, IS in its role of technology dissemination must establish and enforce standards and controls for this technology. Providing adequate security and quality control standards were relatively easy for computer based information systems used in a manufacturing mode; but with the proliferation of end-users, enforcing these standards and controls for end-user computer based information systems is a more difficult task.

Thus, IS management has two distinctly different roles to perform and must have effective communication with the rest of the organization in order to perform these roles. The dilemma faced by IS is how to interlock this communication to ensure that both the IS roles of manufacturing and dissemination are satisfying the computing requirements of the organization and providing ideas on innovative new uses of the technology. In this context, communication interlock implies that:

- there is clear and rapid information flows between the IS roles (manufacturing and dissemination) and other organizational areas

- there is clear and rapid communication between the IS roles (manufacturing and dissemination)

- there is some communication between the IS roles (manufacturing and dissemination) and sources of information outside of the organization (customers, vendors, professional contacts, etc.)

The following story provides an interesting approach for management of these IS roles. Simon (8) introduced two hypothetical watchmakers named Hora and Tempus. Both men were interrupted frequently in their work by people calling on the telephone to order watches. The watches made by each man contained 1000 parts. The watches Tempus made were constructed so that if one of them was only partly assembled and he had to put it down, it fell apart and had to be reassembled. Thus, if Tempus got a phone call before a watch was finished, he had to start all over. Hora's

watches were just as complex as Tempus's, but they had been designed so that they could be put together using subassemblies of ten parts each. Ten of these subassemblies in turn could be assembled into a larger subassembly, and ten of the larger subassemblies could be assembled into a 1000-piece watch. When Hora was interrupted, he lost only a small portion of his work. The result was that if there was one chance in a hundred that either watchmaker was interrupted when he added a part to an assembly, then it would take Tempus 4000 times as long to assemble a watch as it will take Hora. This story demonstrates the importance of segmenting tasks. Since IS has two very different roles to perform, the suggestion of this story is that separate groups within the organization could be formed to handle each role. Separating IS into manufacturing and dissemination components will result in two groups that are loosely coupled (9) and able to specialize in their respective tasks with relative independence from each other.

Allen (1, p. 269) in proposing the importance of proximity in communications stated as a result of his research that,

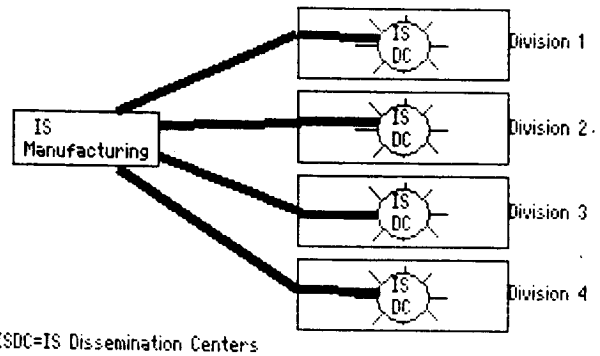
> "A prime determinant of communication choice is the physical distance separating the parties in the organization."

As was discussed, IS is usually positioned in the organizational structure as a staff or line position. The crossing of organizational boundaries to deal with other divisions creates barriers to communications for IS. The most obvious barrier is the physical one with the IS people housed together in an separate area of the company. An organizational approach for breaking these physical communications barriers is to place the responsibilites for IS dissemination into IS Dissemination Centers (ISDC's) which are located in each organizational division. IS Manufacturing would remain as a staff or line position. The ISDC's would report to the management of the division in which they are located, thus, their loyalities as well as their physical location would be with that division. They would be evaluated based on their level of success in disseminating technology into their division, in coordinating large scale information system development between the divisions and IS Manufacturing, and in providing feedback to IS manufacturing on the service level of the technological infrastructure. Effective performance of these duties requires tightly coupled communication ties both within their division and with IS Manufacturing. While IS Manufacturing and the ISDC's are decoupled in terms of their work roles and administration, it is necessary that they remain tightly coupled in terms of the sharing and exchange of technological information (communication interlock).

The structure suggested is shown in Figure 1.

Figure 1

IS/Organizational Interlock



ISDC=IS Dissemination Centers

This figure indicates that each division in the organization contains an ISDC that is interlocked with that division's communication channels. Additionally, the ISDC's are interlocked with IS Manufacturing in order to serve as a liaison between the divisions and IS Manufacturing. Thus, through this structural arrangement the ISDC's are serving as conduits of information between (1) themselves and the divisions in which they are located and (2) between themselves and IS Manufacturing.

The people chosen to work in the ISDC's must be knowledgeable of both computer technology and how to relate this technology to business needs. In addition to technology and business management training, they must be good communicators - able to translate technology into terms understandable by management and non-management personnel. Also, they need to be in touch with many information sources both within the organization and outside of it (customers, vendors, consultants, professional contacts). Thus, the people working in the ISDC's must have interlocked communication within their division, with IS Manufacturing, and with other professional contacts inside and outside the organization. This description fits that of the technological gatekeeper (1). While employees can be trained in technology and in business, it is not clear that people can be trained to be gatekeepers. Allen (1) stated that the people in Research & Development labs who became technological gatekeepers assumed that role informally due to their inherent communication ability. Thus, the organization needs to be able to identify those individuals to work in the ISDC's who have the necessary technological and business skills along with the inherent penchant for gathering and distilling technical information.

Interlocking communications between IS and the organization involves three components. One component is structural - positioning IS so as to maximize the opportunity for communication flows. Another component consists of placing people in the ISDC's who are natural communicators and competent in technological and business areas. A third more basic component than these, though, for interlock is the provision of effective means (communication channels) to facilitate the flow of communication among IS Manufacturing, the ISDC's, and the divisions. While it is generally accepted that communications is the glue that holds organizations

together, effective communications between the divisions and IS Manufacturing and the ISDC's is particularly important for IS to effectively span organizational boundaries. Providing effective means of communication between the IS components and these areas will promote interaction among them. End-users can make known their wishes for ways to enhance their work activities, and the ISDC'S can inject into functional areas new technological approaches to perform their work. The key to this interaction is that both groups (IS and the divisions) are familiar with the work activies of each other so that each can make useful recommendations and requests. Also, of importance is communication with entities outside organizational boundaries. Contact with vendors, customers, or professional associations can serve as sources of new innovative ideas. This knowledge of what other organizations or others in the same organization are doing may create the perception of a performance gap (7) and cause individuals in the organization to push for new innovations to eliminate the gap.

As a technology disseminator, the ISDC'S need to anticipate technological change and establish conduits to diffuse technology into the organization. For the ISDC'S, communications both inside and outside the organization is critical in facilitating this diffusion. Various means or communication channels are available for handling communications. These include personal, group, or impersonal communication channels. A few of these are computer based technologies, however, most involve traditional technologies. Because of the importance of IS/divisional communications and because of the time constraints faced by both groups, selecting the most effective communications media is extremely important.

Chester Barnard (3, p. 226) said,

> "The first function of the executive is to establish and maintain a system of communication."

This statement could be modified to the following - the first function of IS is to establish, maintain, and use a system of communication. Mintzberg (6) reported these findings: managers spend 90 percent of their working day talking to people, foremen perform a different activity on the average of once every 48 seconds, managers work for 30 minutes uninterrupted on the average of once every two days, etc. Goldhaber et al. (4) indicated that a great deal of managers' information is received serendipitously - the information happened to be available and people paid attention to it, the information is received because other people in the personal contact network thought it would be of interest, and some of the information is received as a result of personal effort. Thus, communications consumes most of the working day of the professional employee and

tends to be rather haphazard in nature. The IS professional whose job it is to service, support, and initiate new technology into other organizational components is bound into a complex web of communications with the divisions, with the ISDC's, with IS Manufacturing, and with vendors and professional contacts outside the organization. Thus, organizations should manage the structural design, staffing requirements, and provide communications media to ensure that IS/organizational interlock occurs.

References

(1) Allen, T. (1977). Managing the Flow of Technology, Cambridge, Mass., MIT Press.

(2) Allen, B. (1987). Make information services pay its way, 87(1), 57-65.

(3) Barnard, R. (1938). The functions of the executive, Harvard University Press, Cambridge, Mass.

(4) Goldhaber, G.M., Rogers, D.P. (1979). Auditing Organizational Communication Systems: The ICA Communication Audit, Kendall/Hunt Publishing Co., Dubuque, Iowa.

(5) Lind, M.L. (1987). Facilitating end-user computing, Proceedings of the Decision Sciences Institute, Boston, Mass.

(6) Mintzberg, H. (1975). The manager's job:Folklore ad fact. Harvard Business Review, 53(4), 49-61.

(7) Rogers, E.M. (1983). Diffusion of Innovations, The Free Press, New York.

(8) Simon, H.A. (1962). The architecture of complexity. Proceedings of the American Philosophical Society, 106(6), 467-482.

(9) Weick, K.E. (1976). Educational organizations as loosely coupled systems, Administrative Science Quarterly, 21, March, 1-19.

(10) Zmud, R.W. (1984). Design alternatives for organizing information system activities. MIS Quarterly, 8(2), 79-93.

THE GENERIC STRUCTURE OF THE ENTERPRISE-WIDE IMS

Dr. Andrew S. Targowski
and
Dr. Margaret M. Sanders
Department of Business Information Systems
Western Michigan University
Kalamazoo, MI 49008

ABSTRACT

The accelerated progress of CIS implementations requires the holistic view of the enterprise-wide computerized environment in education and practice. The criteria of the applied system purpose, the logistics substance, the information contents defines the generic structure of Information Management Systems. This structure serves as a factor in the selection of the scope and priorities of IMS projects.

INTRODUCTION

The purpose of this paper is to provide the scientific framework for the allocation of IMS (Information Management Systems) in the enterprise-wide environment. This framework is needed for the CIS educators as well as for the systems planners, designers, and implementors.

A common goal behind this framework is to link all the pieces of IMS together in an enterprise-wide computerized environment. One example of this approach is computer-integrated manufacturing (CIM), which links IMS from business, engineering, and production.

The scientific categorization of IM systems semantics processing and applied organizations based on the exclusive criteria is the first prerequisite for the correctly harmonized integration of IM systems through effective planning, designing, and implementation phases of systems deliver to the end-users and organization.

In education, at the undergraduate (introduction to CIS) and graduate (CIS for MBA) levels the place of IMS in the enterprise-wide environment is covered in a sketchy manner. In the majority of academic textbooks the idea of IMS is presented in such a way as to impress the student. The student reads in the textbooks about the variety of computer information systems but does not know where the systems "begin and end." In other words, the student is not equipped with tools that allow him to correctly evaluate the computer progress in the business environment.

The same situation is true in the IMS practice, where a holistic view of the computer-based enterprise-wide environment almost does not exist. Progress in business IMS is more motivated by the computer vendors' products than by the users' awareness of their own needs. A good example is DSS which after a few years of experience brings some disappointment as a limited tool for executive decision making.

This paper presents a method of allocating IMS in the business environment based upon its generic structure, defined by some set of criteria. These criteria organize the computerized enterprise-wide IMS environment into exclusive categories of applied organizations and information management systems.

CRITERIA OF SYSTEMS PLANNING

In practice, information management systems have been implemented according to the user needs in a very flexible manner. This manner develops an eclective system very often without knowing how the business environment will look when all its processes will be computerized.

The main difficulty lies in the selection of criteria that classify:

(1) applied systems within the organization

(2) levels of processing information

(3) information technology systems

Regarding the applied systems, the criterion of the system purpose recognizes the following systems in the organization:

(1) logistic system, which converts raw materials, parts, subassemblies or information into ready products,

(2) management system, which controls the logistic system

IMS applied in the logistic system will be called Logistic Support IMS and in the management system will be named Management Support IMS.

In each of these systems one must recognize the further division of applied systems in order to allocate alternations of IMS.

The criterion of the logistic substance recognizes the following types of logistic systems:

(1) production system processing material (e.g. car manufacturing)

(2) para-production system handling material-based subjects (e.g. body shop)

(3) info-based services processing information (e.g. engineering bureau, mass media)

(4) para-info-based services (e.g. library, telephone company, computer center)

As far as the levels of processing information are concerned, one can recognize the following management support processes which are based upon the criterion of the information contents:

(1) data processing (DP), which identifies states of applied systems and provides the measurement of the managerial actions

(2) information processing (IP), which is based upon data and produces directions for the managerial actions

(3) knowledge processing (KP), which is based upon facts and rules and creates the awareness for the managerial actions

(4) wisdom processing (WIP), which is based upon the different types of knowledge and supports the choice among the managerial decisions

SYSTEM MODELING OF THE ENTERPRISE-WIDE ENVIRONMENT

Among the Management Support IMS one can list the following implementations: Data Processing: (1) Record and Data Processing (RDP), (2) Data

Base System (DBS), (3) Local Area Network (LAN), (4) Electronic Mail (EM), and (5) Value Added Network (VAN); Information processing: (1) Management Control System (MCS), (2) Decision Support System (DSS), (3) Executive Information System (EIS), and (4) Professional Information System (PRS); Knowledge processing: (1) Intelligent Decision System (IDS), and (2) Professional Info System based on knowledge base (PRS/KB); Wisdom processing: (1) ROBOT (preprogrammed wisdom), (2) Expert System (EXS), and (3) CYBER (autonomous decision making).

Among the Logistics Support IMS there are such system implementations as: Production system: (1) Computer Aided Design (CAD), (2) Computer Aided Manufacturing (CAM), (3) High/Rise Storage (H/R), (4) Automatically Guided Vehicle (AGV), (5) Flexible Manufacturing Systems (FMS), (6) Numerical Control (NC), (7) Computer Integrated Manufacturing (CIM), and (8) Robotics, etc.; Para-production system: (1) Computer Aided Service (CAS), (2) Computer-Command-Control-Communication and Intelligence (C^4I), and (3) H/R, AGV, NC, Robotics, etc.; Info-based services: (1) Computer Aided Instruction (CAI), (2) Time Sharing System (TSS), (3) Computer Graphics (CG), and (4) CAD, etc.; Para-info-based services: (1) Information Center (ICT), (2) Development Center (DCT), (3) Computer Center (CCT), (4) Library (LIB), (5) Teleconferencing (TC), (6) Typesetting (TS), (7) Word processing (WP), (8) Desktop publishing (DTP), and (9) LAN, VAN, etc. The organized allocation of the IMS alternations in the enterprise-wide environment is shown in Figure 1.

Of course the examples of IMS implementations reflect the practice of the 80s and will be extended or replaced by the more modern solutions that will come along with the developments in information technology.

CONCLUSIONS AND RECOMMENDATIONS

The significance of the generic structure of the enterprise-wide IMS relies on the holistic view of the potential applications of IMS. The CIS student or IMS planner-designer-implementator and particularly general managers, the end users, will have a scientific tool for decision making about the scope and priorities of CIS projects.

It seems to be obvious that the holistic view of the computerized business environment should be covered in the CIS curriculum, particularly in the introductory classes, which very often determine the student's choice of major. At the MBA level this view is essential for the appropriate education of the future managers responsible for the whole business environment.

BIBLIOGRAPHY

Ackoff, R.L. and Emery, F.E. On Purposeful Systems, Aldin-Atherton, NY, 1972.

Beer, Stafford. Brain of the Firm, Allen Lane, 1972.

Bertahamnty, Ludvig von. General Systems Theory, George Braziller, NY, 1968.

Churchman, C. West. The System Approach, Delacote Press, NY, 1968.

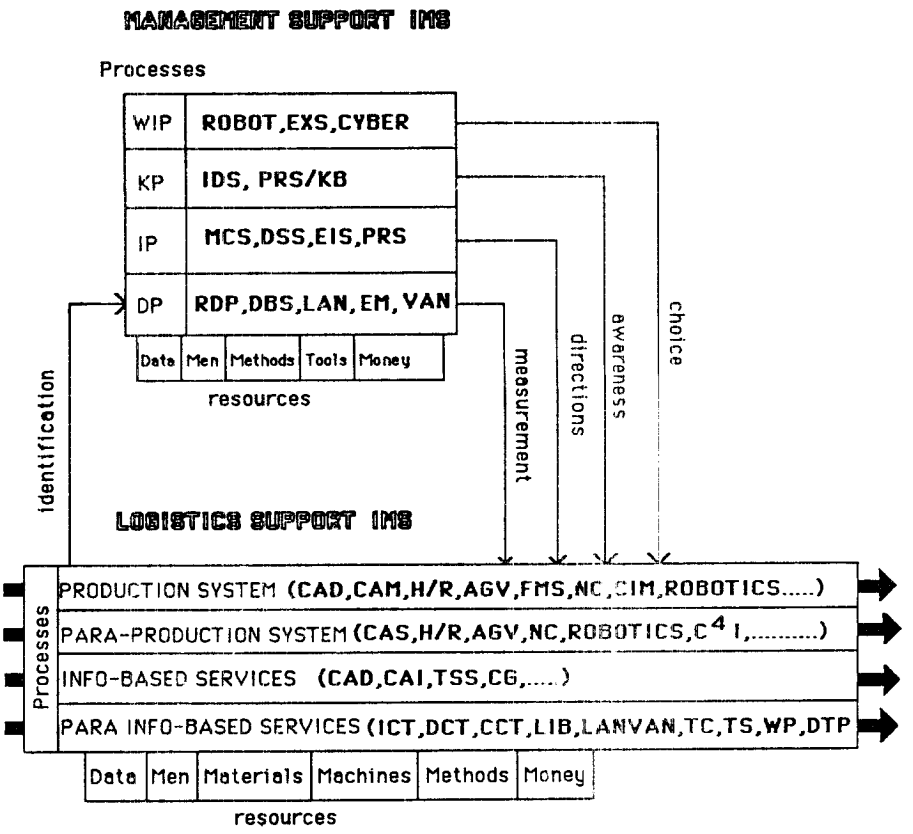
Klir, George J. An Approach to General Systems Theory, Van Norstand, 1969.

Parker, M.M. Enterprise-wide Information Management: Emerging Management Requirements, IBM, June 1984.

Targowski, Andrew. Informatics, PWE, Warsaw, 1980.

Figure 1 Enterprise-wide IM systems

The Targowski model



HUMAN FACTORS DESIGN IN INFORMATION SYSTEMS: CURRICULUM AND TEACHING ISSUES

JEAN B. GASEN
DEPARTMENT OF INFORMATION SYSTEMS
VIRGINIA COMMONWEALTH UNIVERSITY

ABSTRACT

The development and teaching of curricula related to human-computer interaction is still in its early stages within the field of information systems. However, with growing numbers of users from a wide variety of psychological, social and cultural backgrounds, it will be increasingly important to include the knowledge of user characteristics and the interaction process in the CIS curricula. The paper focusses on an outline of topics, issues and approaches important to the teaching of computer-human interaction. A description of one class project is used to illustrate the emphasis on student participation and hands-on evaluation as integral components of the curricula.

INTRODUCTION AND RATIONALE:

One body of knowledge which is receiving growing attention in the area of information systems research, development and teaching is the field of human-computer interaction. It is quite evident that there has been a tremendous increase in the number and diversity of users who are interacting directly with computers, from point-of-sale salespersons, to middle-level managers, to executive level CEO's. Despite such growth, we still know very little about what specific characteristics of users are important to study in order to design the most effective information systems.

The importance of assessing user needs has long been recognized in the field of information systems. However, the systematic study of user characteristics as a focal point for design efforts has only begun to take on significance in the past 15 years or so (7).

There are many areas of research and development for which we still do not have readily available answers. For example, how well do we understand exactly what cognitive processes an individual uses in interacting with a given system? To what extent are personality or cognitive style dimensions important in how an individual approaches a task, or learns a task most efficiently? How well can we specify a set of guidelines for the choice of one input device over another for different types of tasks and/or users? When is a color screen really helpful, and when does it become overkill to the user's eye? Is there an ideal response time to user input, or does it depend on the level of user's knowledge and familiarity with different tasks? What impact do changing levels of response time or display rate have on error rates, blood pressure, anxiety etc.? To what extent does our current knowledge of computer systems inhibit performance on other systems, or contribute to our reluctance to try more

effective or efficient systems in the first place? (5). These and other related issues are the increasing concern of individuals who study the process of human-computer interaction, and design models and procedures for incorporating such knowledge into the design of automated information system.

DEVELOPING CURRICULUM IN HUMAN-COMPUTER INTERACTION:

As Norman (3) notes, the field of human-computer interaction is very much pluralistic with respect to design approaches and philosophies. Similarly, because of the recent interest in teaching human-computer interaction curricula, there is very little information or consensus on what curricula should define the area, and what approaches are most effective in teaching human-computer interaction.

Interestingly, developing curriculum for a course in human-computer interaction shares some of the same problems as other curriculum areas within the field of information systems, but is also unique in other respects. It is similar in that the field is rapidly changing, and the published literature (articles and monographs) lags behind current developments in the field. On the other hand, the field of human-computer interaction is unique in that there are a very limited number of published books which could be considered appropriate as texts for a semester course. In speaking with other colleagues who have taken or taught courses in this field, I was struck by the difficulties shared in selecting a text which would survey the field as broadly as possible, presenting different theoretical approaches and pragmatic implications for design, and do so with a style appropriate to a teaching and learning environment. Those texts which are the most often cited include:

* Card, Moran, & Newell (1983) **The psychology of human-computer interaction** (1).

* Norman, & Draper Eds.(1986) **User centered system design** (2).

* Rubinstein & Hersh (1984) **The human factor: Designing computer systems for people** (4).

* Schneiderman, B. (1980) **Software psychology: Human factors in computer and information systems** (6).

* Schneiderman, B. (1987) **Designing the user interface: Strategies for effective human-computer interaction** (7).

For the course that I taught, I selected Norman and Draper (2), and Rubinstein and Hersh (4), supplemented by a set of readings from current periodicals. The course was organized around a series of topics, beginning with rationale and conceptual issues, followed by system development life cycle topics (analysis,design,development, and implementation). In addition, students were asked to complete a number of short projects which applied course material to topics which we were covering, and one larger project which was intended to integrate and apply concepts learned throughout the semester.

TEACHING HUMAN-COMPUTER INTERACTION:

By its very nature, this course lends itself to active student participation. All students have had some direct computer interaction prior to the course, whether with a

microcomputer, mainframe, or even an automatic teller machine. Therefore, the students themselves bring a wealth of experience which they can apply meaningfully to the course material. Class discussions often centered on the similarities and differences between student experiences which illustrate particularly effective or ineffective user interface designs. Many also brought work related experience, both as designers of systems or users of systems, to the course, and were able to apply the course knowledge back to their job directly.

In addition to didactic presentations facilitated by student discussions and examples, the course included both small and large projects to provide hands-on experience related to key concepts in the course. One example is described here for illustrative purposes:

NOVICE USER: ERROR EVALUATION PROJECT

Students were asked to evaluate a popular wordprocessing package as part of a small project requirement. They were asked to work in pairs and were given very little documentation on the software package (a short handout). They were given a two page printed document(an excerpt from a book on artificial intelligence) which included a number of basic word processing tasks, and were asked to type, save, and print this document. They were, in a very real sense, being asked to become "typical" novice users.

In order to address issues raised in class concerning error identification and correction, they were asked to keep track of any difficulties they encountered, how they attempted to resolve them (at the keystroke level), and what the system's responses to these attempts were. They were also required to write a written report on the strengths and weaknesses of the package, and evaluate it along a number of substantive dimensions relevant to several word processing tasks. Finally, they were asked to provide an affective evaluation, rating it along such dimensions as: stimulating-boring, enjoyable-no fun, friendly-unfriendly, frustrating-satisfying, etc.

This project was illuminating in a number of ways. First, by standardizing the assignment, it was possible to compare and contrast the kinds of problems they encountered, strengths and weaknesses, recommendations etc. through class discussion. Second, by completing the project requirement with another person, some of the anxiety and frustration typically encountered by first-time users working alone was mitigated (although the package was still overwhelmingly considered frustrating). Third, the assignment provided a number of good examples of gaps or breakdowns in the interaction process because of the focus on specific task difficulties and the identification of methods used to attempt to correct them. (For example, some problems were due to a lack of ability to articulate the syntax, such as how to execute a right flush command, or how to block and indent text. Others were due to the system's inability to reflect changes in the system after the user entered information, e.g. inability to see if page numbering was working until the page was printed out). Finally, the standardized evaluation at the end provided a quantitative estimate of how they viewed the experience, as

well as furthered discussion concerning the applicability of such methods as part of the design and evaluation process.

SUMMARY

The development and teaching of curricula related to human-computer interaction is still in its early stages within the field of information systems. However, with growing numbers of users from a wide variety of psychological, social, and cultural backgrounds, it will become increasingly important to include the knowledge of user characteristics and the interaction process in the CIS curricula. This paper is intended to highlight some of the important issues and approaches to teaching such curricula in order to improve the skills of data processing professionals of the future.

REFERENCES

1. Card, S. K., Moran, T.P., and Newell, A. **The psychology of human-computer interaction**, Lawrence Erlbaum & Associates, Hillsdale, NJ (1983).

2. Norman, D.A. & Draper, S.W. (Eds.) **User centered system design.** Lawrence Erlbaum & Associates, Hillsdale, NJ (1986).

3. Norman, D.A. Cognitive engineering principles in the design of human-computer interfaces. in G. Salvendy (Ed.) **Human-computer interaction**, Elsevier Science Publishers, B.V;, Amsterdam, The Netherlands (1984).

4. Rubinstein, R., & Hersh, H. **The human factor: Designing computer systems for people.** Digital Equipment Corporation, Burlington, MA. (1984).

5. Schneiderman, B. Seven minus two central issues in human-computer interaction. **Proceedings from 1986 SIGCHI.** April 1986, pp. 343-349.

6. Schneiderman, B. **Software psychology: Human factors in computer and information systems.** Little, Brown and Co., Boston, MA. (1980).

7. Schneiderman, B. **Designing the user interface.** Addison-Wesley, Reading, MA., (1987).

USING TECHNOLOGY IN TEACHING: INTELLIGENT TUTORING SYSTEMS

H. Ray Souder, Ph.D., CSP, University of Central Florida

ABSTRACT

Previously the production, storage, retrieval and reproduction of information designed for educational purposes has depended almost entirely on the written word. The arrival of the computer in the second half of this century however has altered this state of affairs irrevocably. In today's modern classroom, traditional texts are giving way to electronic media which are establishing the computer as a legitimate tool for the disseminating of quality education. The computer in turn has fostered a wide range of new educational tools. One of the more effective of these tools is the intelligent tutoring system. This paper discusses the nature of an intelligent tutoring system. Four primary issues are present: 1) the basis for these systems, 2) the components of a system, 3) the benefits, and 4) the costs of intelligent tutoring systems.

INTRODUCTION

Previously the production, storage, retrieval and reproduction of information designed for educational purposes has depended almost entirely on the written word. The arrival of the computer in the second half of this century however has altered this state of affairs irrevocably. In today's modern classroom, traditional texts are giving way to electronic media which are establishing the computer as a legitimate tool for the dispensing of quality education. The computer in turn has fostered a wide range of new educational tools. One of the more effective of these tools is the intelligent tutoring system (ITS).

THE BASIS OF INTELLIGENT TUTORING SYSTEMS

Early in the 1950's computer assisted instruction (CAI) began. Simple linear programs were developed. These programs were referred to as linear because of the step-by-step nature of the process used during execution. The final activity of this process was that of informing the student if they were correct in their responses to a give problem environment.

Throughout the 1960's and into the 1970's more sophisticated CAI were designed. Three major features evolved in these two decades. First, "branching" developed. This offered control over the material. The final result was the design and implementation of courseware authoring systems (CAS).

The next major feature was that of the computer system to generate teaching material within a given domain. For example, sentences could be "built" and modified, thus increasing the level of difficulty for the student.

Finally, teaching strategies were incorporated in these CAI systems. The interactive aspects were enhanced and the student's feedback improved considerably.

Late in the 1970's, several groups of professionals began working on intelligent tutoring systems (ITS). These systems are also referred to as intelligent computer assisted instruction (ICAI). An ITS is a computer program that simulates understanding of a given domain and can respond, direct, and lead a student through a problem-solving task and strategy.

ITS incorporates expertise from educators, cognitive psychologists, computer scientists, artificial intelligence techniques, computer technologies, and domain experts. Most ITS are developed utilizing production systems. A production system is a particle method of organizing knowledge. These are the same production systems used in expert system shells. Knowledge as used in this paper refers to: 1) facts, 2) rules, and 3) inferences. Facts are declarative truths, knowledge, about a given case. Rules are procedural methods, knowledge, on how to reason within a given domain. Inference is the control knowledge of how to carry out reasoning from a set of given facts and rules to come up with a solution/ conclusion.

COMPONENTS OF AN ITS

There is not a generally agreed upon model of an ITS. Most noted in the field is the work of John Anderson, C.F. Boyle, and Brain Reiser, all of Carnegie-Mellon. These researchers suggest that there are four primary components: 1) domain expert, 2) bug catalogue, 3) tutoring knowledge, and 4) user interface.

The domain expert is a module capable of solving problem within the domain. The bug catalogue is a library of misconceptions and errors of the domain. When discussing tutoring knowledge we are referring to the strategy(ies) to teach a domain's knowledge. Finally, the user interface is a program which allows interaction between the tutoring system and the student/user.

It must be reiterated here that there other work in the area of the design/ components of an ITS. For those readers interested, at the end of this manuscript there is a list of suggested readings that can assist you.

BENEFITS OF ITS

The major benefits of an ITS are: 1) flexibility in selecting multi-teaching strategies within a domain, 2) the user/student can progress at their own rate, 3) immediacy of feedback and guidance, 4) the ITS can detect and record how new cognitive skills are acquired, and 5) the faculty has more time to work with individual students who have special needs.

As we review the above identified benefits, it is obvious that the ITS provides the faculty member with a multitude of additional teaching aids. With these "helpers" the rigor in the classroom can be reinstated. In addition, productivity and performance can be stressed and measured accurately. The subjectively in these areas can be eliminated. Finally, accountability of faculty can be measured and evaluated. This issue, accountability, usually brings a tumultuous uproar from many of our colleagues. Therefore, further discussion of this issue will be left to another paper.

THE COSTS OF AN ITS

There are four primary costs we can identify of an ITS. These four costs are: 1) new capital funding, 2) training faculty, 3) development time, and 4) establishing "correct" paradigms for acquiring knowledge within a domain. Any of these costs are enough to stop a preliminary cost analysis in most academic institutions. Let us look briefly at each of these.

New capital funding would be needed to purchase the computer hardware and software. These work stations would cost about $10,000 to $15,000 each. These costs for a work station is down from several hundreds of thousands of dollars just five years ago. It is important to note that each student in

a classroom environment would need such a work station. Therefore, for example, a class of 25 would need from $250,000 to $375,000 to outfit the classroom.

The cost to train a faculty member is very difficult to determine. It is safe to say that the cost of an average professional development program is about $ 800 and at least that much for travel and lodging. Any training of a faculty member could be in the $1,000 to $2,000 range per course. Development time of an ITS is one item where accurate estimates exist. It is estimated that for each one hour of classroom usage time there is 200 hours of development time needed. The algorithm to calculate the cost of an ITS is very simple, i.e., 200 times the faculty hourly rate times the number of hours of usage of the ITS. Of course, there are those who suggest that the faculty hourly rate is normally zero for a calculation of such a cost. We all know, that the price is paid one place or the other.

Much like the development cost of an ITS, selecting and or designing paradigms most appropriate for a given domain is a labor intense work task. Also like the development time, the task usually will fall to the faculty or faculty group working on the project. An important issue here is that software development needs to receive academic credit for scholarly efforts for tenure and promotion. To date, most institutions do not recognize it as such.

CONCLUSIONS

The paper has discussed the nature of an intelligent tutoring systems (ITS). The major points presented included:

1. Computer assisted instruction has been developing for over 30 years.

2. An intelligent tutoring system is a computer that simulates understanding of a given domain and can respond, direct, and lead a student through a problem-solving task and strategy.

3. Most ITS utilize a production system in the development of the knowledge base.

4. The four primary components of an ITS are: 1) domain expert, 2) bug catalogue, 3) tutoring knowledge, and 4) user interface.

5. The major benefits of an ITS are: 1) multi-teaching strategies are available, 2) student progresses at their own rate, 3) immediacy of feedback and guidance, 4) can detect and record how new cognitive skills are acquired, and 5) faculty has more time to work with student who have "special" needs.

6. The major costs of an ITS are: 1) new capital funding needs, 2) training faculty, 3) development time of the ITS, and 4) establishing "correct" paradigms within a given domain.

7. It would appear that at this time ITS are too costly for most academic institutions.

8. Of course, the underlying linchpin to utilizing technology in an academic environment is one that cannot be avoided. Can we as professionals afford not to use tools that offer productivity enhancements for faculty and student alike.

SUGGESTED READINGS

1. Anderson, J. R., "Skill Acquisition: Compilation of Weak Method Problem Solutions," Carnegie-Mellon University, 1985.

2. Anderson, J. R., The Architecture of Cognition, Harvard University Press, 1983.

3. Anderson, J. R., Boyle, C. F., and Reiner, B. J., "Intelligent Tutoring Systems," Science, 1985.

4. Anderson, J. R., et al., "Learning to Program in LISP," Cognitive Science, 1984.

5. Bloom, B. S., "The Two Sigma Problem: The Search For Group Instruction as Effective as One-On-One Tutoring," Education Researcher, 1984.

6. Bramer, M., Editor, Research and Development in Expert Systems, Cambridge University Press, 1985.

7. O'Shea, T. and Self, J., Learning and Teaching with Computers, Haravester Press, 1983.

8. Osgood, D., "A Computer on Every Desk: A Survey of Personal Computers in American University," Byte, 1984.

9. Papert, S., Mindstorms, Children, Computers, and Powerful Ideas, Haravester Press, 1980.

10. Sleeman, D. and Brown, J., Editors, Intelligent Tutoring Systems, Academic Press, 1982.

11. Yazani, M., Editor, New Horizons in Educational Computing, Wiley, 1984.

A TUTORIAL ON COURSE AUTHORING SYSTEMS FOR INFORMATION SYSTEMS COURSES

Thomas C. Richards
College of Business
North Texas State University

ABSTRACT

North Texas State University has recently purchased an authoring system marketed by HyperGraphics of Dallas Texas. This is an IBM PC compatible course authoring system which allows the user to create complete courses for both self-paced and instructor-led instructional environments. The presentation quality of the materials make this system the perfect delivery vehicle for instruction in information systems which requires a medium which can produce the maximum in student interest and retention. The attributes of this system are discussed in this paper.

INTRODUCTION

As technology continues to become more available, we will continue to find ways to utilize it to improve the educational process. Instructional television and the computer are two technological advances which are impacting college level instruction. In particular the use of computer assisted instruction (CAI) is not new in information systems but with the advent of the microcomputer it has become much more feasible to use in an instructional environment.

Computer assisted instruction is a cost-effective training method which is finding use in the corporate training environment. Can it also find use in public education? There are a number of reasons why we might find this to be true. The learner can set his or her own personal assimilation rate which increases the confidence level of the learners by allowing them to absorb material at their own pace and by eliminating embarrassment over errors. It can be used for testing (both pre and post), to supplement lectures and for drill and practice.

There are a large variety of software packages available to the instructor for authoring or writing CAI lessons for the learner. A recent issue of Classroom Computer Learning (5) lists some fifty vendors of these packages. It is important that the selection of these tools for the development of instructional materials be carefully done since they offer a wide variety of capabilities and some products are very limited.

Authoring tools fall into two broad categories: authoring systems and authoring languages. Authoring systems are computer programs which require no programming ability on the part of the user or author. These systems are often menu-driven with the actual computer code being generated by the authoring system based upon the responses to the menu, textual input and pictorial input. On the other hand an authoring language

is very similar to computer programming and requires the author to use a set of commands. These commands then make up the program which generate the lesson. An authoring language is more versatile than an authoring system and has many more options.

As might be expected the authoring tools on the market vary in price and complexity. Retail cost vary from $100.00 to $2000.00 for microcomputer based authoring systems. At a minimum they allow the creation of programs with multiple choice and true/false questions, branching and tabulations of student responses. The more expenses systems include graphic capabilities, dynamic display of information, string input, string evaluation, multiple windows and some movement of information on the screen.

EVALUATION OF AUTHORING SYSTEMS

There are a number of criteria by which authoring systems can be compared or evaluated. These considerations fall into five broad categories which include how the lesson content is created (text, graphic and sound editors), lesson definition (structure of lessons, display strategies, student response processing and branching sequences), course management (student management, scoring data, summary information on student performance) and authoring environment (hardware requirements, training requirements and ease of use).

One author who compares a number of authoring packages (1) looks at the following parameters: documentation for the authoring system, methods of displaying textual material, branching and sequencing strategies, the ease of text editing, graphic capabilities, sound generation, animation capabilities, the ability of the author to test his lesson as he builds it, student record keeping, answer processing, feedback provided to the student and the ability of the authoring system to store variable information related to the student. These categories will be used in the following discussion.

AN EXAMINATION OF A PARTICULAR AUTHORING PACKAGE

North Texas State University has recently purchased an authoring system marketed by HyperGraphics of Dallas Texas. This is an IBM PC compatible course authoring system which allows the user to create complete courses for both self-paced and instructor-led instruction environments. The presentation quality of the materials make this system a viable delivery vehicle for instruction in information systems which requires a medium which can produce the maximum in student interest and retention.

By using this system the author of a course may present material with text, graphic and animation. The system also provides highly functional utilities for combining material from different presentations, rearranging course strategy and analyzing courseware page content and structure. There is also a "capture" utility for importing graphic screens from other software as well as from providing pre-drawn "clip art". Features of the system include template question construction, response-sensitive branching and multiple branching strategies. Question types allowed by the system are diverse and flexible. The system also remembers where individuals are within the lesson materials, giving the learner the option of starting where he or she left off or at the beginning of a new course or lesson.

DOCUMENTATION

The documentation for tbtAUTHOR comes in a three ring binder and includes the usual information including setup, training, reference pages and a very brief index. The manual is rather sparse as often the case with manuals. It does have an rather extensive tutorial covering almost all features of the system. The reference section could be expanded with more specific details of the system. The index to the manual is poorly put together and needs many

more items in it to be of use to the user.

TEXT DISPLAY

tbtAUTHOR provides seven text heights and seven text widths. The text can be boxed with variations in both text color and the background in the box. Only one font is provided for the letters.

BRANCHING AND SEQUENCING

Lessons are made up of pages in tbtAUTHOR. These pages can be accessed with a great variety of options. It allows the author to branch to as many as ten other pages from any page in a lesson. The RETURN function allows the return to the last page from which a branch was executed.

TEXT EDITING

Only complete lines of textual material can be changed. The system does not allow the insertion, alteration or deletion of individual words or characters. Since most of the changes made in authoring of lessons occurs at the character or word level this is a deficiency of this system.

GRAPHICS

tbtAUTHOR makes good use of the graphic capability of the IBM PC which is equipped with a graphics board. It comes equipped with a large selection of predrawn figures plus it allows the author to draw circles, boxes, lines and do free hand drawing.

SOUND

The version of tbtAUTHOR which was examined did not have sound capabilities.

ANIMATION

This system contains a move command which allows for the movement of segments of the screen about the screen (or even off the screen). This function is easy to use and a very effective presentation method.

TESTING A LESSON

Since a CAI author will be interested in viewing his lesson or parts of it as it is under development, the ease with which he can test a lesson out before it is completed is an important attribute of any authoring system. The time required to move from the author mode into the test mode and back is critical in the use of an authoring system. The tbtAUTHOR systems allows the testing of most of the functions of the lesson while still in the author mode. To move into the presentation mode requires logging out of the authoring system and into the presentation system. This requires perhaps twenty key scores and about thirty seconds.

STUDENT RECORDS

There is some flexibility of the amount and type of information which is recorded regarding each student. This information includes the learner's name, weighted correct answers, weighted possible score and number of wrong answers recorded. The system also can track a learner and record which lesson he or she has completed. No information is recorded regarding the selection of responses to each question. This type of information would be valuable for item analysis and calculation of discrimination and difficulty indices for each lesson page.

ANSWER PROCESSING

An authoring system should be able to determine if a learner's response to a short answer question is correct. Even if the correct answer is embedded in another word or phase in the student's answer. This authoring system allows wildcard characters as well as several options for handling spaces in a student's response when comparing this response with the anticipated or correct response. The maximum length of an anticipated response is forty characters.

FEEDBACK

Feedback is defined as corrective and reinforcing information provided to the student based on his correct or incorrect response to a question. Each response to a question whether it be a multiple choice question or a short answer can have related to it a branching sequence. This sequence can provide any type of feedback the author may wish to provide to the student.

VARIABLES

In a CAI package variables consist of information which is stored by the computer during the lesson and which can

be accessed at a later point in the lesson. tbtAUTHOR can store and the author can manipulate the following variables during the authoring of a lesson: the student's name, the time, the date, the student's last response to a question, the count of correct answers and unanticipated answers. The usefulness of these variable has not been determined by the investigator at this time.

PROBLEMS WITH CAI

The expectation that CAI would eliminate traditional classroom teaching has not been realized and probably never will be. First of all if you are using CAI tutorial packages which teach people how to use an applications program such as LOTUS these applications programs are becoming more user friendly and the software designer can add "concurrent" tutorial applications in the packages. This is often done via help screens and keys. For example McGraw-Hill offers concurrent training packages for DOS, Lotus and dBASE III. Cdex Corp has a package which takes the user from the Lotus spreadsheet into a training program. This is of course more convenient for users, who can get help at the moment they need it without engaging in a CAI course. This technique provides immediate feedback to the learner since he can try out what he has learned as soon as he learns it.

The issues related to CAI packages which teach non-computer subject such as management skills, accounting concepts, etc. are being attacked from a different standpoint. We don't know much about the tolerance of a learner to sit in front of a terminal and work through a CAI program. Researchers are beginning to suspect that there are some combinations of CAI and traditional instruction that work best for the learner. Some researchers believe that CAI often works best not as a replacement for classroom instruction but as a complement to it. They see the group-interaction and socialization as an important part of learning. They also see the contacts made and relationships formed among students in class as important as the subject matter of the course its self.

Another problem is the cost of developing CAI materials. Estimates run from 85 hours to 300 hours for the preparation of one hour of a CAI lesson (2). The average college professor does not have this sort of time and unless funding is available can not hire graduate students to perform the work. With the emphasis on research and publications few institutions would classify the development of courseware as counting as a publication.

Most CAI courseware simple represent a form of scrambled textbook where a student responds to a question and then based on the response the next frame or material is presented to the learner. There is some work being performed to move CAI courseware from the dumb to the intelligent domain which will be more response to the needs of individual learners (4). Until these system are fully developed and the authoring systems to support them are programmed, we will not experience the full benefits of the use of artificial intelligence in CAI (3).

REFERENCES

1. Burger, Michael L.,"Authoring Languages/Systems Comparisons," **AEDS Journal**,(19,3),Winter/Spring 1986.
2. Lee, Chris and Ron Zemke, "How Long Does it Take?," **Training**, (24,6),June 1987.
3. Wheeler, David L., "Artificial Intelligence Researchers Develop Electronic 'Tutors' to Aid Learning Process," **Chronicle of Higher Education**,(45,34), May 20, 1987.
4. Yang, Jung-Shing, "Individualizing Instruction Through Intellegent Computer Assisted Instruction: A perspective," **Educational Technology**, (27,3), March 1987.
5. "Classroom Computer Learning 1987-88 Buyers' Directory," **Classroom Computer Learning**, (7,4), January 1987.

INSTRUCTIONAL PLANNING FOR COMPUTER ASSISTED INSTRUCTION

by

Y. Sankar (Ph.D. Johns Hopkins University)
Y. Shafai (Ph.D. Michigan State University)
Professor of Management
School of Business Administration
Dalhousie University, Halifax, Canada

ABSTRACT

It is critical for any Computer Assisted Instructional (CAI) module, tutorial, drill, inquiry, or simulation, that the goals of the learning task be identified by the development of specific instructional objectives to meet these goals. To develop instructional objectives it is necessary to analyze the criterion task into elementary behavioral components. The paper is organized around a framework for setting objectives. These instructional objectives are crucial because they determine the orientation, level of cognitive complexity, and evaluation criteria for each CAI module. The design of instructional objectives is also critical because they identify cognitive skills that are more complex than the simple retrieval of information generally associated with CAI modules and they play a major role in instructional planning.

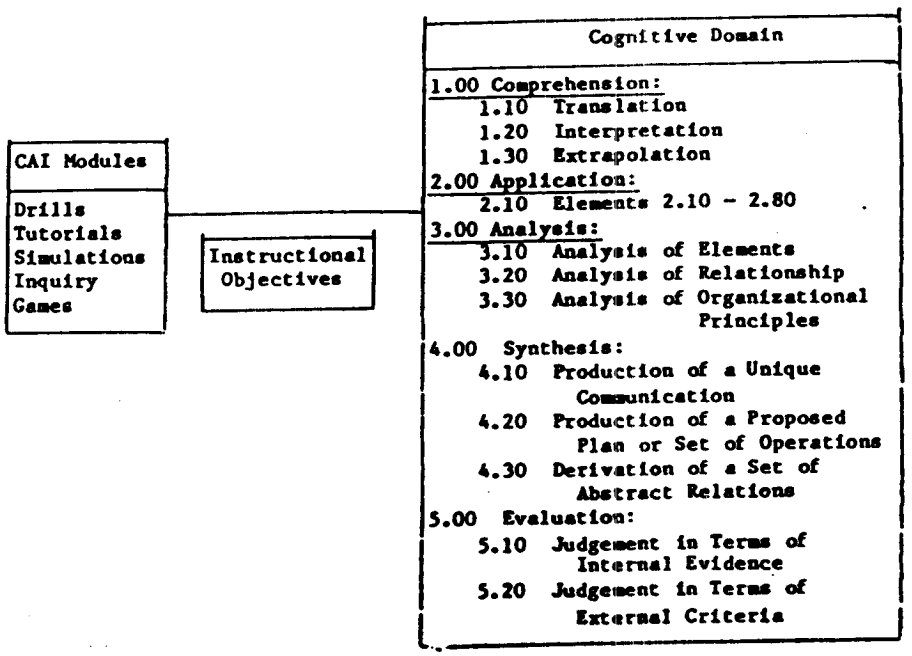
The Criticality of Instructional Planning

Developing courseware is a significant undertaking and if it is to be of high quality according to Chambers et.al. it requires knowledge of the subject matter, learning theory and instructional strategies, principles of motivation, microcomputer design technology, and computer programming. (Chambers et.al. 1983)[1] The manner in which a design activity is approached is our single most effective predictor as to whether or not the activity will actually result in CAI material of some degree of excelllence. Too often CAI is simply a method for the exposition of content, a list of isolated topics or bits of information in different modes or configurations. Sometimes the retrieval of information is the crucial learning objective of the drill or tutorial. The art of retrieving bits of information while necessary is the lowest level of cognitive performance. The processing of these bits of information and the conversion of the information into decision outcomes is a more crucial cognitive strategy. To design meaningful CAI programs it is imperative that the learning objectives of the CAI module be articulated. Godfrey and Sterling (1982)[2] reinforce the relevance of objectives setting. "To date, our own experience leads us to say that CAL (computer aided learning) is only really useful (and courseware is only quickly created and tested) when the author can specify the set of objectives of the course in advance and then define these objectives in terms of a clear list of specific rules".

It is important for any CAI module, tutorial, drill or simulation that goals of the learning task be identified followed by the development of specific instructional objectives to meet these goals. Such objectives must be

for the systematic and precise categorization of learning objectives is the Taxonomy of Educational Objectives edited by Bloom et. al. (1971)[5] It helps us The Taxonomy of Educational Objectives (main headings set out in Graphic I, put objectives into two categories: those of a conceptual nature into the cognitive domain and those of an emotional nature into the affective domain. It is not merely a classification according to type: it is a taxonomy, which implies a definite set of relationships. In this case, it is a hierarchical classification. Objectives at the first level are subsumed in the second; the first and second in third, and so on, so that in the cognitive domain the sixth level, that of evaluation, is a category superordinate to the remainder. Thus, in the cognitive field, the higher one moves in the hierarchy, the more complex the intellectual skills needed to cope with concepts at that level. The value of this taxonomy is that it provides a key by which one can assess the level of functioning of any evaluative device. The logic of a taxonomic analysis suggests that, in general, abilities tested early in a program will be subordinate to those tested at the end of the program and will be subsumed by them.

Figure I
CAI and Instructional Planning



Learning Objectives and Instructional Design

Learning objectives are in most CAI training programs, stated so that only the coverage of the content is explicit, and it is not clear whether this content is to be memorized, thought about, or acted upon to produce a change of attitudes. In other words, the expected behavior is not specified. At the other extreme are generalized statements of behavior, such as developing an ability to think logically or to express oneself clearly, without any indication of the kind of content in which this behavior is to apply. The most useful and clearest statements of objectives are those which specify both the kind of behavior reaction that is expected and the content to which it applies. If the behavior denotes knowing or remembering, the statement of objectives should also indicate what is to be known or remembered. If the statement specifies an

stipulated in concrete behavioral terms. To develop instructional objectives, it is necessary to analyze the criterion task into elementarycomponents and to determine their orientation. The skill level of the students must then be assessed and programs designed to teach the skills. This paper will provide a framework which can be used for setting instructional objectives for CAI modules.

Objectives of Instructional Planning

Instructional objectives are critical to the design of courseware because of the following functions that they perform. Instructional objectives (1) provide goal focus for CAI programs, (2) inform learner of lesson objective (3) clarify expectancies (4) provide performance feedback (5) serve as an instrument for allocation of time and cognitive effort (6) guide sequence of learning (7) provide informative feedback (8) enhance learning transfer (9) provide criteria for evaluation of CAI programs (10) ensure integration of CAI modules (11) provide for visual orientation of the learning task (12) ensures that various levels of the hierarchy of educational objectives are programmed in the CAI modules (13) enhance motivation because of goal directed behavior of the modules. and (14) provide a rationale for the instructional design effort. Sankar (1987)[3]

Instructional Design and Learning:

An instructional plan is simply a strategy for managing the conditions of learning. That is, when the instructional plan is viewed as a plan for learning and not merely as a plan for listing bits of information re the sequence of content, additional considerations emerge regarding the essentials of learning or the "psychological order" of learning. Neither the ideas nor the sample of content for each idea as yet represent a teachable content: a sequence and a logical organization is only partially established. The content needs to be arranged so that the dimensions of inquiry are in a sequential order according to a feasible learning sequence. That is, the instructional plan or learning program is a method of teaching thinking; a pathway for developing generalizations inductively and for applying them: a way for establishing relationships between ideas and facts; a model for processing and converting information though inductive-deductive inquiry; a medium for the perception of complex interaction effects in a phenomenon; and a framework for evaluating a learner's capacity to interpret, translate, apply, analyse, synthesize and evaluate the bits of information. The design of a learning program then must be based on a set of strategies that facilitate the effective management of the conditions of learning. (Gagne, 1982)[4]

Formulating Learning Objectives for CAI

A systematic approach to the formulation and organization of objectives is necessary in order for objectives to serve their functions. There must be a rational basis for the conception of the outcomes of learning and for the grouping and classification of objectives in any computer based learning design.

A major system of classification in the field of education that is helpful for the systematic and precise categorization of learning objectives is the Taxonomy of Educational Objectives edited by Bloom et. al. (1971)[5] It helps us

attitude, then it should also state what the attitude is about. This principle of clarity and specificity suggests the need to analyze complex behaviors into their particular components, so that precise behaviors intended are clearly perceptible and so that the qualities expected of them are fully understood. It is necessary, for example, to break down such objectives as "clear thinking" into the ability to make inferences from specific facts, to apply principles, to use logical processes in detecting assumptions, and to generalize before this objective can guide in the formulation of decisions regarding type of teaching emphasis, learning experiences or activities and evaluation. The value of the taxonomy is that it provides the framework for clarifying and specifying the learning objectives of a program. (Sankar, 1978)

Some Learning Objectives Errors:

The learning objectives associated with most programs are invariably vague, ambiguous, and contradictory. For example, such learning objectives as the student must "know" the concept probability or the simplex method, "appreciate" he utility of the concept, binomial probability distribution, be "aware" of the interaction effects among a set of variables in analytical models, or show clear thinking in his conceptualization or application of a simulation approach are all vague, ambiguous and are of minimal practical value in evaluation. How, for example, do we discriminate between a student who shows an awareness and one who doesn't, what precisely do we mean by "to know", does it mean that the student should be able to define, interpret, translate, apply, analyze, synthesize, or evaluate the concept or variable? With reference to clear thinking, do we mean that the student is expected to be able to make inferences from specific facts, to apply principles, to use logical processes in detecting assumptions, to generalize, to discriminate and so on? Objectives as expressed in these learning programs are impossible to evaluate.

Summary

The identification and definition of performance objectives is an important step in the design of instruction. Objectives serve as guidelines for developing the instruction, and for designing measures of student performance to determine whether the CAI module objectives have been reached.

References

1. **Chambers, J. and Sprecher, J., Computer Assisted Instruction: Its Use in the Classroom, Prentice Hall, Englewoods Cliffs, New Jersey, 1983.**
2. **Godfrey, D. and Sterling, S. The Elements of CAL, Press Porcepic Ltd., London, 1982.**
3. **Sankar, Y. A Framework for Designing Effective Training Programs, Academy of Management National Conference, Chicago, 1985.**
4. **Sankar, J. A Conceptual Model for Evaluating Training Programs, Management International Review, 1978.**
5. **Gagne, R. Developments in Learning Psychology Implications for Instructional Design; and Effects of Computer Technology on Instructional Design and Development. Educational Technology, 1982.**
6. **Bloom, B., Hastings, J. and Madaus, G. Handbook on Formative and Summative Evaluation of Student Learning, McGraw Hill, London, 1971.**

Computer Education in the Schools K through 8: Should the DPMA be Involved?

Gregg Brownell
Bowling Green State University
Bowling Green, Ohio

ABSTRACT

Computer education in the schools, grades K through 8, is a reality. The topics covered at this level in a computer education curriculum relate directly to goals of the DPMA and to the expertise of its members. Accurate, up-to-date coverage of such topics as programming, hardware concepts, computer applications, social implications of computer use, and the history of computing is essential. This is especially true at young ages when children's attitudes about computers are being formed. There is an opportunity for leadership available to the DPMA in the areas of curriculum design, implementation, and teacher training.

INTRODUCTION

As a major force in the implementation of the Information Age, the DPMA has formulated and disseminated materials related to Computer Information Systems education at both the high school and college levels. Over the last decade, computers have increasingly become both an object and a source of instruction in many schools from kindergarten through eighth grade (in addition to high school). Many of the topics covered in a K through 8 computer education curriculum relate to the concerns of the DPMA in general and the Special Interest Group for Education in particular. Due to the need for quality computer education programs at all levels and because of the DPMA's unique position in both the business and education worlds, some effort should be made by the DPMA to aid in the definition of what computer education should be at the K through 8 level.

COMPUTER LITERACY

The general public has grown increasingly more aware of what computers are and how they are affecting our lives. It is obvious to most people that some knowledge about computers is necessary for success in the Information Age. This need has led to a societal concern for schools at all levels to provide students with the necessary skills and information to become knowledgeable about computers. The term computer literacy is often used to describe this knowledge. Unfortunately, the term means different things to different people. As Bitter (1984) points out,

> Although definitions of computer literacy vary, the term generally refers to a fundamental knowledge of computers, their uses and limitations, how they work, and their impact upon society (2).

Brownell, in his recent book Computers and Teaching (1987), offers the following definition.

> In General, one may say that computer literacy is a state in which an individual has an understanding of what a computer is and how it works and is comfortable and effective in making a computer accomplish a necessary task. In reality, however, a good definition of computer literacy will be specific to the situation in which the computer-literate individual must function. A certified public accountant, a Hollywood screenwriter, and a secretary will all need different skills and information to be comfortable and effective in their interaction with computers: each will need something different to become computer literate (4).

A more specific definition of computer literacy for students is then presented.

> Knowledge
> Students must be:
> * familiar with the components of a computer system (hardware and software) and how they work and interact
> * informed about the history of computing
> * aware of the current and projected uses of computers in society and the possible implications of those uses
> * knowledgeable about job opportunities associated with computers
>
> Performance
> Students must be able to:
> * use the computer for instructional purposes by using computer-assisted instruction software of both the tool and tutor type, both with teacher direction and, when presented with appropriate documentation, alone
> * write simple programs in two computer languages
> * engage in problem solving by breaking a complex problem into modules, generating a solution for each module, and combining the solutions to solve the problem (5)

It is because of the need to prepare citizens who will function in a world pervaded by computers that educators have been struggling with the concept of computer literacy. Where computer curricula exist, they are often the result of educators within a school district arriving at a local definition of computer literacy. This definition of computer literacy is then used as a basis for generating a computer curriculum that will produce an individual who is computer literate.

THE COMPUTER CURRICULUM

To date, most computer curricula have several components and begin as early as kindergarten. Students learn about the history of computing and the different components of a computer system as well as the various applications of computers in society. The social implications of computer use including questions of ethics, equity of resource availability in society (hardware, software and information) and present and future job opportunities are addressed. Instruction in programming as well as instruction about and experiences with productivity packages (word processors, data base management systems and spreadsheets) are included. Also covered are experiences using the computer as a source of instruction, i.e., for drill and practice exercises, simulations, and tutorials across a range of content areas such as math, reading, language arts and social studies.

Many of the topics in a K through 8 computer education curriculum relate directly to the concerns of the DPMA. Consider the following four points included in the list of objectives of the DPMA's Special Interest Group for Education.

* Advance the interest of members and society at large concerning

computer-related education.

* Increase member and public awareness of the impact of information processing and the computer on society, stressing members' responsibilities to employers, students and society.
* Foster a better understanding of the vital business role of data processing in society and of the proper relationship between data processing and education.
* Provide suggestions and advice to the DPMA governing body regarding activities in the area of education.

These objectives and the concerns of a K through 8 computer education curriculum show a certain commonality. This is even more obvious when the content of such a curriculum is related to the expertise offered by members of DPMA. It is important that early introduction of computer topics be accurate and present up-to-date information. Children at the K through 6 level, especially, are very impressionable. It is likely that their first introduction to computers will have a lasting effect on how they view and use this potent tool throughout their lives. There are a number of questions to be addressed with relevance to the DPMA's role. If programming is to be taught, what language(s) should be introduced? Are modern programming methods (structured programming, structured design) introduced? What view of the Information Age is presented? How is the role of the computer in society defined? How are ethical issues surrounding computer use covered? Overall, what topics should be included in such a curriculum? What mechanism allows for the revision of the curriculum to keep it up-to-date? What about teacher training? Are teachers well prepared to cover a computer education curriculum? If teachers are not adequately trained and the curriculum is not kept up-to-date, the possibility exists that students may receive inaccurate instruction which will have to be unlearned at the high school or college level. This is a waste of the instructor's time and a disservice to the student.

THE ROLE OF DPMA

There are several areas where the DPMA can make a contribution.

1. Develop a definition of computer literacy for students, grades K through 8.
2. Design a curriculum for computer education grades K through 8. Such a curriculum would relate directly to the definition of computer literacy for students grades K through 8.
3. Assist in developing materials for implementation of the K through 8 curriculum.
4. Become active in teacher training. Such activity could include development of a set of minimum competencies regarding computers and computer education for K through 8 teachers. Also, a set of competencies for K through 8 teachers with a specialty in computer education and/or competencies for a computer education coordinator might be developed.
5. Assist in the implementation of the proposed curriculum by working with interested school districts and/or state departments of education to provide consultant services.

CONCLUSION

Computer education in the early and middle grades is a reality. The topics addressed in a computer education curriculum and the concerns of the DPMA are closely related. There is an opportunity for the DPMA to make a positive, essential contribution to the way young children come into contact with computers and computer concepts. Leadership in the areas of curriculum development and implementation as well as teacher training would mirror the DPMA's unique role as an organization whose members are practical users and

controllers of technology as well as professionals concerned with education and the perceived role of the computer in society.

REFERENCES/FURTHER READING

1. Becker, Henry Jay. "Using Computers for Instruction." Byte 12, no. 2 (1987):149-162.

2. Bitter, Gary and Ruth A. Camuse. Using a Microcomputer in the Classroom. Reston, Va.: Reston Publishing Co., 1984, p.21.

3. Brady, Holly. "Reader's Survey Results: What is Computer Literacy?." Classroom Computer Learning 6, no. 6 (1986):53.

4. Brownell, Gregg. Computers and Teaching. St. Paul, Mn.: West Publishing Co., 1987, p 182.

5. Brownell, Gregg, p. 186.

6. Hunter, Beverly, Donald Dearborn and Bruce Snyder. "Computer Literacy in the K-8 Curriculum." Phi Delta Kappan 65, no.2 (1983):115-118.

7. Judd, Wallace. "A Teacher's Place in the Computer Curriculum." Phi Delta Kappan 65, no 2 (1983): 120-122.

8. Kinzer, Charles K., Robert D. Sherwood and John D. Bransford. Computer Strategies for Education. Columbus, Ohio: Merrill Publishing Co., 1986.

9. Westley, Joan. "How Texas Made History with the New Literacy Texts." Classroom Computer Learning 6, no. 5 (1986):50-52.

A UNIVERSITY COMPUTER SCIENCE COURSE FOR ADVANCED HIGH SCHOOL STUDENTS

Michelle M. Pfister
Robert P. Burton
Douglas M. Campbell
Larry C. Christensen

Brigham Young University
Provo, Utah 84602

ABSTRACT

An Advanced High School Studies Program (AHSSP) is described and analyzed. The Program teaches Pascal and structured design to outstanding high school students in a university setting and permits them to explore Computer Science as a prospective academic major.

Beginning in 1983, the AHSSP has been offered each summer by the Computer Science Department at Brigham Young University. The Program is the equivalent of a two-hour first course for prospective Computer Science majors. CS1 guidelines are followed. It is a rigorous, two week, full time Program.

Participants in the Program are nominated by high school teachers, administrators and PTA Presidents. Final participant selection is made with close attention to high school courses, grades and PSAT or ACT scores. Participants receive full academic scholarships and two hours of university credit.

Attracting bright students to the Department and University is a primary objective. Participants rank the experience at 9.3 on a scale from 0 to 10. The completion rate is 95%. At least 28% subsequently enroll at Brigham Young University.

HISTORY

The Advanced High School Studies Program (AHSSP) was offered first during the Summer of 1983. The number of participants has ranged from 186 in 1983 to 55 in 1985 (see Figure 1).

YEAR	NOMINEES	ENROLLEES(%)	PARTICIPANTS(%)	GRADUATES(%)
1983	n/a	202(n/a)	186(92%)	172(92%)
1984	210	173(82%)	150(87%)	142(95%)
1985	79	57(72%)	55(96%)	55(100%)
1986	90	77(86%)	75(97%)	73(97%)
Total	n/a	509(n/a)	466(92%)	442(95%)

Figure 1

PARTICIPATION

PARTICIPANTS

Participants are outstanding Utah high school students, between the junior and senior years. The Program is scheduled each year prior to the selection of a college or university by the participants. Each nominee needs to complete second-year algebra prior to the start of the Program. He or she must be a junior in the top 10% of the class. No prior computing experience is necessary.

NOMINATORS

Nominators are teachers and administrators affiliated with high schools in the State of Utah. The names of high schools and principals are obtained from the Utah High School Activities Association Handbook. In early September, letters containing information about the Program are sent to each principal, Computer Science teacher, head counselor and PTA President, inviting each to become a nominator. A response form is included which needs to be returned prior to January. In early January, information and forms are sent to the responders, soliciting nominations. In April, a follow-up letter is sent if no nominations have been received from a prospective nominator.

Communication with potential nominators has increased since the Program began. In 1983, nomination packets were sent only to high school principals. Because of a decreased response in 1985, nomination packets, preceded by advanced notices and followed by reminders, were sent to principals, Computer Science teachers, head counselors and PTA Presidents in 1986 to ensure an adequate number of nominees.

SELECTING NOMINEES/PREDICTING SUCCESS

Each nominee is asked to provide: a current transcript (mathematics, science and English grades listed separately), overall grade point average, solutions to two of three logic problems, PSAT or ACT scores, and optionally, two letters of recommendation.

Nominators advise prospective participants that ACT or PSAT tests need to be taken far enough in advance so that test results can be included in the nomination. This requires an early testing date with probable adverse affects on the student's score. However, no other universal measure is available.

Solutions to logic problems measure a student's problem solving skills, writing skills and seriousness of interest in the Program. Some problem statements contain excessive information; others contain incomplete

information. This further measures a student's ability to discern and assimilate information.

Information contained in the nominations is used to select students who will have a successful and positive experience in the Program. A participant is selected if he or she (1) performs well on either the PSAT or the ACT (2) has a high grade point average or (3) performs well in both areas considered together. A persuasive call influences the selection process. Most letters of recommendation play inconsequential roles in the selection process.

ACHIEVING ENROLLMENT

Scholarships covering full tuition and laboratory fees are awarded to all participants. The Computer Science Department funds these scholarships with the anticipation of realizing a return in its increased ability to attract bright students.

Letters of acceptance are sent in mid April, with a registration form, an explanation of campus standards, a campus housing request form and a campus map. To motivate attendance, a $10 nuisance fee is required at the time of registration. The letter of acceptance advises each prospective participant that the Program will require his or her full energies to complete the course of study. The registration form is to be returned by June 1. The following week, a follow-up letter is sent to each prospective participant who has not registered. The letter expresses personal interest in the student's participation, contains a description of the advantages of the Program and inquires if the acceptance has been received. As a result, several additional registration forms are received.

MAINTAINING ENROLLMENT AND ACHIEVING COMPLETION

Four specific steps are taken to maximize the quality of the Program: 1) the pace of the course is fitted to the class's performance and assimilation of the material, 2) assistance is provided by resource personnel, 3) participants are encouraged to help each other complete the course and 4) group social activities are offered. An "esprit de corps" develops which helps each participant feel that his or her personal success is important to the overall success of the Program and to the other participants.

COURSE CONTENT

The AHSSP is the equivalent of a two hour course in structured design and Pascal programming. The Association for Computing Machinery's CS1 guidelines are followed. Selection, looping, procedures, parameters, functions, real numbers, control structures, arrays and records are presented during eight intense days of instruction.

IMPLEMENTATION

SETTING

The twelve-day Program is held on the Brigham Young University campus in late summer to accommodate the participants' summer jobs, vacations, fall football practices, Boy's/Girl's State and other activities. The daily schedule is:

8:00 - 10:00 a.m.	Lecture
10:00 a.m. - 1:00 p.m.	Computer laboratory, study and lunch
1:00 - 3:00 p.m.	Lecture
3:00 - 11:00 p.m.	Computer laboratory, study and dinner

This schedule is followed Monday through Friday of the first week and Monday through Wednesday of the second week. The laboratory is open all day Saturday. Sunday is observed as a day of rest. The final examination is given on Thursday of the second week. A graduation exercise is held on Friday.

Each participant completes 13 written assignments and 15 programs. Each participant takes a 2-hour programming examination and a 2-hour multiple choice examination.

Outside of class instruction, participants are tutored by teaching assistants, dormitory resident assistants and other participants; peer assistance with concepts is encouraged.

Families and friends of participants are invited to the graduation exercise. Certificates of completion are presented following a speech by a prominent University official. The AHSSP graduation is held on the same day as summer commencement, presenting the University in its most favorable light.

Those who successfully complete the Program are eligible for two hours of university credit upon enrollment at Brigham Young University.

At present, there are no provisions for students to maintain their skills during the year between completion of the AHSSP and enrollment in a college or university. To remedy this situation, each nominator for AHSSP '87 will be asked to describe a follow-up program (teacher assisting, advanced placement class instruction, etc.) for each participant when he or she returns to the high school. The skills and training of each participant can then be maintained and can contribute to the high school environment.

INSTURCTORS, TA'S AND RA'S

A senior faculty member presents the lectures. Outstanding, personable students in Computer Science are selected to be teaching assistants and dormitory resident assistants. The number of student assistants varies with the number of participants. During 1986, twelve teaching assistants graded assignments, assisted in the laboratory and conducted help sessions; three resident assistants lived in the dormitories with the participants and were available during the evenings to answer questions.

EQUIPMENT AND FACILITIES

Equipment and facilities provided by the University include classrooms, computer laboratories and campus dormitories. In the past the computer laboratories contained 45 Apple II Plus computers; in the future they will be equipped with AT&T 6300s. Texts are provided complimentarily by the publisher.

COSTS

Costs incurred by the Computer Science Department include mailings, salary for the instructor and teaching assistants, diskettes, handouts and graduation certificates. The costs of food, housing, resident assistants and entertainment are met by students staying in the dormitories. Participant costs are held intentionally to a minimum.

SUPPORT/DIVERSION ACTIVITIES

Social activities include a swimming party, dance, movie and barbecue, scheduled throughout the two week period. Spontaneous activities are common. Activities unify the group, encourage student integration and play an important role in creating an environment conducive to individual success.

Integration is encouraged with activities and alphabetical seating in the classroom which also facilitates attendance taking. The participants form new friendships which thrive during and after the Program. Subsequent reunions planned spontaneously by the participants are common. Peer association consistently is listed as one of the most outstanding aspects of the Program.

The teaching assistants, resident assistants, the instructor and the instructor's family participate in the group social activities and provide a familial atmosphere. All offer encouragement, guidance and friendship.

EVALUATION

STUDENT PERFORMANCE

Students are graded on an absolute percentage scale (80/90/100) with the following weights: written assignments, 15%; programming assignments, 45%; examinations, 30%; attendance, 10%. A overall percentage of 66.7 is required to graduate.
On the final examinations, the percentage of students performing at or above 66.7% has increased each year; for years '83 through '86, the percentages have been 61, 88, 93 and 94 respectively. In the case of low scores, completion of programs and written assignments typically raises the participant's score to the passing level.

Participants in AHSSP '83 and AHSSP '84 received grades ranging from A to E. During 1985, all 55 students graduated with a B- or better. In 1986, every student who stayed for the duration of the Program (two left early for personal reasons) graduated with a B or better. While the difficulty of the course has remained fairly constant, the participant selection process has been improved and the maximum number of participants has been limited.

STUDENT RESPONSE AND BEHAVIOR

During the first two days of the Program, participant workloads vary widely. Participants with previous programming experience find the material quite easy; participants with minimal or no experience find the material challenging. To accommodate the wide spectrum of experience, the first few assignments are intentionally uncomplicated. From the third day to the Program's end, most participants find the material demanding and challenging. Overall percentages show no correlation with previous experience. To offset the initial contrast between experienced and inexperienced participants, introductory material may be mailed to AHSSP '87 participants prior to the start of the Program.

To maximize student learning, lecture sessions are limited to two hours. Since student tardiness in an intense, demanding program can become a problem, attendance is taken at the beginning of each lecture and becomes a part of the grade.

At the end of each lecture, sign-up sheets are made available to schedule prime laboratory time. Laboratory facilities are allocated according to availability and enrollment. During prime laboratory times (10:00 a.m. - 1:00 p.m. and 3:00 -6:00 p.m.), students may sign up for one hour of computer time with the option of additional use of a vacant computer. During non-prime times, laboratory facilities are available on a first come, first served basis. An overflow laboratory provides additional computers. Teaching assistants are available to answer questions, guide programming logic and assist in debugging programs. Help sections are scheduled throughout the day.

Many participants experience homesickness during the first few days of the course. The difficulty of the course causes some participants to question their ability to complete it. This sense of anxiety and frustration increases until the end of the first week. During the last full week of the course, the group feeling becomes "we can do it; we're surviving." The last few days are marked by a feeling of pride and group accomplishment.

Approximately 65% of the participants stay with relatives or friends near the University to reduce housing expenses, but with minimal opportunities for association with their hosts. The remaining 35% who stay in the dormitories enjoy the advantages of proximity to course facilities, around-the-clock interaction with other participants and absence of distractions.

Written student reactions are obtained through an anonymous two-page survey completed and returned after graduation. Graduates provide responses from 0 to 10 (0=strong NO; 10=strong YES) as well as written comments. Questions and tabulated results are provided in Figure 2.

ACCOMPLISHMENT OF OBJECTIVES

The AHSSP has a cumulative completion rate of 95%. The 1985 Program achieved 100% completion and AHSSP '86 graduated every student who stayed to the end of the Program (75 out of 77). The measures used to evaluate students in the equivalent university course indicate that intended design and programming skills are acquired.

The participants respond successfully to the extraordinary demands of the course. All are accustomed to finishing at or near the top of the class. Most are unwilling to accept performance at a lower level. Their abilities

and established self-expectations together with the extraordinary demands of the Program permit them to graduate with a keen sense of growth and accomplishment.

Of 391 participants in Programs '83 - '85, 108 are known to have enrolled at Brigham Young University and 41 have requested credit. Six AHSSP graduates are currently Computer Science majors at Brigham Young University.

ANALYSIS

ACHIEVING SUCCESS

Of 130 initial participants in the 1985 and 1986 Programs, 128 completed the course of study and graduated. In comparison to previous years, this success can be attributed to 1) more thorough screening of nominees and 2) smaller groups; participants received more individual attention and had better access to the computers. The enrollments of over 200 in 1983 and nearly 175 in 1984 strained resources and left discouraged several students who failed to complete the course. The subsequent levels of 57 and 77 have improved the Program from the points of view of the participants and the University.

HIGH SCHOOL BENEFITS

The high schools benefit when an AHSSP graduate returns and shares his or her new skills. Observing the abilities of the graduates, teachers are encouraged to seek similar training.

STRENGTHS AND WEAKNESSES

Observed strengths of the Program include:

- the success rate
- the positive atmosphere created by the students and faculty
- good instruction and assistance
- favorable impressions made by the University and the Computer Science Department

Points of concern or weakness include:

- the number of potential nominators who do not advise students of the opportunity
- no universal measure except early PSAT/ACT scores
- no pre-course session for participants lacking computer experience
- limited time to present the course
- the small number of Computer Science majors generated by the Program

Issues about which little is known include:

- success in attracting students who otherwise would not come to Brigham Young University
- retention of knowledge

DECREASED ENROLLMENTS

Decreased enrollments are attributed to two factors: 1) Computer Science is becoming more available in the high schools; student need to look to the universities for instruction is decreasing, and 2) a decline in the glamour of Computer Science. National enrollments in computer related majors are diminishing. Enrollments at Brigham Young University have followed national trends, declining from 6% in 1982 to 3% in 1986.

FUTURE

The future of the AHSSP is bright. Student interest and continued endorsement by the University will perpetuate the AHSSP for the foreseeable future.

1. Were you already familiar with computing?
2. Were you already familiar with Pascal?
3. Did the course produce new and useful knowledge, skills and awareness in you?
4. Did the class presentations help you learn the material?
5. Did the help sessions help you learn the material?
6. Did the written homework help you learn the material?
7. Did the programming assignments help you learn the material?
8. Were you generally motivated to learn the material?
9. Were the final examinations appropriate measures of your knowledge, understanding and ability to perform?
10. Will the Advanced High School Studies Program be a pleasant memory for you?*

1983	1984	1985	1986	
--	--	6.08	5.05	1
--	--	2.42	1.09	2
8.91	9.41	8.35	8.50	3
6.54	7.35	7.12	7.59	4
--	5.03	6.57	5.82	5
8.16	7.71	7.00	6.27	6
--	8.65	8.81	8.77	7
8.00	8.71	8.65	8.36	8
7.17	9.00	8.38	7.05	9
9.60	9.00	9.42	9.18	10

* Several responses exceeded 10. These are tabulated as 10.
0 = strong NO, 10 = strong YES

Figure 2
STUDENT RESPONSES

SELECTING SOFTWARE FOR THE BUSINESS DATA PROCESSING COURSE

SUMIT SIRCAR
Department of Information Systems and Management Sciences
College of Business Administration
University of Texas at Arlington
Arlington, TX 76019

Although there is basic agreement on the objectives of software training for the introductory BDP course, several pedagogical decisions need to be made. These involve the need to teach programming and the particular kinds of software packages and associated hardware required. Other issues include: the value of integrated software versus stand-alone software; low-end versus full-featured software; menu-driven versus command-driven software; crippled software; and bundling of textbooks. Three packages are compared to illustrate these issues.

INTRODUCTION

The widespread use of computers in all walks of life, including business, has made the introductory Business Data Processing course one of the most important and valuable courses in the curriculum of colleges of business administration. Given the dynamic nature of the field, the course description for this course has evolved rapidly over the last few years to the point that it has now achieved some stability with a selection of excellent text books available. The major concern has now shifted away from the text book to the software to be used. Selection of the appropriate software has major long-term implications for the student. An unpleasant learning experience could provide an insurmountable mental block for some, or at least hinder their familiarity with computing. On the other hand, expertise in the use of appropriate computing tools should be an invaluable advantage in a wide variety of settings, including subsequent courses.

This paper starts with a statement of objectives for the software training provided in the business data processing course. Several relevant issues are then discussed, which make the selection process quite complex.

OBJECTIVES OF SOFTWARE TRAINING

The objectives for requiring exposure to computer software must obviously be a subset of the objectives of the course itself. The ability to use computers will almost certainly be one of the goals. Because the majority of the students will not be information systems majors or be likely to become professional programmers, the following guidelines are offered as appropriate targets. The students are sophomores who will be entering the work-force several years from now. At that time they will be using terminals for "knowledge work," i.e. word and text processing, storage and retrieval of data, communications, decision support, graphics, and end-user applications. Ideally, they should be honing these skills in the various courses they take while preparing for their careers in the business world.

Assuming there is general agreement on these goals, several issues need to be resolved for implementation. Only three of these are purely pedagogical--the others have to do with administrative or availability constraints. A brief discussion of these follows.

PEDAGOGICAL DECISIONS

1. Inclusion of BASIC. For several years, BASIC has been the most widely used programming language for this course. Lately, however, the advent of packaged software for personal computers has caused a large number of schools to substitute them for instruction in BASIC. Is this decision consistent with the objectives outlined above? For much of the knowledge work described, BASIC is unnecessary, because of the availability of software packages. However, several problem-solving situations may require the user to write his own program, whereupon programming skills would be useful. BASIC (or some other appropriate interactive language) should therefore be taught in order to meet the objectives fully. If, however, it is infeasible to accommodate both a programming language and several software packages in the course, the former should be dropped, simply because the majority of end-user computing tasks can be accomplished without recourse to programming. Almost every text book for this course comes in with-BASIC and without-BASIC versions.

A consequence (or is it the cause?) of the popularity of PC software packages for spreadsheets, file management and word processing, has been the shift away from distributed computing facilities for this course, to stand-alone PC laboratories having IBM PC compatibility.

2. Integrated Software versus Stand-Alone. There is some controversy over whether to use integrated software packages instead of separate packages for word processing, spreadsheets, file management and data communications. From the viewpoint of having to prepare comprehensive documents containing several elements of these, it would appear that integrated packages are required. Integrated packages also provide a common interface for the user, so moving from one application to another would not require major relearning. Till recently, the major integrated packages (Symphony, Framework, and Enable) have been unable to even approach the popularity of stand-alone software such as Wordstar, Lotus 1-2-3 and dBase III. The latter are generally better at what they do individually. Moreover, the integrated packages are usually more difficult to learn and use, leading many users to give up the advantages of integration, which they often do not need.

A new breed of integrated software (such as First Choice, Able-One, DeskMate/PC, Electric Desk, and Words & Figures), attempts to alleviate these problems by offering less power and reducing complexity. An in-between approach is to sell separate packages which are not integrated, but have a common user interface. "Common" should read "similar" in this context, because the interface for word processing is necessarily very different from that for spreadsheets or data bases.

3. Ease of Learning and Use. There are other factors affecting ease of learning and use, apart from the question of integration. This is an issue of major concern, because very little class time can be devoted to teaching the actual use of the packages. Students are expected to pick up many of the features of the software by themselves from the documentation, with occasional help from a laboratory assistant, when available. Factors to consider in this regard are as follows:

3.1. Workbooks/Documentation. Manuals

supplied by the software vendor can only be used for reference, and not as a teaching tool, like a text book. Student workbooks written by text book authors are imperative. The most helpful are those which provide actual key-by-key instructions on how to accomplish the various operations that the software is capable of.

3.2. Low-end Versus Full-featured Software. Although not necessarily so, full-featured software tends to be more complex to learn and use than so-called low-end software, such as the PFS series. Ideally, it would be nice to have more rather than fewer features available to the student, but there is a definite trade-off in terms of complexity. For example, there are 24 spreadsheet functions available in PFS:First Choice, whereas Symphony has 90. Software Publishing Corporation, publishers of the PFS series, offers a way out of the dilemma for advanced users by selling the PFS Professional Series. However, this may not be an option for business schools.

3.3. Menu-driven Versus Command-driven Software. Menu-driven software is far easier to learn and therefore should be a required characteristic of the software selected. However, advanced users may become tired of menus and may prefer to use commands sometimes, so this option is also desirable. Menus are also a lifesaver for the intermittent user, who cannot remember commands.

4. Crippled Software. Many instructors would like to use software packages that have wide acceptance in the commercial world. Not only would their students be using powerful software, but it would be of value on their resumes and at work. Unfortunately, many of the most widely used products. e.g. Wordstar, dBase II and Lotus 1-2-3, may not be representative of the state-of-the-art, and in fact may no longer be the best sellers in their category. Another problem is that vendors of always cripple their products for the educational product, to protect their commercial sales. The crippling may be one or both of two kinds: crippled features or crippled data handling capability. After much resistance, even Lotus 1-2-2 and WordPerfect have recently been released in educational versions.

5. Bundled Text Books. A major constraint in the selection of software is, of course, their availability at prices affordable by students. Textbook publishers have attacked the problem by establishing arrangements with software vendors to package their software with one or more textbooks. The exact nature of this "bundling" can be crucial. PFS:First Choice, for example, comes bundled with the Athey and Zmud textbook, Computers and Information Systems, and another variant. PerfectWare, however, comes with a variety of possible books, one of which can be used as the instruction manual for the software itself, leaving the school free to choose any text it prefers. Similarly, Enable (educational version) comes with a textbook specially written for it.

The method of providing the software also varies. Some publishers include the diskettes with the texts, while others merely make several copies available to the school for the students to work with. The latter usually provide an option for the student to buy copies for himself at extra cost.

6. Vendor Stability. Although vendor stability and maturity are not as important as in a business setting, there are several advantages to using software from a major vendor. First, the software is more or less debugged. Second, upgrades can be expected from time to time, although this may not necessarily be true as in the case of PerfectWare. Third, students prefer to use name-brand software compared to

no-name or generic software. Finally, there is a far greater chance of being able to interface with other software.

SOME SOFTWARE EXAMPLES

Assuming that integrated software is the preferred approach, or at least modules with common user interfaces, it is interesting to compare the following three packages, using the factors outlined above: Enable, PFS:First Choice, and PerfectWare. Each of these products has been adopted by a large number of schools. Rather than describing every feature of each package, an attempt will be made to highlight the major advantages and disadvantages of each product from the viewpoint of course selection.

ENABLE

Educational version of a major commercial product, sold with specially prepared manual by Charles Spezzano for $35. Not bundled with any text. Is an "all-in-one" integrated package, containing word processing, spreadsheet, database, graphics and telecommunications capabilities. The full-featured version was considered superior (on balance) to Framework and Symphony by PC Magazine (June 10, 1986). Chief disadvantages are the need to learn an extensive command set, and a rather poor word processing module. The educational version is rather severely crippled: a 10-page limit for the word processor; 50 rows X 50 columns for the spreadsheet; and 50 records for the data base.

PFS:FIRST CHOICE

This is an integrated version of the popular PFS Series. It was designed for the educational market, more specifically for Pcs with 256K of memory. The designers also limited themselves to just two diskettes (one of which is a dictionary) and were therefore forced to drop the graphics capability. Included are word processing, spreadsheet, file management and telecommunications capabilities. The package and an accompanying workbook are bundled free with the Athey and Zmud texts referred to earlier, which may be unpalatable to some schools. Diskettes are only supplied to the school--students can buy their own set for $35, with complete documentation.

First Choice is very easy to learn, being menu driven but with the ability to suppress menus when learned. The different modules are not as tightly integrated as Enable--only one file can be open at a time, and all transfers must take place via a "clip board." When a hard disk is not being used, there can be an annoying degree of shuffling diskettes in and out. Although it is not as fully featured as some of the heavyweights around, First Choice is considered very suitable for beginners (PC Magazine, February, 1987). Datapro also ranked it second to Framework, in a survey of nine different packages (November, 1986).

PERFECTWARE

This is an older series of commercially popular software products, dating from 1981. Surprisingly, the full-featured commercial products, except for telecommunications and graphics software, will be made available free to schools upon adoption of any one of a series of books. The educational version will be sold to students for $7.50. One of the book choices happens to be a textbook for the software, written by Wilson Price. The school can therefore choose any other text for the course.

The PerfectWare components (PerfectWriter, PerfectCalc, and PerfectFiler) are not tightly coupled as Enable and First Choice, probably due to their earlier vintage. All the components do present as fairly standard user interface, however. It is possible to generate a report containing text and

spreadsheets by patching together different files. The word processing module is very impressive and easy to learn, while the spreadsheet and file management products have been surpassed by more recent software, in terms of capabilities.

SUMMARY

The selection of the appropriate software for the business data processing course is indeed a complex process. In some cases, the process includes selection of the textbook for the course, because of bundling by publishers. Actual hands-on use is recommended before any decision, and this makes the process far more time-consuming than for textbooks. Whereas a textbook may take a couple of hours to review, a software product may take four or five times that long to evaluate.

The foregoing discussion is intended to serve as a framework for comparing different competing products. The description of Enable, PFS:First Choice and PerfectWare may serve as a basis for evaluating other products.

COMING TO GRIPS WITH THE MANAGEMENT OF INFORMATION: AN EXPERIENTIAL APPROACH

Tim O. Peterson
Department of Management
Texas A&M University

Jon W. Beard
Department of Management
Texas A&M University

Dorothy J. McBride
School of Systems and Logistics
Air Force Institute of Technology

ABSTRACT

This experiential exercise introduces students to some of the basic concepts about information and the importance of managing information as a resource. It addresses evaluating information on the basis of its quality (accuracy), timeliness, quantity and relevance. In addition, the exercise helps students to recognize the different information needs of managers in different functional areas as well as at different levels within the organization. This paper provides the reader with the instructions and materials to use this exercise in classroom setting.

INTRODUCTION

Many of the current foundation textbooks in Management Information Systems (MIS) (Lucas, 1986; Murdick & Munson, 1986; Sprague & McNurlin, 1986; Davis & Olson, 1985; Kanter, 1984; Hicks, 1984; Hodge, Fleck, & Honess, 1984; McLeod, 1983; Murdick, 1980) include a chapter or a major portion of a chapter on the topic of information. These chapters generally include topics such as the meaning or definition of information, the attributes of information, the value of information, and the characteristics of information. The textbooks all seem to want to drive one point home. The point is that information is a resource just like materials, labor, and capital, and this resource must be managed like these other resources. The problem is that Master of Business Administration (MBA) students seem to have some difficulty in grasping this conceptual point about information. In-class experiential exercises on this topic presently do not exist.

The proliferation of personal computers as well as other innovations in information technology makes it absolutely essential that managers are familiar, not only with the technology, but also with the underlying concepts of information, its flow, characteristics, value, and attributions in organizations. McBride and Peterson (1985) developed an experiential exercise to be used in an undergraduate Principles of Management class to introduce students to the concepts of Management Information Systems. The exercise presented here extends the original exercise to be used in an MIS course for MBA students or in an

introductory Systems Analysis course at either the graduate or undergraduate level. The exercise helps students to appreciate the potential complexity of information flow in organizations, why it is necessary to assess an organization's information needs, how to evaluate available information, and why managers at different levels of the organization have different information needs.

THE EXPERIENTIAL EXERCISE

Objectives: At the conclusion of the exercise, the student should be able to:

1. Appreciate the complexity of information flow in an organization.
2. Given an objective or decision criteria, identify basic information needs.
3. Evaluate information on the basis of its quality, timeliness, quantity, and relevance.
4. Explain why managers at different levels of the organization have different information needs.

Time Required: We have completed this exercise in a 50-minute class sessions, but the debriefing and discussion is much more effective in a 75 to 90-minute session. Another alternative is to complete the exercise one class period and process the experience the next class period.

Materials Required (Available from the authors):
1. A copy of the initial description of Solutions Diversified, Inc., for each student.
2. A copy of the instruction sheet for each division (President, Inventory Control, Quality Control, SOLTECH, SOLGRAPHICS, and SOLART).
3. A division name card (folded index card) for each division.
4. Report forms for SOLTECH, Inventory Control, and Quality Control.
5. Connect-the-dot puzzles (6 sets of 10) These numbers must be doubled if two companies are being run simultaneously. These are available from the first author.
6. Math problems (5 sets of 12) These numbers must be doubled if two companies are being run simultaneously. These are available from the first author.
7. Math solutions (one set for each company) These are available from the first author.
8. Crayons (at least one box of eight for each company).
9. Observation sheets and complete copies of the instructions for the observers.

Group Size: The exercise can be used in classes of six students up to 50 students. Each company works best with a group size from 9 to 14 participants. The minimum number of participants in a company is 6. In classes larger than 14 students, you can assign some students as observers, or operate several companies at the same time. (The addition of other companies injects the potential for competitors in the environment which adds the concept of external information sources to the exercise.)

Instructions:
1. One lesson before running the exercise, give each student a copy of the half-page description of Solutions Diversified, Inc. Reading this description should be required preparation for the next class meeting.
2. Before class starts, arrange the desks or tables so each division has a distinct work space. Place the division name cards on each desk or table to identify these work spaces.
3. Assign the students to divisions so that you have:
   - 1 President,
   - 1 or 2 in Inventory Control,
   - 1 or 2 in Quality Control,
   - 2 or 3 in SOLTECH,
   - 2 or 3 in SOLGRAPHICS,
   - 2 or 3 in SOLART.

The President is allowed to reassign people, but be sure to note these

reassignments so you can discuss why (s)he made that decision, what information (s)he used, what the sources of that information was to make the decision, and how (s)he obtained the information. Any remaining people can be used as impartial observers.
4. Assign a supervisor in any division with more than one person and give the supervisor the appropriate instruction sheet and materials (report forms for SOLTECH, Inventory Control and Quality Control; sets of puzzles and math problems to Inventory Control; and crayons to SOLART).
5. Remind the students to be alert for information issues and tell them to start. Be sure they are aware of the start time so they can use it as the basis for turning in periodic reports on schedule.
6. Allow the exercise to continue for 30 minutes (40 to 45 minutes if you have more than a 50-minute session). Longer sessions do require additional sets of puzzles.
7. Each time Quality Control collects ten problems from SOLTECH, you will be expected to provide the matched solutions. Provide solutions only for problems attempted.
8. Some procedures are purposely left vague in the instructions. Be alert to how the group resolves issues of missing or confusing information. Avoid answering questions concerning their written instructions. Some specific issues that may arise are:

a. Restructuring the organization by combining functions, creating new ones, relocating work spaces, reassigning people.

b. Revising the various reporting procedures by changing the content or timing of reports.

c. Specifying procedures with Inventory Control concerning the number of packs of puzzles that can be checked out at a time, whether finished goods inventory can be checked in individually or must be maintained in original packs, precisely how to fill in the Inventory Transaction Log, and whether or not to maintain control of goods-in-process inventory.

d. Revising the economic batch size for Quality Control testing, adding a quality control step for the products of SOLGRAPHICS and SOLART, and determining if preliminary checking or feedback to SOLTECH with an opportunity for reworking problems would be appropriate.

DEBRIEFING THE EXERCISE

During the remaining class time you can discuss the information issues that come up during the exercise. You might focus attention on issues concerning the objectives stated earlier.

1. Complexity of information flow in the organization:

a. Consider the volume of information that was generated during the exercise in various forms (written instructions and reports, verbal communication, possibly graphics, etc.) and with various content (probably mostly task oriented, but possibly social as well).

b. Recognize that typical organizations will also be faced with volumes of additional information, both internal (schedules, budgets, and personnel records, etc.) and external (concerning customers, suppliers, competitors, legislation, the economy, etc.).

2. Assessing information needs:

a. To track progress toward the productivity objective, the President needs information on correct solutions generated per hour and some specific target. The information system ought to provide that information such that the President can use it directly with no further calculation or manipulation and perhaps on an exception basis when productivity levels differ significantly from the target. The target might be based on

past performance, customer demands, competitor performance, etc.

b. The production divisions ought to be getting feedback on their productivity.

c. Should they have other information such as the status of goods-in-process inventory, categories of errors causing rejects, etc.?

3. Evaluating information:

a. Quality (accuracy)

-If the organization assumes that products from SOLGRAPHICS and SOLART are correct when they reach Inventory Control, they have inaccurate information on productivity.

-How accurate are the reports that exist?

-What level of accuracy is appropriate?

-The format of the Inventory Transaction Log makes that information difficult to use.

-Math problems and dot puzzles come in different size packets, but that's not clear on the form.

-There's no clear sequence for entering data on the form, so raw materials and finished goods inventory and different product inventories could all be mixed in a series of entries.

-Its not clear whether finished goods are kept in packets or individually.

b. Timeliness

-What is the appropriate timing for reports and feedback at various levels in the organization?

-Does limiting the timeframe of the exercise have an impact on the appropriate timing? How would it be different in a functioning organization?

c. Quantity

-Do the existing reports provide the right amount of information for decision making in the organization? Why or why not? (Recall the assessment of information needs above.)

-Note that giving the Inventory Transaction Log to the President provides much data that the President doesn't need.

d. Relevance

- President tends to get details (especially with ithe Inventory Transaction Log) that are irrelevant compared to getting a productivity measure, which is not provided.

4. Different information needs at different levels of the organization:

a. Production level and staff activities need details in their areas or responsibility.

-SOLTECH may recognize that the math problems fall in categories (probability, algebra, geometry, etc.), may have or ask for information on specific skills or preferences of the people in the work group, and may decide to specialize.

-Inventory Control tracks details of inventory.

-Quality Control has access to specific solutions to math problems.

-Real organizations would similarly generate information in other areas (accounting, finance, personnel, marketing, etc.)

b. Top level managers need summarized information and more external information.

-The President should only need the overall productivity figures (perhaps on an exception basis), not the details.

-External information on customer needs, competitor actions, status of the economy, government decisions, etc., would be necessary to put the internal information in perspective and to keep the company competitive.

-Much of the President's information may be gathered informally (touring the plant, telephone conversations, golf matches, etc.) rather than through formal reports.

* References available upon request from the first author.

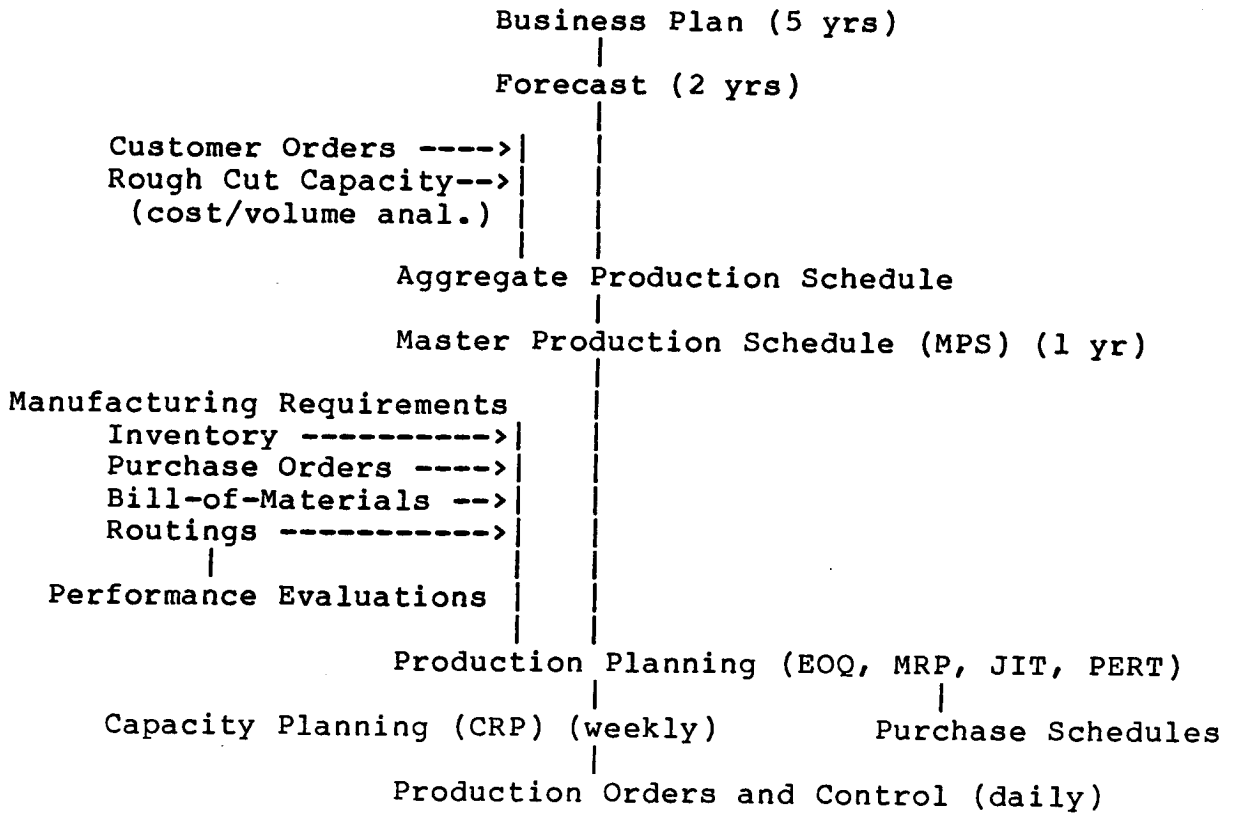
DECISION SUPPORT SYSTEMS WHAT CAN BE DONE WITH THE TOOLS WE ALREADY HAVE?

Gerhard Plenert
School of Business
Department of Accounting and Management Science
California State University, Chico
Chico, CA 95929-0011
(916) 895-5876

THE FUNCTION OF DECISION SUPPORT SYSTEMS

Decision Support Systems (DSS) incorporates tools that help a decision-maker make decisions when there doesn't seem to be a right answer available. Many of these decisions are made strictly by intuition because the manager hasn't been shown that tools already exist that can help in the formulation of a "best guess". In this presentation, I will discuss the requirements of Decision Support Systems in a manufacturing environment, and I will discuss some of the analytical tools that are needed to perform this "best guess" analysis. I will review some existing software packages that can be used to perform decision analysis.

It isn't necessary to wait for sophisticated Expert Systems or Artificial Intelligence (AI) Systems in order to take advantage of DSS analysis. Most of the necessary tools are available now. We should start using these tools, and as we become experts at using them, the development of Expert Systems that incorporate them will be much easier as well. I will start by analyzing what tools we need and what tools are available for us to satisfy these needs. But first, it is necessary to get the entire system into perspective. To do this, I will start by reviewing the total information flow required for a manufacturing organization. This is diagrammed in the following chart.



When using this diagram, the first rule is that no system in the diagram will function properly unless all the other systems above it are functioning properly. You cannot produce a valid Master Production Schedule without having a valid, effective, and usable Business Plan, Forecast, Aggregate Production Schedule, and Rough Cut Capacity Plan.

A second rule is that all steps are not used by all businesses. For example, CRP and Production Orders and Control would not be appropriate in a retailing environment.

In looking at decision analysis for these functions, a variety of tools are available to perform the required analysis. I have listed some of the appropriate and most commonly used tools for each but many are not listed.

System	Decision Analysis Tools
Business Plan (5 yrs)	
	Portfolio Analysis
	Financial Analysis
	Probability Analysis
Forecast (2 yrs)	
	Sales Strategy Analysis
	Statistical Analysis
	Simple, Multiple, and Multivariate Regression
	Moving Averages
	Exponential Smoothing
Customer Orders	
	Order Tracking
Rough Cut Capacity	
	Cost/Volume Analysis
	Linear Programming
	Make/Buy Decision Analysis
Aggregate Production Schedule	
	Linear Programming
	Financial Analysis
	Transportation Model
	Assignment Scheduling
Master Production Schedule (MPS)	
	Planning Bill-of-Materials
	Product Mix Planning
Inventory	
	Perpetual Inventory System
Purchase Orders	
	Accounts Payable Open Order System
Bill-of-Materials	
	Requirements Explosion
Routings	
	Lead Time Analysis
Performance Evaluations	
	Labor Efficiency Reporting
Production Planning	
	Economic Order Quantity (EOQ)
	Material Requirements Planning
	Just-In-Time (JIT)
	Project Evaluation and Review Technique (PERT)
Purchase Schedules	
	Accounts Payable Open Order System
Capacity Planning (CRP) (weekly)	
	Work Load Analysis
	Make/Buy Decision Analysis
Production Orders and Control	
	Johnson's Rule
	Job Order Scheduling Rules

Having developed a list of tools required for DSS analysis, we can now see what data processing tools are readily available to perform these functions.

WHAT COMPUTER TOOLS ARE AVAILABLE?

DSS attempts to apply user friendly systems that will allow any of the above analysis functions to be programmed quickly and efficiently. Unfortunately, many of the existing analysis packages are incomplete, expensive, and not available on a microcomputer. I have found it necessary to perform a multitude of analysis jobs without the aid of these packages. I used a Personnal Computer (PC) with a few basic software packages, and I found that I could do most of the work listed above with the following tools:

- Spreadsheet Software (SS)
- Word Processing (WP)
- Relational Data Base (DB)
- Linear Program Package (LP)

Regression Package (R)
Basic Programming Language (B)

A large variety of spreadsheet packages exist. Most microcomputers have some kind. Currently the most popular is Lotus 123. Others such as Visicalc and Supercalc are also popular and will do just as good a job in most cases. To maximize effectiveness, a spreadsheet that has graphics capability would be the best.

There is also quite a variety of word processors. These have been valuable in doing the write-ups that go with the DSS analysis. It is helpful to have a word processor that is compatible with the spreadsheet and the data base manager. Then information generated by either of these can be transferred directly to the body of your report.

Relational data bases such as DBASE II, DBASE III, RBASE 5000 allow tables, charts, and lists of information to be compiled. These can be used for name and address lists and are extremely valuable for inventory analysis or purchase order analysis.

Linear programming packages such as LINDO are affordable and easy to acquire. Any LP package will do the job. The same is also true of regression packages such as MINITAB. A variety of PC versions are available.

Basic is listed as a tool, not because it will be necessary, but because it is a catch-all that will allow simple formulations to be programmed quickly. In all my academic experience, never once have the students needed to use Basic to perform any of the above functions. But it may be necessary for exceptions.

Using the listed tools, I will next discuss which tool I used to perform each of the required analysis.

Decision Analysis Tool	Software Package Used
Accounts Payable Open Order System	DB & WP
Assignment Scheduling	LP
Cost/Volume Analysis	SS
Economic Order Quantity (EOQ)	SS
Exponential Smoothing	SS
Financial Analysis	SS
Job Order Scheduling Rules	SS
Johnson's Rule	SS
Just-In-Time (JIT)	SS
Labor Efficiency Reporting	DB & SS
Lead Time Analysis	DB
Linear Programming	LP
Make/Buy Decision Analysis	SS & LP
Material Requirements Planning (MRP)	SS & DB
Moving Averages	SS
Order Tracking	DB & WP
Perpetual Inventory System	DB
Planning Bill-of-Materials	SS & DB
Portfolio Analysis	SS
Probability Analysis	SS
Product Mix Planning	LP
Project Evaluation and Review Techniqie (PERT)	SS & LP
Regression	R
Requirements Explosion	SS & DB
Sales Strategy Analysis	SS,DB,&WP
Statistical Analysis	SS
Transportation Model	LP
Work Load Analysis	DB & SS

HOW IT WORKS

Using financial analysis as an example, I will now demonstrate how these tools work. In order to perform financial analysis, set up a spreadsheet that has the following layout (may vary depending upon the country, tax

system, and what you are analyzing):

Year	1	2
Gross Revenues		
- Royalties		
Net Revenues		
- Operating Costs		
- Depreciation		
- Amortization		
- Book Write-off		
- Depletion		
- Loss Forward		
Taxable Income		
- Tax		
Net Income		
+ Loss Forward		
+ Depreciation		
+ Amortization		
+ Book Write-off		
+ Depletion		
- Capital Costs		
Cash Flow		

Using the information from the Cash Flow row, financial evaluations can be done on Net Present Value (NPV) or Rates of Return (ROR). These computations are automatically available in most spreadsheet software packages.

A more complicated example would be forecasting. Here we are trying to predict lumber sales, for example. In order to do this, we may have information available to us about housing starts and the cost of living index. The steps required for forecasting analysis would be as follows:

1) Load the information about lumber sales, housing starts, seasonality factors (these can be computed on the spreadsheet if necessary), and the cost of living index each into their own column on a spreadsheet.

2) Deseasonalize lumber sales and housing starts into other spreadsheet columns.

3) Compute Exponential Smoothing forecasts for the entire history of deseasonalized lumber sales data into another column of the spreadsheet.

4) Compute MAD (Mean Absolute Deviation) and MSE (Mean Squared Error) values for the Exponential Smoothing data.

5) Perform steps 3 and 4 for several different alpha factors in Exponential Smoothing.

6) Perform steps 3 and 4 for several different Moving Average Models as well.

7) Steps 3 and 4 can be repeated for any time series modeling method, including double and triple Exponential Smoothing and double or triple Moving Average.

8) Time Series Regression can be used by taking the lumber sales data, running it through the regression model and coming up with the time series equation. This equation is then taken back into the spreadsheet and steps 3 and 4 are repeated.

9) To perform associate modeling, use the regression package and generate as many model equations as you feel appropriate. For example, (1) using housing starts as a predictor of lumber sales, (2) using cost-of-living as a predictor of lumber sales, and (3) using both housing starts and cost-of-living as a predictor of lumber sales. This gives you three models, each of which can be analyzed again on the spreadsheet via steps 3 and 4.

10) At this point, several models are being tested on the spreadsheet. Select the one giving the best performance (using the MAD and MSE values to select). Then, using this best model, forecast future lumber sales. The final

step is to put seasonality back into the prediction, and you have a forecasting model.

11) Once the spreadsheet has been set up, forecasting in the future is simply a process of running the selected regressions, putting the models into the spreadsheet, and letting the spreadsheet compute the results. The spreadsheet can even be allowed to select the best model and make the forecast prediction.

SUMMARY

Decision Support Systems does not require a sophisticated software package. DSS should not be considered a thing of the future that will not be successful until Artificial Intelligence (AI) or advanced expert systems become available. More sophisticated computer systems are available. Many of my students have developed analysis tools to do most of the functions mentioned in this article. This software is available to those interested, upon request. As we have seen, the software for DSS evaluations is available using the basic tools that come with most microcomputers.