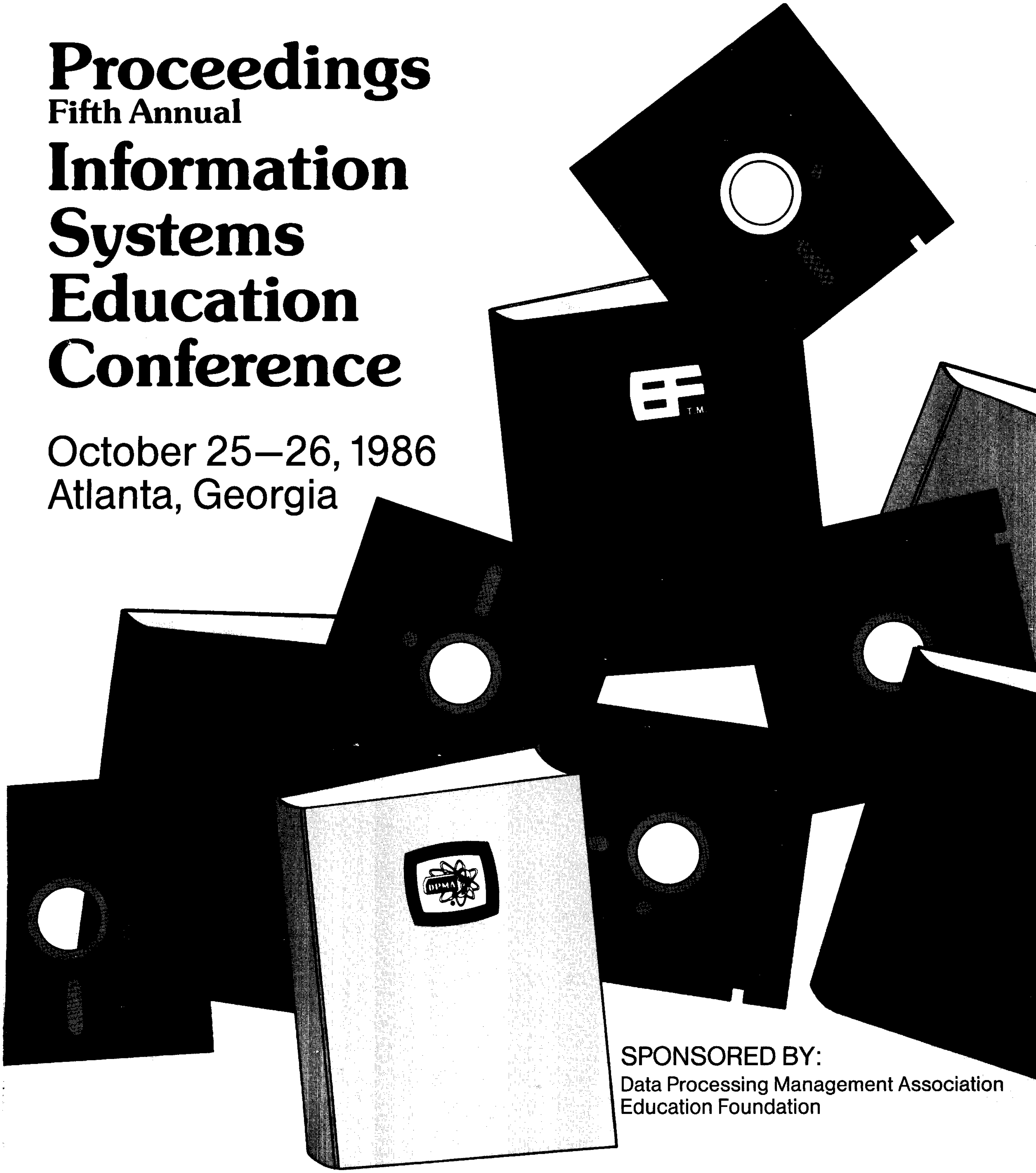
ISECON '86

Proceedings
Fifth Annual
Information Systems Education Conference

October 25–26, 1986
Atlanta, Georgia



SPONSORED BY:
Data Processing Management Association
Education Foundation

ISECON '86

Proceedings
Fifth Annual
Information Systems Education Conference

WELCOME TO ISECON '86

by: Edward A. Otting, President
DPMA Education Foundation

Pick up a newspaper today and you just might read that we're now in the INFORMATION AGE. At first, that may be hard to understand. After all, there's no more information today than there ever was. Information exists whether we recognize it or not. Cave people probably never conducted a census, but census data still were there for the taking.

When you get down to it, though, one thing sets apart today's information from yesteryear's--value. Information is worth only what people make of it. It's like the old thought, "If a tree falls in the woods but no one's around to hear it, does it make a noise?" People give information meaning.

We're in the INFORMATION AGE because today's technology allows people to generate more uses for information. Technology lets us group information, route it, graph it, send it, receive it, fold, spindle and mutilate it. Best of all, technology lets us do all the above with the touch of a finger. This sounds almost magical, but as a person concerned with information systems education, you know too well there's no hocus pocus involved.

That's where ISECON '86 makes an impact. Magicians perform illusions; information professionals so effectively perform hard work and thought that to end users the results seem like magic. Users' expectations are endless. Just like a magician's audience, end users eagerly wait to see one feat topped by a better one. Information professionals must drive themselves to keep pace. Each passing day redefines effective information management. Information professionals need the ability to deal with new situations on what can seem a day-to-day basis. Proper education provides that ability.

ISECON '86 is a gathering of minds that will pass on knowledge about teaching information systems students. Attendees bring from all corners their experiences and share them. Bring their own ideas and reveal them for consideration. Hear others' ideas, and contemplate how they might apply those concepts. Rapid technological advances mean one thing for the typical information systems educator--no one knows everything, but everyone knows something. ISECON '86 brings together outstanding individuals in a setting that promotes discourse and analysis. It's a unique event and it brings unique results to the thousands of people learning to become information systems professionals.

The INFORMATION AGE will exist only as long as there are professionals who can grab hold of raw data and put it to good use, thus give value where there previously was none. Times will change and information managers must change with them. ISECON '86 provides the first step in equipping tomorrow's information professionals with the ability to succeed in the present and adapt to the future. The value of tomorrow's information will be the yield of investments made here.

ISECON '86

EXECUTIVE COMMITTEE

CONFERENCE CHAIRPERSON

HERBERT REBHUN

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

PROGRAM CHAIRPERSON

KATHY BRITTAIN WHITE

Associate Professor of IS
University of North Carolina-Greensboro

MARKETING CHAIRPERSON

RONALD J. KIZIOR

Assistant Professor of Management Science
Loyola University of Chicago

EXHIBITS CHAIRPERSON

MS. SHOHREH HASHEMI

Assistant Professor of IS
University of Houston-Downtown

FACILITIES CHAIRPERSON

ED NEMETH

Dean of Computer Information Systems
DeVry Institute of Technology

PLACEMENT CHAIRPERSON

WILLIAM MITCHELL

Professor & Chairman of
the Department of Computer Science

EX-OFFICIO

GEORGE FOWLER

Associate Professor of MIS
Texas A&M University

TABLE OF CONTENTS

Fundamental Principles of Information Systems Development

Harlan D. Mills
IBM Corporation and
University of Maryland

Richard C. Linger
IBM Corporation

Alan R. Hevner
University of Maryland

ABSTRACT: Information systems is a new field in which we are just discovering the fundamental principles and methodologies. Four principles of information systems development are presented and analyzed, namely: Referential Transparency, System Completeness, Data Decentralization, and Modular Reusability. Box Structure Methodology applies these principles, in more technical terms, to information systems development.

1. PRINCIPLES AND APPEARANCES IN INFORMATION SYSTEMS DEVELOPMENT

Information systems play increasingly critical roles in business operations, and span the business process in complexity. Business operations are old, and thereby benefit from a long history of lessons learned, natural selection of effective ideas and processes, and a stable university curriculum [Drucker 1973]. In contrast, the subject of information systems is new -- less than a human generation old -- with most of its history ahead of it. Such new subjects are necessarily defined more by apprenticeship and appearances than by systematic principles and procedures. But it is time for information systems to move from appearances to principles for greater flexibility of application and increased effectiveness.

We can teach trick dogs the appearance of arithmetic, but they cannot use it off stage. But when we teach children the principles of arithmetic, they use it the rest of their lives in problems their teachers have never imagined. Even so, it has taken human society thousands of years to develop and simplify the principles of arithmetic for teaching to children. For example, long division was beyond the wisest of the ancient Greeks, and was a graduate university subject in the Renaissance.

We believe that certain fundamental principles of information systems are discernible now, and discuss four of them in this paper, namely, principles of:

Referential Transparency
System Completeness
Data Decentralization
Modular Reusability

A methodology for information systems development should support these four fundamental principles. The **Box Structure Methodology,** as described in [Mills 1986], satisfies this requirement. The correct use of box structures in information systems development enforces a complete set of detailed analysis and design principles. The four fundamental principles discussed in this paper have direct, more technical forms in the Box Structure Methodology. The correspondence is as follows (by number):

Fundamental Principles

Information Systems

(1) Referential Transparency
(2) System Completeness
(3) Data Decentralization
(4) Modular Reusability

Box Structure Methodology

(1) Black Box Replacement
(2) Transaction Closure
(3) State Migration
(4) Common Services

2. REFERENTIAL TRANSPARENCY

Principle of Referential Transparency -- In the delegation of any system part for design and implementation, all requirements should be specified explicitly, so that no further communication or coordination is required to complete the system part.

At first glance, the principle of referential transparency may seem impossible or impractical. Most commercial system methodologies do not require it, and opt instead for easier-to-explain diagrams or structures that subsequently require communication and coordination between separate system parts right down to their detailed implementation.

In fact, such a principle is indeed impossible, unless the right kinds of system parts are identified. Modern developments in computer science and software engineering identify the kinds of parts required to make the principle possible. The specification of these parts must be defined at the stimulus/response level, and account for the effect of any internal data storage possible in the part. General methods that use ideas such as structure charts or data flow diagrams lose vital information and thus make referential transparency impossible. It takes the right kinds of system parts to defer details without losing them.

Whether such a principle, made possible in theory by correct logical definitions, is practical depends on its appropriateness to information systems development processes. Human society has had over thirty years to discover how difficult it is to develop and maintain relevant, reliable, and modifiable information systems, all the while seeing the business stakes rise to incredible proportions. The value of intellectual and management control of information systems makes referential transparency practical, indeed, for those who know how to achieve it. The up-front work is more demanding, but the alternative of optimistically setting out to implement system parts without referential transparency is an expensive one, possibly a fatal one, for a business today.

The principle of referential transparency provides a crisp discipline for management delegation and assignment of responsibility. The lack of referential transparency can lead to management nightmares where nothing works and no one is to blame. When designers and implementers are required to communicate and coordinate about details of separate parts after their assignment of responsibilities, gamesmanship becomes an important part of a day's work, in addition to system development. It's only sensible, with all defined responsibilities, to cover one's bets and tracks with activities and documents designed more to protect than to illuminate.

Even with the best of intentions, extensive communication and coordination about design and implementation details opens up that many more opportunities for misunderstandings and errors. Such errors are always written off as human fallibilities (nobody is perfect), but errors of unnecessary communication and coordination should be charged to the methodologies that require them, not to the people forced to do the unnecessary communication and coordination.

In the Box Structure Methodology the principle of black box replacement incorporates the concept of referential transparency.

> **Principle of Black Box Replacement --** Any correct implementation of a correctly specified black box will be satisfactory as a system part. There are two parties that are accountable, the specifier and the implementor. A correct implementation to an incorrect specification puts the specification at fault; an incorrect implementation to a correct specification puts the implementor at fault.

The principle of black box replacement is key in system development for the accountability and flexibility it provides management. A black box can be designed and implemented independently of its surroundings in a system, so accountablity is perceived in delegation. In flexibility, a black box can be redesigned with different state machines and clear boxes. As long as the new black box behavior is identical to the original, the rest of the system will operate exactly as before. Such black box replacement may be required or desirable for purposes of better performance, changing hardware, or even changing from manual to automatic operations.

3. SYSTEM COMPLETENESS

> **Principle of System Completeness --** The transactions of a system part should be sufficient for the acquisition and preservation of all its system data, and its system data should be sufficient for the completion of all its transactions. In particular, system integrity should be considered as well as user function in achieving system completeness.

The principle of system completeness can forestall many surprises and after-thoughts in specifying and designing systems. A common mistake for amateur (and not so amateur) analysts and designers is concentrating so much on primary user function that the secondary functions to make user functions available and reliable become awkward or impossible. For example, if system security or recovery requirements are not identified up-front, then an ideal user system (imagined in a perfect world of hardware and people) may end up with data structures that make security or recovery difficult or impossible. So functions provided for security and recovery need to be defined as carefully and as early as user functions.

The principle of system completeness defines a systematic, iterative specification process, in ensuring that a sufficient set of transactions is identified to acquire and preserve a sufficient set of system data. The iteration begins with the transactions for the primary users, and the system data needed for those transactions, then considers the transactions required for the acquisition and preservation of that system data, then identifies the system data needed for those transactions, and so on. Eventually, no more transactions will be required in an iteration, and system completeness will have been achieved.

The concept of system integrity plays a special role in system completeness. System completeness assuming perfect hardware and people is not enough; many transactions can only be defined once specific hardware and people are identified for system use. For example, an information system using an operating system with automatic checkpoint and restart facilities won't need checkpoint and restart transactions, but one without them will. The problem of system security gives a classic example. Many operating systems and database systems in wide use today cannot be retrofitted for high level multilevel security because they were conceived and specified before such security requirements were identified.

In simplest terms, information systems integrity is the property of the system fulfilling its function, while handling all of the system issues inherent in its implementation. For example, systems are expected to be correct, secure, reliable, and capable of handling their applications. These requirements may not be explicitly stated by managers, users, or operators, but it is clear that the designed system must have provisions for such properties. Questions of system integrity are largely independent of the function of the system, but are dependent on its means of implementation, manual or automatic. Manual implementations must deal with the fallibilities of people, beginning with their very absence or presence (so back-up personnel may be required), that include limited ability and speed in doing arithmetic, limited memory capability for detailed facts, lapses in performance from fatigue or boredom, and so on. Automatic implementation must deal with the fallibilities of computer hardware and software, beginning with their total lack of common sense, that include limited processing and storage capabilities (much larger than for people, but still limited), hardware and software errors, security weaknesses, and so on.

In the Box Structure Methodology the principle of transaction closure incorporates the concept of system completeness. In a state machine, a transaction is defined by a sequence of stimuli that form a user request of the system, and the response therefrom.

> **Principle of Transaction Closure --** The transactions of a state machine must be sufficient for the acquisition and preservation of all its state data, and its state data must be sufficient for the execution of all its transactions.

The process of transaction closure is essential in the development of a top level black box for any system. A useful beginning of this search for a top level black box begins with the most obvious users of the system to be, but seldom ends there. These most obvious users often interact with the system daily, even minute by minute in entering and accessing data -- for example, a clerk in an airline reservations system. But usually, the data they use are provided in part by other users that enter and access data less frequently - for example, those entering flight availability information. And other users even more distant from the obvious users enter and access data even less frequently -- for example, users who add route schedule information. All the while, an entirely different group, the operators of the system, is entering and accessing system control data that affects the users in terms of more or less access to the system because of limited capacity or availability.

The top level black box must accommodate the transactions of all these users and operators, not just the most obvious ones. A cross check can be made between the top level black box and its top level state machine. Every item of data in the top level state must have been loaded with the original system or acquired by previous black box transactions. Are there any items not so loaded or acquired? It is easy, in concentrating on one set of transactions, to assume the existence of data to carry them out. A close comprehensive scrutiny of these needed data items can discover such unwarranted assumptions early.

4. DATA DECENTRALIZATION

> **Principle of Data Decentralization --** System data should be decentralized to the smallest system parts that do not require duplicating data updates. If, for geographic or security reasons, system data should be decentralized to smaller system parts, the system should be redesigned to ensure correctly duplicated data updates.

The principle of data decentralization eliminates the need for instant decisions (often faulty) about how data should be structured and how it should be stored in a system. Instead, it permits the definition of system data at a conceptual level, and its concrete form and location to be worked out interactively with the system design and decomposition into system parts. As better design ideas emerge, system data can be relocated effectively to accommodate such ideas, all the while maintaining correct function as required in the system transactions.

When system data needs to be decentralized to smaller system parts than allowed by the principle of data decentralization, the smallest system defined by this part must be redesigned to accommodate correct duplicate updating. In this case, it is a different system, and should be recognized as such from the outset. The problem of incorrect updating of duplicated data is a well-known burden of faulty system designs.

System data in a box structure hierarchy is distributed into the states of its component box structures. State migration then becomes an important principle in the Box Structure Methodology.

> **Principle of State Migration** -- State data should be migrated to the lowest level state machine possible without duplicating updates. If, for reasons of geography or security, state data should be migrated to lower levels, the box structure should be redesigned.

State migration through the box structure hierarchy is a powerful tool in managing system development. It permits the placement of state data at the most effective level for its use. Downward migration is possible when black boxes are identified in a higher level clear box; state data used solely within one black box can be migrated to the next lower level of the box structure hierarchy. The isolation of state data at its essential level in the system hierarchy provides important criteria for the design of database systems and file systems. Upward migration is possible when duplicate state data is updated in identical ways in several places in the hierarchy. This data can be migrated up to the closest common parent subsystem for consistent update at one location.

5. MODULAR REUSABILITY

> **Principle of Modular Reusability** -- System parts with multiple uses should be considered for definition as reusable modules. A corollary principle is to create as many opportunities as possible for reusability within and between system parts.

Operating systems, data management and database systems, network and terminal control systems are all illustrations of common services between systems. It is axiomatic in today's technology to seek as much reuse of common services as possible to multiply productivity and increase reliability. These common services need to satisfy the principle of referential transparency in their use, so their specifications are as important as their implementations. On a smaller scale, effective system design seeks and creates commonality for services and identifies system parts for widespread multiple uses within a system.

The principle of modular reusability is reflected in the Box Structure Methodology as the principle of common services.

> **Principle of Common Services** -- Black boxes with multiple uses in a box structure should be considered for definition as common services.

When several black boxes of a clear box expansion access or alter a common state part, it is generally inadvisable to migrate the state part to those lower levels. But it may be advisable to define a new box structure to provide access to or to alter this common state part for these several black boxes. Such a new box structure must be invoked in the clear box expansions of these black boxes. This new box structure thereby provides a common service to these several black boxes. Such a common service structure in effect encapsulates a state part, by providing the only means for accessing or altering it in the box structure hierarchy.

State encapsulation requires a new box structure whose state will contain the common state part, and whose transactions will provide common access to that state part for multiple users. In essence, state encapsulation permits state migration to be carried out in another form, with the provision that the only possible access to the migrated state is by invoking transactions of the new box structure that encapsulates it.

Common service box structures are ubiquitous in information systems. For example, any database system behaves as a common service box structure to the people and programs that use it. In simple illustration, consider a clear box expansion of a master file update state machine. Such a clear box would contain a number of black boxes which operate on the master file, for example, to open, close, read, and write the file, as well as black boxes to access transaction files, directory and authorization information, etc. The master file of the clear box state cannot be migrated to the lower level black boxes without duplication. However, the master file can be encapsulated, without duplication, in a new box structure that provides the required transactions to open, close, read, and write the file. This box structure can be designed to ensure the integrity of the master file and all access directed to it. In fact, when the master file is migrated to this common service, it is protected from faulty access by the box structure in an effective way.

6. PRINCIPLES AND APPEARANCES IN DEVELOPMENT METHODOLOGIES

Many commercial development methodologies embody the appearances but not the principles of information system development. In illustration, consider the principle of referential transparency applied to the common-sense techniques of structure charts [Yourdon 1978] and data flow diagrams [DeMarco 1979].

Structure charts are convenient and easily understood summarizations of function decomposition, a good intuitive first step in a divide-and-conquer strategy. However, in order to achieve referential transparency in function decomposition, a second step is required. That second step is to reconnect what is to be divided in such a way to guarantee that the function has indeed been conquered, namely to realize a divide, reconnect, and conquer strategy.

What is missing in structure charts is the precise connections between the parent and children in each decomposition in terms of execution control and data communication. Even detailed function statements (for example, to "accept and check input", "determine patient status", "format and print report", etc.) are not enough to provide referential transparency when such statements require communication and coordination between people implementing the children, rather than only between people responsible for one child and the parent. For example, when the person implementing "accept and check input" must ask the person implementing "check patient status" how often input is to be accepted and where to put it for further use, referential transparency has been lost. Instead, the entire specification of "accept and check input" should come from the designer of the parent function, who should also be responsible that the execution control of the parent, with any and all correct implementations of the specification, will provide correct parent function.

In summary, for a function decomposition into n parts at the next level there are n + 1 responsible parties. The parent designer is responsible for the parent control, for n specifications, and for the correctness of the parent function using that control with correct implementations of the n specifications. The children implementors are each responsible for correct implementation of their specifications with no dependencies among themselves.

Data flow diagrams are descriptive summarizations of data flow (arcs) among data transformations (nodes). However, as with structure charts, data flow diagrams lack referential transparency in decompositions into more detailed diagrams. Data flow diagrams do not define the precise control and data communication between parent and children in all possible circumstances required for independent elaboration of the detailed data flow of the children. As a result, decompositions can be carried out only at the expense of further communication among people defining the data flow of the parent and its children.

For example, imagine a customer order data item that flows through transformations to "validate order", "fill order", and "ship order". That seems like a common sense data flow process, but consider the decomposition of "fill order". Suppose the order cannot be filled from inventory, say because some items are out of stock. In this case, the unfilled order should not flow to "ship order", but that is the only possible flow in the parent diagram. So additional communication between people responsible for "fill order" and "ship order" is required, to account for the case of unfilled orders.

Alternately, the problem could be seen as a failure to fully define the data flow at the parent level, which could be modified to route unfilled orders to a new "back order" node. But this modification itself leads to new problems of communication between people responsible for detailing the data flows of "back order" and "fill order," in defining precise control of the back order filling process.

In short, the problem is not one of design, but of methodology, in the absence of referential transparency for control of data communication among nodes in data flow diagrams. Data flow diagrams are a strategy for dividing, but not conquering, because the vital reconnections are missing.

Of course, even in a design methodology that provides referential transparency in decompositions, there is no law against good communication among people at all levels of decomposition. Better ideas discovered in decomposition activities can always be communicated up, leading to better divide, reconnect, and conquer strategies at higher levels, which can in turn be communicated back down with referential transparency -- but with delegated responsibilities always intact.

References

[DeMarco 1979] DeMarco, Tom, Structured Analysis and System Specification, Yourdon Press, 1979.

[Drucker 1973] Drucker, Peter F., Management: Tasks, Responsibilities, Practice, Harper and Row, 1973.

[Mills 1986] Mills, H. D. , Linger, R. C., and Hevner, A. R. Principles of Information Systems Analysis and Design, Academic Press, 1986.

[Yourdon 1978] Yourdon, E. and Constantine, L. Structured Design: Fundamentals of a Discipline of Computer, Program and Systems Design, Yourdon Press, 1978.

Managing Information Systems Development

Harlan D. Mills
IBM Corporation and
Univ. of Maryland

Richard C. Linger
IBM Corporation

Alan R. Hevner
Univ. Of Maryland

Abstract: Information systems development requires a flexible paradigm for generating a development plan of investigation, specification, and implementation activities. The development process can be based on the Box Structure Methodology for information systems analysis and design. This methodology provides extensive management control facilities. A system development spiral accurately represents the actual work in system development.

1. MANAGING WITH THE BOX STRUCTURE METHODOLOGY

Information systems development should be managed as a systematic business engineering discipline. This discipline can be based on rigorous principles of computer science and engineering found in the Box Structure Methodology [Mills 1986a] applied directly to business problems. The primary benefit is effective management control over information system development.

The main problem in system development, as identified by Brooks [Brooks 1975], is achieving conceptual integrity, namely, the means for intellectual control over the mass of details in information systems. The basis for intellectual control in software was discovered by Dijkstra and others with the principles of structured programming [Dahl 1972, Dijkstra 1976]. The mathematical foundations for structured programming [Linger 1979] provide principles for managing software development as discussed in [Mills 1983]. In a similar way, the Box Structure Methodology [Mills 1985, Mills 1986b] provides new principles for performing information systems development.

Information systems are more and more an integral part of business operations, and subject to business pressures. Faced with such business pressures, it is easy to fall into a harum-scarum, disorderly mode of operation. What is needed in information systems development in real business environments are management principles to balance urgent business needs with requirements for systematic work. Such principles are not new in business and management. They involve a spectrum of short range to long range planning [Keen 1978]. Long range plans deal with fundamental business objectives and trends; short range plans deal with near term needs and account for current conditions.

The Box Structure Methodology provides a foundation for management principles in information system development. It introduces three basic structures, namely **black box**, **state machine**, and **clear box**, that can be nested over and over in a hierarchical system structure. They provide three views of the same information system or any of its subsystems.

The **black box** gives an external view of a system or subsystem that accepts stimuli, and for each stimulus, it produces a response before accepting the next stimulus. A black box could be a calculator, a personal computer, an information system, or even a manual work procedure that accepts stimuli from the environment and produces responses one by one. As the name implies, a black box description of a system omits all details of internal structure and operations and deals solely with the behavior that is visible to its user in terms of stimuli and responses. Any black box response is uniquely determined by the system's stimulus history.

The **state machine** gives an intermediate system view that defines an internal system state, namely the data stored from stimulus to stimulus. It can be established mathematically that every system described by a black box has a state machine description.

The machine of a state machine is a black box that accepts as its stimulus both the external stimulus and the internal state and produces as a response both the external response and a new internal state which replaces the old state. The role of the state machine is to open up the black box description of a system one step by making its data visible.

The **clear box**, as the name suggests, opens up the state machine description of a system one more step in an internal view that describes the system processing of the stimulus and state (stored data). The processing can be described in terms of sequence, alternation, iteration, and concurrent structures. The clear box can contain embedded black boxes that represent subsystems yet to be elaborated.

At this point, a hierarchical, top-down description can be repeated for each of the embedded black boxes at the next lower level of description. Each black box is described by a state machine, then by a clear box containing even smaller black boxes, and so on.

These three views represent an increasing order of internal system detail. The black box describes the system from a user view. The user view is foremost since the objective of business systems is to provide user services. The state machine adds the consideration of system data (state) and its manipulation (machine). The clear box completes the description by adding internal processing details of the machine and recognizing embedded subsystems (black boxes). Describing each subsystem in these increasingly detailed views provides an internal consistency that is essential in developing and managing systems. The data structures must be consistent with the user view, and the processing structures must be consistent with the data structures.

The box structure hierarchy that results from the analysis and design of an information system is a powerful management tool for delegation of responsibility and control of systems development. In particular, the principle of referential transparency [Mills 1986b] can be used to management benefit in a box structured system development.

2. THE SYSTEM DEVELOPMENT PROCESS

In information systems development the use of the Box Structure Methodology is embodied in the definition of a set of limited, time phased **activities** to decompose and manage the various kinds of work required. A **development plan** is constructed that defines and schedules the specific activities needed to address a specific problem. The development plan represents long range planning for information system development; the activity plans represent short range planning. As each activity is completed, the entire development plan should be updated to account for the current situation.

Although the activities of a development plan are always specific to a particular system development problem, they can be categorized into three general classes, **investigation**, **specification**, and **implementation**. An investigation is a fact-finding, exploratory study, usually to assess the feasibility of an information system. A specification is more focused to define a specific information system and its benefits to the business. An implementation converts a specification into an operational system.

System development requires focused, creative work carried out with strict discipline. It requires mental inspiration and mental perspiration in the usual ratio of 5% inspiration to 95% perspiration. This need for both creativity and discipline calls for a rigorous management process to define short term and long term objectives, measure progress, introduce mid-course corrections, and insure completion and success in system development.

The system development process provides a paradigm for generating system development activities of investigation, specification, or implementation, based on:

(1) A development plan consisting of a time phased set of planned activities to meet a specific business need.

(2) At each completed activity, a development plan update to account for progress made, lessons learned, and changes in the business need.

Typically, a development plan is the joint product of business management and system development management. Frequently, the initial development plan is quite general, beginning with an investigation whose primary purpose is to recommend a more definitive development plan.

The system development process is recorded in libraries that contain information used and generated by the development activities. The following four libraries are needed for development:

Management Library. The management library holds the development plan and other information needed by the development team to control and support the development effort. Control information includes schedules, day-to-day correspondence, and budgets. Support information includes the documentation provided to the business management, operators, and users, such as system proposals, feasibility studies, and review documents.

Analysis Library. The analysis library documents the analysis performed to create the new system. The box structure methodology emphasizes the use of box structure diagrams as a creative, flexible tool for analysis. This library is principally for communication among the development team and with the business environment during information system development.

Design Library. The developing system design is recorded in this library in a formal box structure design language. The design library is used by implementation activities as the basis for the system. The design library is updated to contain any design changes that may occur during implementation.

Evaluation Library. Evaluations of development results are recorded in this library.

Examples of evaluation include design verification through box structure analysis, software testing, and system testing. These results serve as an information resource for proposals and review documents that are contained in the management library.

These four libraries are at the center of the system development process.

3. THE SYSTEM DEVELOPMENT SPIRAL

Many current methods of information systems development reflect intuitive appearances rather than sound principles. One of the obvious appearances in information systems is the system development life cycle. It is certainly apparent that information systems go through various stages of conception, specification, design, implementation, operation, maintenance, modification, and so on. However, although these terms are suggestive, real information systems do not pass through these stages in any simple or straightforward way.

If information systems were developed for their own intrinsic worth, by people with infinite knowledge and intelligence, given unlimited time and budgets, such an information system life cycle might be possible and sensible. But, information systems are developed for business purposes with limited time and budgets by real people, often under conditions that are far from ideal because of business pressures.

If a competitive hotel chain announces a new reservation system, the business needs a quick response with whatever system that can be put into operation, not a system developed to an orderly time table that arrives too late to save the business. If a banking law changes and more immediate financial information can save interest rates, every day spent in a fixed life cycle development cycle is money lost.

In contrast to a fixed life cycle, the box structured system development process is defined by a set of time-phased activities that are initiated and managed dynamically based on the outcome of previous activities in the development. This progression of activities is conveniently represented in a flexible **system development spiral** that reflects the actual progress of a development effort.

The time-phased set of activities in a spiral can be strictly sequential, or may have concurrent parts. If a development is sequential, it can be pictured, in prospect or retrospect, as shown in Figure 1. In this example, the activity sequence is a straightforward progression of

Investigation
Specification
Implementation

with a management approval to enter each activity and to end the entire development. Such a progression for developing a system is an ideal, but is not necessarily possible or even desirable.

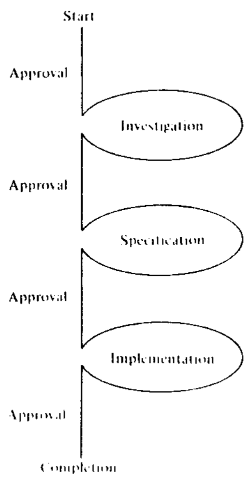


Figure 1: Example System Development Spiral

It may not be possible because the business problem is too complex and needs several investigation activities to arrive at a solution. It may not be possible because the system development problem is too complex and needs several specification/implementation activities in an incremental development. It may not be desirable because the business problem is too acute and a less-than-best implementation is called for as soon as possible. It may not be desirable because the happy outcome of the first investigation activity is the discovery of an existing implementation to meet the business need.

If a development is concurrent, it can be pictured in a network of spirals, as in the example of Figure 2. In this network, activity dependencies are shown by the approval lines ("A" lines here). For example, Investigation 1 enables both Specification 1 and Investigation 2, while both Implementation 1 and Specification 2 must be completed before Implementation 2 can be started. The specific network pictured might, for example, represent the concurrent development of a database system (Implementation 1) and an application system (Implementation 2) that uses it.

The time-phased activities of a development plan will be expressed in calendar time, often tied to business events. In fact, many times the system development itself will influence these events. For example, a system to improve customer service may be advertised, and so require advertising copy and commercials to be developed and placed ahead of time, customer service personnel to be trained, equipment to be purchased, and so on. Needless to say, if the development is late, the business costs may be substantial and way out of proportion to the cost of the development overrun. More and more, information systems are at the heart of businesses and their competitive positions, so the stakes for effective information systems development to calendar schedules can be very large.

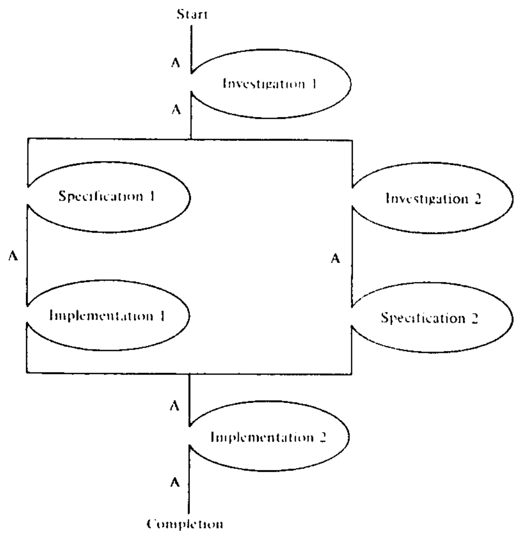


Figure 2. Example System Development Spiral Network

Information systems development can be successful even though no system is developed, and can be a failure even though a system is developed. There are several ways a development can be successful without developing a system:

(1) An investigation activity can discover an existing system to meet the business need, saving the cost of specification and implementation.

(2) An investigation activity can discover how to improve the existing business process so much that a new system cannot be cost justified.

(3) A specification activity can tailor the needs of the business to a form such that a specialized vendor can supply a system at greatly reduced cost.

An information system can be a technical success and still be a business failure in several ways:

(1) The system addresses the wrong problem because of insufficient investigation and understanding of the real business process.

(2) The system addresses the right problem, but is too hard to use because of insufficient investigation of user skills.

(3) The system addresses the right problem and is easy to use, but cannot be kept on the air because of operator or integrity problems.

In short, there are any number of ways an information system development can succeed or fail. They are rooted in the business and the final judge of success is the business. For that reason the system development process must be flexible and responsive to the needs of the business.

4. ACTIVITY MANAGEMENT

The system development process generates limited, time-phased activities of investigation, specification and implementation that must be managed. The stages of **planning, performance**, and **evaluation** in each activity define an orderly process for this management. The box structure methodology provides a great deal of commonality across these activities for the analysis and design work that is required. The management problems are also very similar. As the names might imply, the most challenging stages for management are planning and evaluation, while the performance stage is most challenging for technical professionals.

PLANNING

There are three basic results from the planning stage of any activity:

(1) **Activity Objective.** A clear statement of what the activity is to produce.

(2) **Activity Statement of Work** (SOW). A clear statement of how the activity will achieve its objective.

(3) **Activity Schedule.** A clear assignment of work items in the SOW to professionals, together with completion dates which each of the professionals agree to.

With such a plan, the entire development team understands the objectives, statement of work, and the individual responsibilities for making the work objectives and schedule good. Such a plan not only requires the agreement of the professionals, but also requires their direct participation in the planning process. But the planning process must be led by managers to address the proper questions and problems for the activity in the overall development plan.

The outputs of all previous activity loops in the development spiral and the feedback from the business environment combine to help the management and the development team decide what type of activity is required next. The first task is to define an activity objective and to derive a plan for meeting that objective.

Activity loops can be scheduled with Gantt charts and project networks. Activity schedules should be as detailed and specific as possible, since they are the primary means of management control. At the same time, the overall system development schedule must be updated to reflect the resources allocated to the activity.

All of the planning information, objectives, SOW, and schedules are included in a formal **activity proposal** that is presented to the managers of the development. The proposal is accepted, modified, or rejected based upon management analysis. The eventual acceptance of an activity proposal marks the end of the planning stage and the beginning of the activity performance. The accepted proposal is stored in the management library for reference.

PERFORMANCE

If plans are well made, performance is focused and predictable. The management job in performance is to assess and track progress against the SOW and schedules, to identify unexpected problems and help professionals decide how to meet them, and to identify unexpected windfalls in solutions that can free up people, or unexpected problems that will require additional resources to solve. It is here that good understandings and agreements on assignments and schedules pay off.

First, each team member understands his/her role in the activity, and the need for completing the work to schedule. In contrast, a common misunderstanding between managers and professionals pits the "manager's schedules" against the "professional's design." Such disagreements should be ironed out in planning, not late in performance. A common set of system values from the box structure methodology permits managers and professionals to communicate effectively and alleviate such disagreements. For example, a professional may be reluctant to adapt a less than the best system to a reservation system because a better one can be built; but if the professional understands that a quick and dirty system means the literal survival of the business, there will be little reluctance for a whole-hearted effort to get the quick and dirty system on the air.

Second, good schedules and a common understanding of box structure methodology make progress assessments and tracking more accurate and more rewarding. When managers can recognize good and timely work, they can acknowledge it privately and publicly, and when warranted, arrange for awards for extra performance. A major morale problem among high-performance professionals is just the fact that good work is often not recognized, and mediocre work by others is as well rewarded as their own. Good progress assessments and tracking of well understood assignments go a long way in recognizing good work.

EVALUATION

Evaluation is both a closing out of one activity and a basis for selecting and commencing one or more following activities. The objectives and results of performance can be compared and related to the business and its situation. Even if objectives are not met, the lessons learned may be useful. If the objectives are met, so much the better and the expected next activities can be initiated. In particular, the evaluation stage is the point where the development plan for future activities can be assessed and modified.

These activities and stages can be organized in table form, as shown in Table 1, that indicate typical tasks in system development. A detailed discussion of these tasks is found in [Mills 86a].

5. **CONCLUSIONS**

The Box Structure Methodology provides a structure for management control of information systems development. The system development process is a management paradigm for defining and scheduling work in investigation, specification and implementation activities. The construction of a system development spiral reflects the actual way work unfolds in a development. The system development process must be viewed as a flexible, dynamic process with well-defined management checkpoints. These principles are not well represented in the appearances of a simple system life cycle; but good activity plans in a system development spiral permit work to be focused and predictable.

References

[Brooks 1975] Brooks, F. P. The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Reading, MA, 1975.

[Dahl 1972] Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R. Structured Programming, Academic Press, New York, 1972.

[Dijkstra 1976] Dijkstra, E. W. A Discipline of Programming, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[Keen 1978] Keen, P. G. W. and Scott Morton, M. S. Decision Support Systems: An Organizational Perspective, Addison-Wesley, Reading, MA, 1978.

[Linger 1979] Linger, R. C., Mills, H. D., and Witt, B. I. Structure Programming: Theory and Practice, Addison-Wesley, Reading, MA, 1979.

[Mills 1983] Mills, H. D. Software Productivity, Little, Brown, Boston, MA, 1983.

[Mills 1985] Mills, H. D. "A New Course in Information Systems Development," Proceedings ISECON '85, Houston, 1985.

[Mills 1986a] Mills, H. D., Linger, R. C., and Hevner, A. R. Principles of Information Systems Analysis and Design, Academic Press, Orlando, FL, 1986.

[Mills 1986b] Mills, H. D., Linger, R. C., and Hevner, A. R. "Fundamental Principles of Information Systems Development," Proceedings ISECON '86, Atlanta, 1986.

Table 1.
Stages in Activities: Typical Tasks

	Stages		
Activities	Planning	Performance	Evaluation
Investigation	Activity Objective Statement of Work Scheduling	Business Process and Objectives Requirements Analysis System Prototype	Feasibility Assessment Review and Acceptance Development Plan Update
Specification	Activity Objective Statement of Work Scheduling	Systems Analysis and Design Operations Analysis and Design	Design Verification Review and Acceptance Development Plan Update
Implementation	Activity Objective Statement of Work Scheduling	Resource Acquisition Systems Integration Operations Education	System Testing Review and Acceptance Development Plan Update

INFORMATION SYSTEM PLANNING: THE MISSING LINK IN THE CIS CURRICULUM

Engming Lin
Chang-Yang Lin
College of Business
Eastern Kentucky University
Richmond, Kentucky 40475

ABSTRACT

In many organizations the lack of information system planning has caused serious problems. This papaer suggests that systems analysis and design course in the undergraduate CIS curriculum be expanded to include the fundamental concept of information system planning.

I. INTRODUCTION

CIS '86, the updated version of the DPMA Model Curriculum for Undergraduate Computer Information Systems, was introduced at ISECON '85 held in Houston, Texas. Compared with the first DPMA Model Curriculum published in 1981, the new curriculum has many significant revisions of existing courses and several new courses that reflect the rapid changes that have taken place in the information-processing business during the 1980's. CIS students will definitely be benefited from the new curriculum.

Although the new curriculum has taken care of almost all the state-of-the-art techniques and the current development in business that will affect all areas of the DP profession, information system planning is not a topic in any CIS core course of CIS '86. CIS/86-18, Information Resource Planning and Management, does include the basic concept of information system planning and some planning techniques. However, it is a suggested elective course. Many students might graduate with a degree in CIS without knowing anything about information systems planning. This is not helpful to them because the information-system plan is the foundation for the development of information systems.

This paper proposes that information system planning be covered in a CIS core course. In this paper "information system planning" and "MIS planning" are used interchangeably.

II. WHY INFORMATION SYSTEM PLANNING?

Information system planning is not a new idea as the need for planning was recognized in the 1960s. Emphasizing the importance of MIS planning, Murdick and Ross stated the symptom of the lack of MIS planning in their book published in 1975:

> ... we have seen the development of "islands of mechanization" following unrelated starts in quick-payoff areas.... This patchwork approach has resulted in the development of unrelated and sometimes incompatible subsystems [5, p.247].

The symptom still exists in the 1980s. A 1984 survey of "problems with today's information systems in the healthcare industries," conducted by Health Data Analysis, Inc., shows that "lack of systems interfaces" was the number-two problem reported [2]. The lack of information system planning is obviously a very serious problem in business.

Planning is one of the main functions of management. Organizations that plan tend to achieve better results than organizations that do not. Information systems have to be planned too. Some textbooks for MIS or systems analysis and design define "planning" as the first phase in the information systems development life cycle. The main tasks included in this phase are recognizing the problem, defining the problem, setting system objectives, identifying system constraints, preparing the study project proposal, approving or disapproving the stydy project, and establishing the control mechanism [4]. Executing these tasks is essential for the success of the systems development project. As part of the systems development life cycle, these tasks are covered in CIS/86-5 of the DPMA Model Curriculum and are included in most systems analysis and design books. In order to avoid the creation of "incompatible systems," an MIS master plan must be in place before a systems development project starts. This MIS master plan is the results of the long-range MIS planning.

The facts, that all functional departments in an organization need information system support, and that information systems in the organization are not independent, are well understood. Because not all information systems can be developed and implemented concurrently and because every organization has limited resources, priorities must be set. An overall MIS master plan is needed to guide the initial information system development and consequent change. Davis and Olson [3] suggest that the MIS master plan should contain four major sections:

a. Information system goals, objectives, and architecture.
2. Inventory of current capabilities.
3. Forecast of developments affecting the plan.
4. The specific plan (i.e., system development projects).

Every well-managed organization has an overall business plan (or strategic plan) that reflects organizational goals, objectives, and strategies. Each goal, objective, and strategy in the plan can be analyzed for required information system support. These are then organized into information system goals, objectives, and strategies that are further used as the basis for planning individual information systems development projects. Each project then goes through a typical information system development life cycle. Figure 1 shows the information systems planning and development process. MIS planning is performed in three stages. Activities during the MIS planning phase link the systems development projects with the overall business plan. The three-stage model of information system planning process was developed by Bowman, Davis, and Wetherbe [1][3].

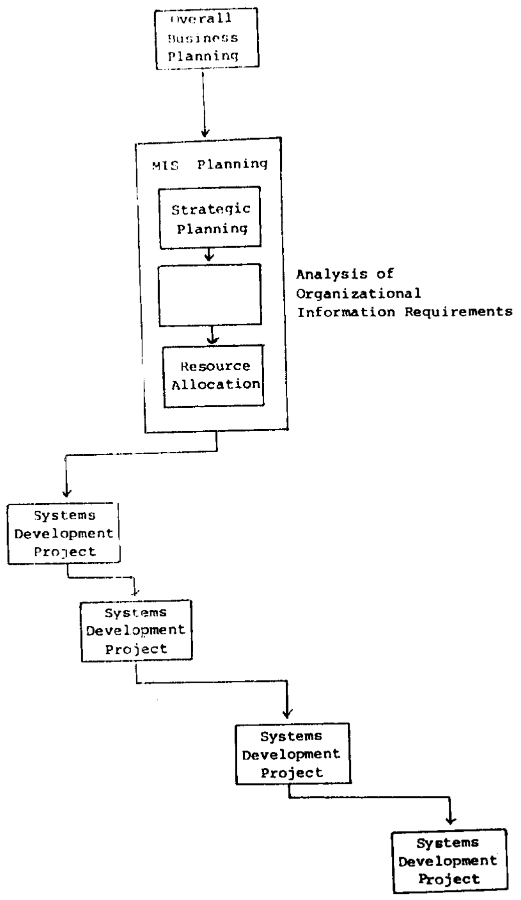
III. MIS PLANNING IN THE CIS CORE CURRICULUM

This paper does not propose an in-depth coverage of methodologies for information system planning in any CIS core course. The fundamental concept and major activities of MIS planning should be sufficient. The topic will not take more than 10% of a typical three-semester-hour course. Since the MIS master plan is the foundation for all information systems development projects, the ideal place for the topic is the very beginning of CIS/86-5, Systems Development Methodologies: A Survey.

The importance of information system planning cannot be overstated. The proposed change should strengthen the DPMA Model Curriculum for Undergraduate Computer Information Systems.

Figure 1.

INFORMATION SYSTEMS PLANNING AND DEVELOPMENT PROCESS


Overall Business Planning
MIS Planning
Strategic Planning
Analysis of Organizational Information Requirements
Resource Allocation
Systems Development Project
Systems Development Project
Systems Development Project
Systems Development Project

REFERENCES

1. B. Bowman, G. B. Davis, and J. C. Wetherbe, "Modeling for MIS," Datamation, July 1980, pp.155-162.

2. B. W. Childs, "Systems Interfaces (to be or not to be)," Healthcare Computing & Communications, Vol. 2 No. 10, October 1985, p.6 (Editorial).

3. G. B. Davis and M. H. Olson, Management Information Systems, Conceptual Foundations, Structure, and Development, 2nd ed., McGraw-Hill Book Co., New York, 1985.

4. R. Mcleod, Jr., Management Information Systems, 2nd ed., Science Research Associates, Inc., Chicago, 1983.

5. R. G. Murdick and J. E. Ross, Information Systems for Modern Management, 2nd ed., Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

CONTINUITY IN COMPUTER INFORMATION SYSTEM (CIS) COURSES

Merle P. Martin
Management Information Systems Department
School of Business Administration
California State University, Long Beach
1250 Bellflower Boulevard
Long Beach, CA 90840

ABSTRACT

A serious problem in the development and administration of a CIS curriculum is maintaining a logical continuity between courses. There must be a minimum of gaps or overlaps between successive offerings. This paper discusses the development and use of a CIS questionnaire which can assist curricula developers and administrators in assuring continuity in CIS courses.

INTRODUCTION

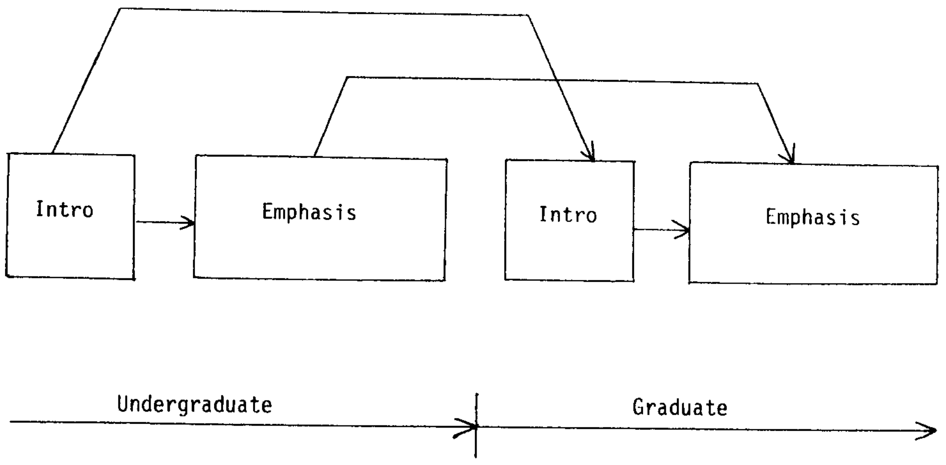
One measure of the effectiveness of any CIS curriculum is the continuity between the courses comprising that curriculum. What students are expected to learn in a prerequisite course should be assumed (and not retaught) in succeeding courses. In addition, assumed student knowledge at the beginning of any CIS course must be coupled with adequate coverage in prerequisite courses.

While these objectives are ideologically obvious, they are too often pragmatically violated. Differences in textbooks, professional knowledge and styles, and locations where CIS students accrue credit detract from desired continuity. What is required is a structured approach for detecting and then correcting CIS curricula discontinuities.

One such structured approach appears to be a questionnaire developed at Texas A & M University two years ago. The original purpose for this questionnaire was to establish a measurement instrument for computer knowledge. That instrument was used in a doctoral dissertation experimental design (3). However, the structure and method of development of this questionnaire would seem to have some potential for establishing CIS curricula continuity.

The questionnaire was developed according to two important assumptions. First, desired continuity between CIS courses should flow according to Figure 1. Second, description of course concepts and a general description of topics covered do not provide criteria sufficiently measurable to assess course continuity. Therefore, knowledge of specific CIS terms was utilized. While such terminology knowledge certainly does not

Figure 1: CIS Course Continuity



comprise all of what must be imparted in such courses, such knowledge can at least be agreed upon, and thus measured. This paper discusses the use of this questionnaire for assessing the continuity of a CIS curriculum. The discussion is organized in the following sections: (1) questionnaire development, (2) the prototype questionnaire, (3) final questionnaire administration, and (4) its use for assessing CIS continuity.

QUESTIONNAIRE DEVELOPMENT

The computer knowledge questionnaire was developed in the following sequence.

1. A list of 65 computer terms was extracted from current computer textbooks.

2. This list of terms was presented to the professors responsible for the:
   a) undergraduate introductory CIS course;
   b) graduate introductory CIS course (for which the undergraduate course was prerequisite); and
   c) graduate systems analysis and design course (for which the graduate introductory course was prerequisite).

3. Each of these MIS professors was asked to annotate for each of the computer terms whether or not that term was:
   a) expected to be known by students at the beginning of the course;
   b) expected to be known by students at the end of the course; or
   c) not expected to be known by students anytime during the course.

4. The participating professors were encouraged to add computer terms to those already listed.

5. Submissions from the professors were compared. Continuity discrepancies were removed from the list of candidate terms. For example, a particular term may have been expected to be known by undergraduate introductory students at the end of that course. However, the same term was not expected to be known by students of the graduate introductory CIS course at start of that course.

6. Of the remaining terms, 25 were selected for inclusion in a prototype questionnaire. The selection objective was to create a uniform and relatively continuous distribution across the knowledge spectrum of Figure 1. The terms included in the prototype are shown in Table 1.

7. Multiple choice questions were developed for each of the 25 selected computer terms. The questions were developed so that each contained one completely and one partially correct answer. Awarding points for partially correct answers contributed to the goal of continuity across the spectrum of test scores.

PROTOTYPE QUESTIONNAIRE

The prototype was administered to two sections of the undergraduate introduction course and one section of an undergraduate advanced COBOL course. The results included the following.

1. The distribution of scores met a chi-square goodness-of-fit test for a uniform distribution.

2. Three of the 25 questions had to be modified because of lack of adequate discrimination between students of the two prototype courses.

FINAL QUESTIONNAIRE ADMINISTRATION

The final questionnaire was administered the first week of the fall semester to 92 students of the graduate introductory CIS course and 48 students of the graduate systems analysis and design course. The resulting scores had the following characteristics:

1. The range of scores was 10 to 84 percent. Had the questionnaire been ad-

Table 1: CIS Terms Included in Prototype Questionnaire

bit	EDP	parity check
byte	encryption	peripheral equipment
CPU	hash total	protocol (handshaking)
check digit	index sequential	software
compilation	interactive programming	software package
CRT	left justify	tight code
database	MIS	memory
data element dictionary	modem	virtual
digit	nanosecond	

ministered at the end rather than the beginning of the semester, the top score would have undoubtedly been higher.

2. The range of scores met a chi-square goodness-of-fit test for a uniform distribution at $p < .05$.

3. The scores were stratified by course (Table 2). The difference in scores between courses was tested using chi-square, and found to be statistically significant at p less than .01.

questionnaire could be used to identify students overly qualified for specific CIS courses. The questionnaire could also be used to identify students seemingly deficient in prerequisite knowledge.

4. Teaching effectiveness. In large CIS teaching environments, there are often multiple sections of many CIS courses. These multiple sections are often taught, not only by several full-time faculty, but by graduate assistants and part-time faculty as well. Use of a standard CIS knowledge questionnaire would serve to identify classes of students whose instructors have failed to impart minimal knowledge.

Table 2: Questionnaire Scores by Course (Number of Students)

Score Grouping	Course		
	Introductory	Analysis/Design	Total
Lower Third	44	2	46
Middle Third	29	18	47
Upper Third	19	28	47
Total	92	48	140

ASSESSING CONTINUITY

The development of a computer knowledge questionnaire by the means described above provides several tools which can be used to assess and mandate continuity in a CIS curriculum. Among these tools are the following:

1. Potential Discrepancies List - identification of terms that students are:

   a) expected to know at the end of one course but not at the beginning of the next sequential course.

   b) expected to know at the beginning of one course, but for which there has been no associative expectations of learning during earlier courses.

2. CIS course standards - a published list of CIS definitions expected to be known at the beginning and end of each CIS course. This list should assist in reducing undesirable teaching variances between instructors, textbooks, and academic locations.

3. Student skills assessment. Often students are overly qualified for the course in which they are enrolled. This occurs through such means as failure to grant transfer credit for CIS courses taken elsewhere and accumulation of computer knowledge outside the classroom (e.g., home computer or work experience). A CIS knowledge

Another use for such a questionnaire must be mentioned, although it is not curricula focused. Cortney and DeSanctis (2) call for a common management game to be used for continuity of research in information systems. Stratification of users according to relative expertise (e.g., novice or expert) is also germain to such research (1). There is no practical, common instrument available to measure computer knowledge. The IBM Programming Examination measures aptitude rather than knowledge. The CDP examination is too lengthy and too varied.

An organization such as DPMA could develop such a knowledge questionnaire to match their model CIS curricula. The questionnaire could then be used, not only for achieving continuity in CIS programs, but for information system experimertation as well.

CONCLUSION

A problem in the development and administration of a CIS curriculum is maintaining a logical continuity between courses. Development of a computer knowledge questionnaire by means of the structured approach suggested in this paper will assist the CIS academician in addressing this problem. Such a developed questionnaire may also provide additional continuity in CIS research.

REFERENCES

1. Carey, T., "User Differences in Interface Design," Computer, November 1982.

2. Cortney, J., DeSanctis, G. and Kasper, G., "Continuity in MIS/DSS Laboratory Research: The Case for a Common Gaming Simulation," Decision Sciences, Vol. 14, July 1983.

3. Martin, M., "Designing Decision Support Systems for a Broad Range of User Experience,"(Texas A & M University Dissertation, December 1984).

IMPLEMENTATION OF A DEPARTMENT WIDE DESIGN STANDARD THROUGH AN INTRODUCTORY DESIGN COURSE

Ricky J. Barker
Assistant Professor

Jack L. Decker
Assistant Professor

Computer Information Sciences Department
Washburn University
1700 College Avenue
Topeka, Kansas 66621
913-295-6491

ABSTRACT

The Computer Information Sciences Department of Washburn University has adopted a departmental standard program documentation and design methodology. A new course has been developed which introduced this concept into the curriculum allowing many of the topics needed for our individual language courses to be combined and redunancy avoided. This paper discusses the advantages of such a course and how it has been implemented at Washburn University of Topeka, Kansas. The general course topics and method of instruction are also explained.

INTRODUCTION

The Computer Information Sciences Department of Washburn University has adopted a departmental standard program documentation and design methodology. The standard utilizes the Warnier-Orr (henceforth W-O) diagrams and Ken Orr and Associates (henceforth KOA) design methodology.

HISTORICAL JUSTIFICATION

A degree in Computer Information Systems was instituted at Washburn University in 1981. The degree is issued through the Computer Information Sciences Department. From the beginning the emphasis of the program has been the development of programmer/analysts. In addition, strong programming skills are also developed within the CIS curriculum at Washburn, with COBOL being the core language. As our curriculum developed, it followed a common path (the languages first) with COBOL and FORTRAN, then RPG II, Assembler, Advanced COBOL, BASIC, and RPG III. During this process we also developed more theory oriented CIS classes. A deficiency was noted concerning the language classes, namely that many topics covered are the same for each class and some could be more efficiently taught in an introductory course. We added an Introduction to EDP course (CIS/86-1) that covers the history of computing and the basic concepts of computing, while providing hands on experience with single user microcomputers and with a multi-user minicomputer. Washburn's implementation of this course does not require knowledge of any programming language. This class became a prerequisite to our language classes since it taught the students how to access our main computer and how to create files with a text editor on the system. However this course did not deal with algorithm development methods. At the same time, the Washburn University Philosophy Department developed a Logic of Computer Programming class which dealt with predicate logic, flowcharting techniques, structured programming concepts, and an introduction to W-O diagrams. Our department added this course as a language prerequisite in an effort to give the student an algorithm development base to use in the programming classes.

There were some problems with this approach. One was that a student, upon developing an interest in computing by taking the introductory course, was required to wait a semester before taking the COBOL class. Another problem was that we have teachers using their own preferred design and documentation tools and this caused some confusion and frustration for our students. To solve these problems, the department developed a new course, "Introduction to Structured Programming" (CIS/86-3), which teaches W-O diagrams and KOA design methodology. In the second half of the semester, the students study Pascal and implement algorithms designed during the first half of the semester. This course may be concurrent with the Philosophy course which places the student on a computer within one and a half semesters of starting in our program. This course in effect establishes a departmental standard for design methodology for individual instructors to build upon, the Warnier-Orr (W-O) diagrams and Ken-Orr and Associates (KOA) design methodology.

Introduction to Structured Programming (henceforth CM 111) allows the department to combine many of the topics needed for an individual language course and thus avoid duplication. CM 111 also encourages the student to logically design a program before writing code, hopefully before they have a chance to develop bad habits. This is done by requiring the student to design three separate programs completely and be graded on this design before they start learning a language or write code. In this way the students are taught to design correct programs not to test and modify a program until it is correct. The projects are developed using KOA's data structured design methodology, text: Data Structured Program Design, by Kirk Hansen (4). This gives the students an excellent basis for attacking programming projects in future classes.

The main advantage this course gives the department is a common standard as a base for all our programming classes, our systems analysis and design courses, and our database course. All of this reduces the confusion our students experienced in the past as they went from teacher to teacher and method to method. Having this common thread throughout the curriculum would be enough justification to establish a standard, the encouragement of correct and modern techniques is an added benefit.

One might justifiably ask how and why the choice of W-O diagrams and KOA methodology was made. First of all KOA (and W-O diagrams) is a good design and documentation methodology. It uses the same documentation tool (W-O) for output design, input design, program design, system design, database design, and hierarchy representation. It is one of the three or four main methodologies in use today. The KOA methodology is used in over 90 companies from 25 states and Switzerland and is growing in popularity. The W-O diagrams appear in two books on design tools, one by James Martin and the other by William Davis. The methodology is quite readable to the non computer literate user as well. The final reason, to be honest, is that KOA is based in Topeka, Kansas, our home as well and we are able to consult and interact with the firm whenever questions or problems of interpretation arise.

Washburn CIS CURRICULUM

With CM 111 out of the way, a student may then progress to COBOL Programming (CM 121 - CIS/86-4) and/or Information Structures (CM 305 - CIS/86-6). Following CM 121, Advanced COBOL Programming (CM 221) and Assembler Language Techniques (CM 231) are required. Information Structures (CM 305) is another required course which must follow CM 111. The completion of CM 305 and CM 231 then provide the background necessary for Introduction to Operating Systems (CM 322). A very important part of Washburn's CIS curriculum usually starts during the student's junior year. Systems Analysis for Management (CM 337) and then Systems Design (CM 338) expands the student's knowledge by discussing the theory and application of Data Base Management Systems (DBMS). In addition to all other requirements a major must take one of the following three programming languages: RPG, BASIC, or FORTRAN.

CM 111 - COURSE DESCRIPTION

Our Introduction to Structured Programming course (CM 111) is divided into two parts. The first takes approximately six weeks of a fifteen week semester and is an introduction to the KOA methodology. The three main principles of this methodology are: 1. Output oriented, 2. Data structured, and 3. Logical issues before physical issues. We teach the step by step process of developing an algorithm (called a logical process structure) based on the book Data Structured Program Design, by Kirk Hansen (4). We assign three projects for design by the student. Each project builds in complexity on the previous project(s). Each design is graded and handed back before the next one is assigned. We cover the first eleven chapters of Hansen's book.

The second part of CM 111 is teaching a language and utilizing that language to implement the designs from the first part. We feel that it is important for the students to implement their designs in the same course. We have chosen Pascal for the language because it is a fairly easy language to learn as a first language. It also has the structured concepts that allow easy conversion from algorithm to code. We feel that this modest exposure to a simpler language facilitates the students ability to learn and use COBOL, our core language.

EVALUATION

CM 111 was first taught in the spring semester of 1985. While there has not been enough time to fully evaluate the effectiveness of the course, the instructors of advanced language courses have indicated improved performance by CM 111 students as opposed to those students without this background. Furthermore, the instructors also report they are able to cover more material in their courses since a basic design methodology is already known by the students. We have found some difficulty with finding language textbooks which use the W-O diagrams, but with modest effort on the part of the instructors, they are able to adapt texts with a generic design method. CM 111 also requires more time by the instructor to grade the algorithms from the first part of the semester. Since the algorithms have not yet been implemented on the computer, the algorithms and their documentation must be read in depth and carefully evaluated. Some concern was raised on the infringement of the instructor's freedom of choice with respect to class content, however we feel that the efficiencies provided by the existence of a common methodology overrides this concern. Since this methodology is to be used as a base, others may also be covered at the same time. The students are also exposed to other methods in the course "Philosophy of Computer Programming" utilizing Harold Rood's book (6).

SUMMARY

While there have been some positive results since the implementation of CM 111 and the design standard of W-O diagrams and KOA methodology, it will be several years before it can be fully evaluated. It takes time for the influence of the course to move through our curriculum and be utilized in our upper division analysis, design, and database courses. At Washburn University, we feel that the future will belong to the designers of systems, not the code writers, and thus a good background in design methodology is necessary. Hence we believe that the implementation of CM 111 and our departmental standard is definitely a positive step towards this goal.

REFERENCES

1. Dale, N. and Orshalich, D.; PASCAL, Lexington, Mass.: D.C. Heath and Co., 1983.

2. Davis, W.; Systems Analysis and Design, Reading, Mass.: Addison-Wesley Publishing Co., 1983.

3. Davis, W.; Tools and Techniques for Structured Systems Analysis and Design, Reading, Mass.: Addison-Wesley Publishing Co., 1983.

4. Hansen, K.; Data Structured Program Design, Englewood Cliffs, New Jersey: Prentice-Hall, 1986.

5. Martin, J. and McClure, C.; Diagramming Techniques for Analysts and Programmers, Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

6. Rood, H.; Logic and Structured Design for Computer Programmers, Boston: Prindle, Weber & Schmidt, 1985.

7. DPMA Model Curriculum CIS'81; DPMA Education Foundation Committee on Curriculum Development, 1981.

8. DPMA Model Curriculum CIS'86; DPMA October 1985.

"FOURTH-GENERATION LANGUAGES AND THE 'CIS' CURRICULUM: THE ANALYSIS, DESIGN, AND DEVELOPMENT OF COMPUTER INFORMATION SYSTEMS"

by

Dr. Jan Prickett
Information Systems Department
College of Business
Northern Kentucky University

ABSTRACT

The methodology of the analysis, design, and development of computer information systems is currently undergoing major changes in both the business and academic environments. An important aspect of this change is the employment of fourth-generation languages. These languages provide greater efficiency and cost savings for businesses. Academically, implementation of fourth-generation languages within the curriculum increases the problem-solving potential of the student. Recent changes in the DPMA Model Curriculum (1986 version) reflect the growing importance of fourth-generation languages in the CIS curriculum.

INTRODUCTION

Fourth-generation languages provide the emphasis for a significant redirection of the pedagogical approach to a part of the computer information systems curriculum. Those courses which deal with analysis, design, and development have the opportunity to use these advanced languages as a new 'tool'. We can define a true fourth-generation language as one which performs the functions of query language, report generator, graphics generator, application generator, and high-level programming language. As fourth-generation languages are used increasingly in the business environment, the need arises to implement their use in the classroom. But this need is not merely born of the necessity to keep pace with the market into which our students are placed. Fourth-generation languages allow for more efficient use of class time and a better grasp of the role of computer systems in our society.

FOURTH-GENERATION LANGUAGES AND THE BUSINESS ENVIRONMENT

The problem of application program development has grown to calamitous proportions. In the late 1970's, computer time became less costly than the time of people, as determined by instructions per second. Personnel costs have increased as programming backlogs have grown. Backlogs have increased to the point where even maintenance backlogs would require many months or years to eliminate if no new requests were accepted. (This is illustrated by a recent survey taken by the Quality Assurance Institute at a software maintenance conference. The 37 surveyed organizations would require an average of 22.8 months to complete their existing maintenance backlogs.) The need to more quickly meet the requests of end-users can be partially solved by turning to fourth-generation languages. These languages allow for information to be formulated in less time, and for end-users to bypass programmer/analysts in their quest for information. James Martin has stated that, "In typical corporations with substantial well-designed data bases in existence, 70% of end-user needs can be met with query languages and report generators. In many cases less than 10% of the end-user demands for new applications require conventional DP development with formal programming specifications and languages such as COBOL or PL/1." (Application Development Without Programmers, Prentice Hall, 1982, page 83)

Fourth-generation languages are currently changing the manner in which computer information systems are being developed and used in the business environment. As with the advant of each of the first three generations (machine language, symbolic language, and third-generation languages such as COBOL), fourth-generation languages provide advantages over previous generations. Languages such as FOCUS, NOMAD, and RAMIS II allow for greater ease of programming, shorter development time, incorporated data-management tools, and more end-user involvement in information creation. While conventional application programming (e.g., COBOL) will not be eliminated, case studies have shown that in some corporations up to 95% of conventional application programs can be replaced by higher level language programs.

Fourth-generation languages envolved from the concept of prototyping. Before the 1980's, modeling could not be used in developing application programs because the cost of the prototype approximated the cost of the program. Fourth-generation languages have changed this. Not only do fourth-generation languages allow for the cost efficient creation of prototypes, but the 'prototype' often becomes the application program itself. Traditional program development throws numerous blocks between the user and the programmer. These blocks include factors within the System Development Life Cycle itself, the backlog of application requests, the relatively lengthy programming period in third-generation languages, and the complexity of the user-programmer/analyst interface. Fourth-generation languages help to eliminate these blocks. By reducing the complexity of system development, the nature of the System Development Life Cycle changes. As the user and programmer/analyst work together, former developmental steps are either not needed, or are reduced in time required for completion. Lengthy specifications are not written; procedural, structured design tools are eliminated; lines of code are reduced to a fraction of those required in the past; and debugging is greatly reduced because errors are largely the result of analysis and design.

1981 CIS CURRICULUM AND THE TRADITIONAL APPROACH

DPMA Model Curriculum for Undergraduate Computer Information Systems Education outlined a suggested curriculum for college and university study. This material was prepared by the DPMA Education Foundation Committee on Curriculum Development and published by the DPMA Education Foundation in 1981. This curriculum set forth a series of four courses which involved the analysis, design, and implementation of computer information systems.

CIS-4 Systems Analysis Methods
CIS-5 Structured Systems Analysis and Design
CIS-6 Database Program Development
CIS-7 Applied Software Development Project

CIS-4 and CIS-5 followed the traditional path of analysis, design, and development. They stressed the System Development Life Cycle, and associated tools and techniques such as:

structured charting
design criteria, including considerations of cohesion and coupling
structured English
decision tables and decision trees

CIS-6, 'Database Program Development', was established as "an introduction to application program development in a database environment with an emphasis on ... using a host language (COBOL)." (page 36) Finally, CIS-7, 'Applied Software Development Project', was a course in which the knowledge acquired in the previous three courses would be applied within a group project.

This four-course series relied on tools and techniques that are time consuming, and whose complexity leads to duplication of effort. The new curriculum outline is designed to reduce the time spent in analysis, design, and development.

1986 DPMA MODEL CURRICULUM

The 1986 DPMA model curriculum (The DPMA Model Curriculum for Undergraduate Computer Information Systems, 1986) reflects the current state of the business environment --- an increased reliance upon fourth-generation languages, and decreased use of traditional design and implementation tools. The 1986 curriculum courses corresponding to the above mentioned 1981 curriculum courses are:

# CIS/86-5,	Systems Development Methodologies: A Survey
# CIS/86-6,	Data Files and Databases
# CIS/86-7,	Information Center Functions
# CIS/86-8,	Systems Development Project

With the use of non-procedural, fourth-generation languages, the same degree of stress on the tools used in the former CIS-4 and CIS-5 is no longer necessary. This allows the compression of these two courses into the new 'Systems Development Methodologies: A Survey' (CIS/86-5).

The core course on data bases, CIS/86-6, maintains its position following the instruction of analysis and design methodologies. However, the course description reflects the trend towards implementation of higher-level languages. Such languages implement the use of relational data bases. If fourth-generation languages are to be an integral part of the analysis-design-development process, data-base concepts must be increasingly stressed. To allow information access by end-users requires that initial data-base design is well constructed.

'Information Center Functions' (CIS/86-7) is an addition to the DPMA model curriculum. This course further reflects change within the business environment. As fourth-generation languages decrease the need for traditional life-cycle methodologies, information systems needs will continue to change. The role of the programmer/analyst will evolve to include ever-increasing intra-corporate consultation --- for example, guiding end-users in the capacities of fourth-generation languages for information retrieval. 'Information centers' have arisen to meet this need.

'Systems Development Project' (CIS/86-8) serves as the replacement for the former CIS-7. This course is the capstone course which allows students to display the knowledge acquired in previous classes. Emphasis is to be placed upon current business tools which would include fourth-generation languages. Traditionally, a fully developed systems project took longer than one semester to complete, or involved the effort of a large team. Often the project was begun in a previous course and carried forward into the systems development course. The use of fourth-generation languages eliminates these problems. Many previously used design tools may be avoided; and coding is reduced to a fraction of what was previously required. While third-generation languages may still be needed for certain aspects of system development, the bulk of the system can be quickly developed using a fourth-generation language.

CONCLUSION

The greater emphasis upon fourth-generation languages within the DPMA curriculum is therefore significant for several reasons. First, it reflects the reality of the business environment. Fourth-generation languages are providing solutions to problems involving the analysis, design, and development of computer information systems. This reality must be reflected within the education acquired by our students. This new emphasis is for the betterment of students' personal success, and the health of the organization for which they work. Secondly, the use of fourth-generation languages improves our ability to teach the process of analysis, design, and development. By reducing the amount of students' time spent upon specific former traditional methodologies, greater emphasis can be placed in other areas. Among the benefits accrued are increased individual involvement in the overall system, and increased contact time between student and professor.

SELECTED REFERENCES

Adams, David R. and Athey, Thomas H., editors. DPMA Model Curriculum for Undergraduate Computer Information Systems Education, Data Processing Management Association, Park Ridge Illinois, 1981.

Boar, Bernard H. Application Prototyping, New York: John Wiley and Sons, 1984.

CIS '86: The DPMA Model Curriculum for Undergraduate Computer Information Systems, Data Processing Management Association, Park Ridge, Illinois, October 1985.

Martin, James. Applications Development Without Programmers, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1982.

Martin, James. An Information Systems Manifesto, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1984.

Winston, Patrick H. and Prendergast K. The AI Business: Commercial Uses of Artificial Intelligence, Cambridge: MIT Press, 1984.

Teaching CIS/86-2:

An examination of curriculum issues

David L. Russell, Assistant Professor

Quantitative Methods and Computer Information Systems Department
School of Business
Western New England College
Springfield, MA 01119

ABSTRACT

The 1986 DPMA model CIS curriculum offers a major new challenge in CIS/86-2, a course focusing on microcomputer application software. Three major problems are identified: (1) the lack of hardware; (2) the lack of appropriate texts; and (3) the lack of appropriate teaching software. Potential solutions to these problems are discussed. The paper concludes with a brief description of the approach the author and his colleagues at Western New England College have found successful.

INTRODUCTION

CIS '86, the DPMA Model Curriculum for Undergraduate Computer Information Systems presents a new departure in the CIS curriculum. CIS/86-2, "Microcomputer Applications in Business", is a new course reflecting two major trends in the MIS profession: first, a course is provided for computer instruction in a non-procedural, non-language mode; and second, a specific location in the curriculum is now given to microcomputing.

Many CIS departments how face the challenge: how do we teach CIS/86-2? A number of curriculum related issues must be dealt with; the objective of this paper is to discuss several of these issues and raise potential solutions. The discussion will draw from the experience of the author and his colleagues at Western New England College, where a course similar to CIS/86-2 has been offered for several years as a junior-level CIS elective.

PROBLEMS ASSOCIATED WITH TEACHING CIS/86-2

A number of problems need to be overcome in order to teach CIS/86-2 well. Three particular problems can be clearly identified: the lack of appropriate microcomputing hardware; the lack of relevant textbooks; and the need for appropriate applications software. These three problems will be briefly outlined in this section; approaches to their solution will be discussed in the next section.

Many universities and colleges face a serious lack of appropriate microcomputers. The first and most basic problem is one of numbers: many universities and colleges simply do not have enough microcomputers available. A number of additional factors combine to exacerbate this problem. Past investments in hardware that is no longer in the mainstream of business microcomputing hampers development of strong curricula, since it is difficult to "write off" these investments when the macinery is still functioning well. Moreover, many senior administrators feel "burned" by past microcomputer investments and as a result display excessive caution in purchasing additional hardware. Finally, the acquisition of sufficient microcomputers does not in itself solve the problem: there is a need for the microcomputer to be configured as a mainstream machine, with appropriate video cards, memory expansion cards, disk capacity and output peripherals. Many administrators do not fully realize this need, and sole-source purchasing from microcomputer manufacturers does not provide for the full range of enhancements required.

Until recently, the single greatest problem was the availability of high-quality textbooks focusing on microcomputing and particularly on microcomputer applications software. Prior to 1984, such texts simply did not exist; after 1984, texts were available but of uneven quality. The texts available today are often too generic, providing few operational details, or are too specific, focusing on a single brand of microcomputer and a limited selection of software. Further, the texts that are available seem to divide into two camps: those that focus on existing commercial software vs. those that utilize special pedagogical software that mimics commercial software.

A closely related problem is the

availability of appropriate application software. In general, commercially available software is quite expensive, even with education discounts and site licensing when available. Some firms, for example Lotus Development Corporation, originators of the Lotus 1-2-3 spreadsheet package, make no provision for purchase by educational institutions. Other firms, such as Computer Associates, providers of the SuperCalc4 spreadsheet, and Ashton-Tate, providers of the Framework integrated package, have been far more accommodating. Nonetheless, the problem remains: software that is actually in use in the business world tends to be expensive, sometimes prohibitively so. Some propose pedagogical software as the solution, a solution which simultaneously solves a number of licensure problems. Both approaches have their advantages, but the texts using pedagogical software have the disadvantage of not providing the student with crucial practical experience in commercial software.

SOME APPROACHES TO SOLVING THESE PROBLEMS

The problems enumerated above, and others, must be solved in some fashion in order to provide a strong microcomputer course. Some approaches to solving these problems are offered below, and are presented in the same order as the previous section.

First, we must recognize that the IBM PC family has come to dominate business microcomputing, and that it must be our primary, if not sole focus. Problems associated with the number of microcomputers available are often a function of local situations and thus not amenable to a generalized solution. If cost is the major governing factor, as it so often is, then compatible microcomputers become a viable alternative. Today, brand-name IBM compatibles are available from 70-100% of the cost of the IBM brand. Off-brand IBM compatibles are even less. In general, those that cost nearly what an IBM does compete with IBM on the basis of improved performance, while those with lower prices tend to only equal the IBM in performance and compete on the basis of price alone. The discounts offered often equal IBM's corporate discount, and often exceed those of IBM resellers and retailers: thus, the relative price relationship remains constant. A difficulty (one to which we will return on several occasions) is the transferability of knowledge gained on the compatible to IBM machines. In general, and particularly with the "clones", this is not a significant factor; some compatibles, however, have different keyboard and diskette layouts, which exacerbate transferability problems. A further factor must be considered: to what degree does experience gained on an IBM-brand machine aid students in their subsequent careers, and to what degree does experience on IBM compatibles do them harm? Many firms require IBM-specific experience, and unsophisticated personnel officers may not fully appreciate the nuances of difference between the IBM and the compatibles.

Beyond simple numbers is the need for machines configured closely to the mainstream of business microcomputing. Machines should have graphics adapters that support useful graphics and be provided with sufficient memory to run most types of software effectively. In today's business microcomputing environment, this generally means an IBM-class machine with (1) a Hercules-compatible monochrome graphics card or a CGA- or, preferably, EGA-compatible color graphics card, (2) more than 256KB of memory (preferably with the DOS conventional memory maximum of 640KB), and (3) commonly-used output peripherals. These additional factors cost money, but a more complex problem is the sole-source purchase of microcomputers from manufacturers. An IBM PC purchased directly from IBM, for instance, comes with IBM's inadequate monochrome adapter card and 256KB on the motherboard, necessitating an expensive memory card for expansion beyond 256KB (recent technological improvements may render this purchase less expensive, however). CIS instructors should insist that minimal PCs be purchased, with enhancements purchased from third-party vendors being added subsequently. In general, the prevailing standards of the industry should govern the enhancements selected. Multiple-source purchasing as advocated here can be troublesome and marginally more expensive, but it results in a machine more attuned to the body of application software CIS/86-2 seeks to address.

Many universities and colleges are saddled with outdated machines, such as the Apple II, that are now out of the mainstream of business microcomputing. It is a mistake to invest further funds in these machines, even if they are still functional, as the large body of application software does not run on these machines. Instead, it is best to transfer these machines to dedicated-purpose uses, such as word-processing stations. This enables the institution to gather some use from the machine during its remaining life, and allows a redirection of the microcomputer focus to viable machines.

Along a similar line, serious questions must be raised regarding excellent quality machines that have not enjoyed wide acceptance by the business microcomputing community, such as the Apple Macintosh and the Commodore Amiga. Despite their undeniable technological excellence, the fact remains that these machines do not address the body of application software used by business, and thus should not be the focus of CIS/86-2. It would be useful to have a few of these machines on hand in order to demonstrate functions not commonly available on the IBM PC, such as desktop publishing on the Macintosh.

The situation noted earlier with regard to appropriate textbooks has improved dramatically since 1984. A number of texts are available that focus on microcomputing, most of which include reference to specific software. Virtually all reference the IBM class of microcomputer (e.g., McLeod) while a few are more generic in focus (e.g., Loebbecke). Other texts focus nearly exclusively on the application software, with relatively little attention to other related issues (e.g., Duffy). Still another class (Leigh, Dravillas) utilize special pedagogical software in place of standard business software.

Another trend is of benefit to the instructor of CIS/86-2: a number of introductory texts intended for CIS/86-1 have associated with them software packages. Again a distinction can be made between those offering commercial software (Sanders, Shelly and Cashman) vs. those utilizing pedagogical software. This association allows very close integration between the first two courses in the CIS curriculum and, since these courses are often mandated for all business school students, can be of great benefit to the entire student body.

The situation with regard to the availability of software is also improving. While actual commercial software remains expensive, educational discounts and site licensing agreements are mitigating this situation somewhat. The remaining problem is Lotus 1-2-3, which remains the dominant and standard-setting software of its class. Its producer, Lotus Development Corporation, has not shown any inclination to offer substantial discounts or site licenses, nor has it produced limited student versions.

Lotus 1-2-3 aside, many other software distributors are accommodating educational needs. The question is, however: for CIS/86-2, do we need the full power of commercial software? The author proposes that we do not. Instead, we can make use of a variety of limited function "student versions" of popular software. This software is characterized by being fully functional. The software is generally disabled in such a way so as to prevent its use in a normal business environment. These limitations, however, have little effect on the educational environment. For example, the student version of WordStar prints a message at the bottom of each page noting that it was produced with the student version; the student version of dBase II is limited in terms of the number of records it can contain.

These limited student versions are available from a number of sources. Prentice-Hall's PC Apprentice series, featuring WordStar, dBase II and several titles from the pfs line is a leader in this field. This series is particularly useful for students learning several pieces of software, as each product is quite inexpensive (approximately $15). In addition, some of the software associated with textbooks can be purchased separately: McGraw-Hill's student version of Framework, intended to accompany Sanders' textbook, is also available separately (approximate cost: $25). The student version of WordPerfect should also be considered (approximate cost: $75).

Alternative sources of low priced software include demonstration versions of software produced by software firms. These versions are either limited in capacity or lack a crucial command, rendering them useless in normal functioning. For example, Microrim's demonstration version of R:base 5000 has very limited capacity (though still enough for teaching purposes), while Computer Associate's SuperCalc4 and SuperProject demonstration versions disable the Save command. Such demonstration versions are available at very little cost, or are sometimes free.

Both of the above alternatives transfer the cost of acquiring the software to the student, whether or not bundled with the purchase of a textbook. In situations where this is not possible or desirable, institutions should look at "shareware". This software can be legally copied at will, but is vendor-supported (this constraint excludes the majority of public-domain software). Quicksoft's PC-Write, for example, is a fine word processor, the free distribution of which is encouraged. Other "shareware" products in the spreadsheet and database classes are available.

Texts using generic pedagogical software can also be considered. These have the advantage of eliminating licensing concerns, and generally do not have the capacity limitations associated with educational versions of commercial software. However, the fact remains that this software does not exist in the commercial world. While the skills learned may be transferred to commercial software with various degrees of ease, they do not directly prepare the student for the business world. Further, these products do not offer significant pedagogical advantages: they take no greater or lesser time to teach than does commercial software, nor do they explicate principles more efficiently than does commercial software. Since the use of commercial software, even in limited student versions, can be taught just as efficiently as pedagogical software, this author feels that the career advantages of familiarity with established commercial software should result in the teaching of commercial software whenever possible. This same argument applies to "shareware" and public-domain software as well.

WESTERN NEW ENGLAND COLLEGE'S APPROACH

At the author's institution, several of these approaches have been attempted, individually and in combination, with varying degrees of success. In recent semesters, the author has found success in using the student version of Framework as the unifying tool in his microcomputer courses. The word processing, spreadsheet, database and graphics components of Framework have been used to introduce these classes of software, with subsequent "self-learning" by students of commercial "stand-alone" products of the same class. Limited versions from the PC Apprentice series and commercial demonstration versions are extensively utilized. The author feels this self-learning is a crucial step, since students must learn to model the learning process for themselves: indeed, nearly all the software they will face in the future will be learned by actually using it. Later, the integrated nature of Framework is explored. A semester project incorporating a hardware and software proposal to solve a specific business problem rounds out the course.

CONCLUSION

The objective of this paper was to explore several curriculum issues surrounding the teaching of CIS/86-2. It is hoped that the discussion, or possibly debate, generated by this paper will aid the reader in implementing CIS/86-2 at his or her home institution.

REFERENCES

Dologite, D.G., Using Small Business Computers, Prentice-Hall, 1984

Dravillas, Paul, Steven Stilwell and Brian K. Williams, PowerPack for the IBM PC, Times Mirror/Mosby, 1986

Duffy, Tim, Four Software Tools, Wadsworth, 1986

Leigh, William E., Annette J. Thomason, and Noemi M. Pez, Microcomputers: a Hands-on Introduction, Boyd & Fraser, 1986

Loebbecke, James K. and Miklos A. Vasarhelyi, Microcomputers: applications to business problems, Irwin, 1986

McLeod, Raymond, Jr., Decision Support Software for the IBM Personal Computer, SRA, 1985

Sanders, Donald, Computers Today, 2nd ed., McGraw-Hill, 1985

Shelly, Gary B., and Thomas J. Cashman, Computer Fundamentals with Application Software, Boyd & Fraser, 1986

STUDENT PERFORMANCE APPRAISAL IN CIS-7

ADAPTATION OF A BUSINESS TOOL TO THE EDUCATIONAL ENVIRONMENT

James K. Carter
Assistant Professor of Computing
University of Wisconsin - Stevens Point

One of the objectives of the CIS-7 curriculum, as implemented at the University of Wisconsin - Stevens Point, is to provide each student with an experience that resembles actual business practices as closely as practicable. In striving to achieve this objective, the author has developed and is in the process of implementing a formal student performance appraisal program which is modeled after the technique which he practiced during his eleven years in the business world. This paper discusses techniques for developing and implementing the appraisal program as part of the CIS-7 curriculum, including insights into establishing program objectives, developing forms for use, and implementing the appraisal procedures. In addition, it presents several precautionary areas to heed, and discusses some of the benefits of an effective program.

I have often been heard in my office muttering something to the effect that one cannot teach the CIS-7 curriculum, and further, that a student cannot learn it. Rather, it must be lived by both instructor and student. In order to provide maximum benefit to the student, this semester-long process of living the curriculum should strive to model the real world as much as practicable. This objective can be advanced by adapting business world tools to the educational environment. Whereas many enlightened businesses use the process of performance appraisal to evaluate employees, such evaluation techniques can be adapted to serve as the basis for student evaluation in CIS-7. Indeed, such adaptation is currently underway at the University of Wisconsin - Stevens Point.

Adaptation of the techniques is threefold in nature, beginning with the establishment of clear objectives for the student appraisal program. An appropriate set of forms can then be developed to assist in achieving these objectives. Finally, the performance appraisal procedures can be developed and implemented.

ESTABLISHING APPRAISAL OBJECTIVES

In the business world, the construction of an effective performance appraisal program begins with a clear definition of the program's objectives. In my managerial experiences, there have been many such objectives, which in general can be summarized into five statements. Following is a discussion of these five primary objectives as adapted for use in the CIS-7 curriculum at UW-SP.

- OBJECTIVE 1 - To provide a means of evaluating student performance as a basis for grading

In the business world, evaluation of employees serves as one of the foundation blocks for compensation reviews - wage and salary levels. This objective is adapted to support the student "compensation" review - grades.

- OBJECTIVE 2 - To provide a means of clearly defining and understanding areas of student performance that require improvement

Students, like employees, sometimes fall short of the expectations of their superiors. To meet this objective, the performance appraisal process strives to correct such behavior. Herein lies the heart of the teaching and learning experiences of the CIS-7 course. Each student will encounter varying levels of success and failure in applying the concepts and techniques learned in prior CIS coursework. The CIS-7 instructor is obliged to recognize such shortfalls and guide the student toward improving the application of his or her system development skills. At the same time, the instructor is obliged to positively reinforce correct application of these skills.

- OBJECTIVE 3 - To provide a mechanism for motivating students to achieve higher levels of performance

Admittedly, this objective is somewhat selfish in that it primarily benefits the students' clients. As in the business world, however, higher levels of performance are consistent with both career advancement goals and self-actualization needs.

- OBJECTIVE 4 - To provide a means of specifying mutually agreeable student goals which will advance the client project toward completion

Adaptation of this objective is difficult due to the relatively short timeframe involved, i.e. one semester. However, if the first round of the performance appraisal process is completed half way into the semester, then student goals for the balance of the semester can be based on the project work plans or other similar project management tools. The establishment of such goals has three primary benefits. First, it gives the student a concrete target at which to aim for the second half of the semester. Second, it provides a performance standard against which future performance can be measured. And third, it helps to keep project work plans at a conscious level, increasing the likelihood that the project will not stray too far from the intended path.

- OBJECTIVE 5 - To provide a forum in which the student is encouraged to advance ideas for improvements either to the course or to the instructor's performance

Often times those who do the work and who are governed by the rules are the best constructive critics of the procedures and/or rulers. The instructor can, for example, learn a great deal about interpersonal skills and motivation techniques by listening closely to a student who has been carelessly reprimanded for making a mistake. In addition to improving the instructor's skills or the course curriculum, the use of this forum can provide invaluable experience to the student in the area of how to deal effectively with superiors.

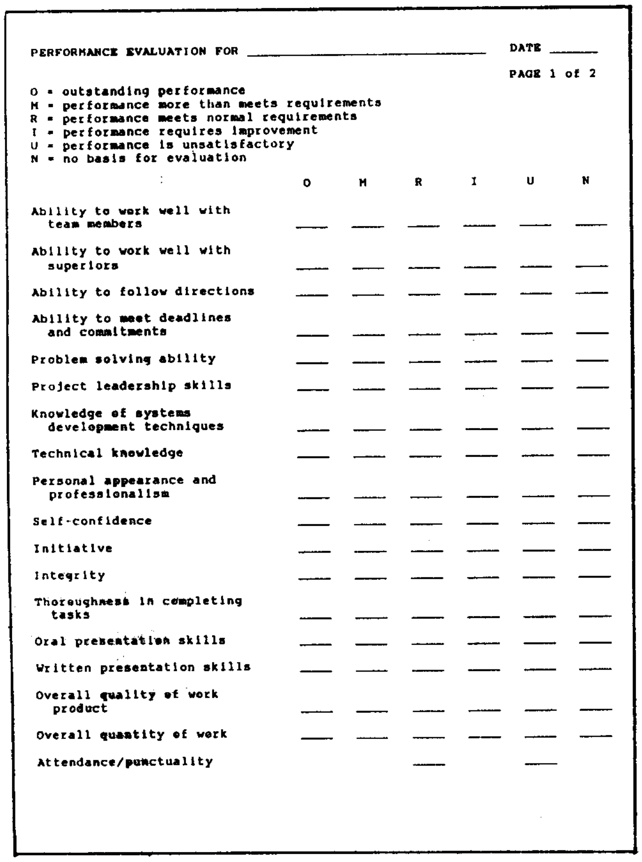
DEVELOPING THE APPROPRIATE FORMS

Development of the appraisal forms should be guided by the above five objectives. With this in mind, the development process should consider the guidelines that follow. For reference, a sample of the format of the basic form is depicted in Figure 1 below.

1. Separate, customized versions of the basic form should be developed for individual completion by the student, the instructor, and appropriate client personnel. All three parties should be encouraged to provide input into the process.

2. The forms should be regarded as tools to be used in the process, and not as ends in and of themselves. In this regard, they should be developed to facilitate a confidential, one-on-one appraisal interview between the instructor and the student.

3. The instructor's version of the form should contain a summary section for documenting the interview highlights and results.

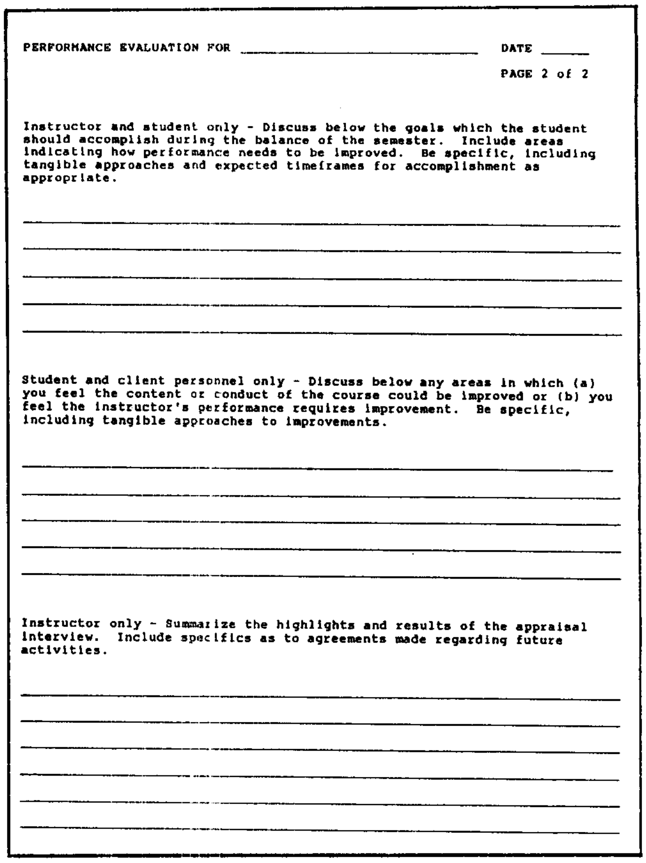


PERFORMANCE EVALUATION FOR ______________________ DATE ______

PAGE 1 of 2

O = outstanding performance
M = performance more than meets requirements
R = performance meets normal requirements
I = performance requires improvement
U = performance is unsatisfactory
N = no basis for evaluation

	O	M	R	I	U	N
Ability to work well with team members	___	___	___	___	___	___
Ability to work well with superiors	___	___	___	___	___	___
Ability to follow directions	___	___	___	___	___	___
Ability to meet deadlines and commitments	___	___	___	___	___	___
Problem solving ability	___	___	___	___	___	___
Project leadership skills	___	___	___	___	___	___
Knowledge of systems development techniques	___	___	___	___	___	___
Technical knowledge	___	___	___	___	___	___
Personal appearance and professionalism	___	___	___	___	___	___
Self-confidence	___	___	___	___	___	___
Initiative	___	___	___	___	___	___
Integrity	___	___	___	___	___	___
Thoroughness in completing tasks	___	___	___	___	___	___
Oral presentation skills	___	___	___	___	___	___
Written presentation skills	___	___	___	___	___	___
Overall quality of work product	___	___	___	___	___	___
Overall quantity of work	___	___	___	___	___	___
Attendance/punctuality			___		___	



PERFORMANCE EVALUATION FOR ______________________ DATE ______

PAGE 2 of 2

Instructor and student only - Discuss below the goals which the student should accomplish during the balance of the semester. Include areas indicating how performance needs to be improved. Be specific, including tangible approaches and expected timeframes for accomplishment as appropriate.

Student and client personnel only - Discuss below any areas in which (a) you feel the content or conduct of the course could be improved or (b) you feel the instructor's performance requires improvement. Be specific, including tangible approaches to improvements.

Instructor only - Summarize the highlights and results of the appraisal interview. Include specifics as to agreements made regarding future activities.

FIGURE 1 - Sample Performance Appraisal Form

4. All versions of the form should be simple and easily understood, without being overly objective in nature.

5. In order to facilitate achieving program objectives 1, 2 and 3, each version of the form should contain a section where the user of the form can rate, based on his or her personal opinion and experiences, the performance of the student in the following categories:

   - Ability to work well with team members
   - Ability to work well with superiors
   - Ability to follow directions
   - Ability to meet deadlines/commitments
   - Problem solving ability
   - Project leadership skills
   - System development techniques
   - Technical knowledge
   - Personal appearance/professionalism
   - Self confidence
   - Initiative
   - Integrity
   - Thoroughness in completing tasks
   - Oral presentation skills
   - Written presentation skills
   - Overall quality of work product
   - Overall quantity of work produced
   - Attendance and punctuality

   The rating scale for the above categories should allow for exercising judgement and should include possible ratings of outstanding, more than meets requirements, meets normal requirements, needs improvement, unsatisfactory and no basis for evaluation.

6. The instructor's and student's versions of the form should include a section for identifying possible goals to be accomplished during the remainder of the semester. This will facilitate achieving objective 4.

7. The student's and client's versions of the form should include a section for identifying possible ways in which the course or the instructor's performance can be improved. This will assist in addressing objective 5 by allowing the student an opportunity to provide honest feedback.

It should be noted that the above guidelines are, indeed, guidelines. As with other managerial techniques, they must be customized to fit the particular circumstances at hand.

IMPLEMENTING THE PROCEDURES

Procedures will vary depending on class size, size of project teams and the overall nature of the instructor and students. In general, however, the following schedule and timetable can be used as a guideline for developing customized procedures.

On or near the first day of class, the appraisal program should be announced so that the students will be aware of its intended use. This can be done by including a statement of the program objectives and procedures in the course syllabus. Sometime during the sixth or seventh week of the semester, the program objectives and procedures should be announced again. This time, however, the appropriate forms should be issued with instructions for completion.

At this point, ample time should be allowed for completion of the forms and return of them to the instructor, i.e. at least one week. An individual, confidential interview between each student and the instructor can then be scheduled. Approximately 30 to 45 minutes should be allocated for each interview. During the interview process, all forms should be made available for student viewing. The ratings and comments on the form should serve as a vehicle for stimulating conversation about the student's performance. Any specific remedies to poor performance should be documented in the appropriate section of the instructor's form. After conducting the interview, the instructor should immediately complete the summary section of the form while the process is fresh in his or her mind. Copies of all forms should be forwarded to the student as a matter of courtesy and good faith.

Alas, the process is not yet complete. The entire cycle should be repeated at the end of the semester, with particular attention paid to the agreed upon goals resulting from the first interview. With the results of both rounds of the appraisal process in hand, final course grades can be assigned. Having gone through the entire process twice during the semester, assignment of grades should be a relatively painless exercise.

PRECAUTIONS TO OBSERVE

Several precautionary notes are in order which should be heeded by those who would attempt to develop a similar student performance appraisal program for the CIS-7 course.

First, it must be stressed to all users of the forms (instructor, student and client personnel) that the forms are tools to stimulate conversation during the appraisal interview. They should not be viewed as objective instruments to be graded in the traditional sense. Further, completion of the forms is relative to the level of performance that the user equates to "meeting normal requirements", i.e. performance which client personnel view as outstanding may be regarded by the instructor as simply meeting normal requirements.

Second, the interview should be conducted as scheduled, if at all possible, without postponements. Unnecessary postponements can create the impression that the instructor does not value the appraisal program. This impression will render the results meaningless.

Third, the interview must not be approached by either party as an adversarial exercise. Effort should be exerted to keep it on a professional basis, stressing positive angles whenever and wherever possible. In addition, the instructor should concentrate on listening, whereas the student should concentrate on talking. This will provide the best opportunity for the instructor to accurately assess the student's performance, as well as provide a motivating atmosphere consistent with objective 5.

Fourth, further insight into the process may be beneficial to those instructors who have not had the occasion to engage previously in a formal performance appraisal program. Such further insight can be obtained either by reviewing the applicable discussions in the publications listed at the end of this paper, or by discussing the process with those colleagues having expertise in the area of human resource management or human behavior.

POTENTIAL BENEFITS

The benefits of successfully implementing an effective student performance appraisal program in CIS-7 are intuitively obvious based on the program objectives and procedures. Grading becomes painless, yet is not particularly exposed to appeal since student performance is well documented, exhaustively discussed with the student, and supported by independent client evaluations. Incorrect application of system development skills is recognized and corrected during the semester, and more importantly, prior to graduation. Clients receive a better product since students are motivated to perform at higher levels, and since instructors are obliged to remain in close contact with the students' activities in order to be able to administer the appraisal program. And most importantly, the students receive invaluable benefit from the experience of living the course as they will undoubtedly live their careers in the business world. And I believe that this is what the CIS-7 curriculum intended all along.

SUGGESTED REFERENCE MATERIAL

Although the publications listed below have not been specifically cited or referenced in preparing this paper, a review reveals that they offer some valuable and practical insights into the performance appraisal process. Also please note that this list is by no means an exhaustive bibliography of the subject.

ALEWINE, THOMAS C. "Performance Appraisals and Performance Standards." Personnel Journal 6, No. 3 (March 1982): 210-213.

"Appraising the Performance Appraisal." Business Week (May 1980): 153-4.

BARRETT, RICHARD S. Performance Rating. Chicago, Illinois: Science Research Associates, 1966.

BEER, MICHAEL. "Performance Appraisal: Dilemmas and Possibilities." Organizational Dynamics 9 (Winter 1981): 24-36.

HENDERSON, RICHARD I. Performance Appraisal, 2nd ed. Reston, Virginia: Reston Publishing, 1984.

WILLIAMSON, DENNY. "A Primer on Performance Appraisals." Supervisory Management 24 (June 1979): 35-7.

ASSESSING THE CURRENT MIS CURRICULA

Mehdi Khosrowpour, D.B.A.
The Pennsylvania State University at Harrisburg
The Capital College
Middletown, PA 17057

The rapid integration of Management Information Systems into all aspects of organizations during the past few decades has created a demand for MIS staff candidates who are not limited to knowledge of the technical aspects of information systems, but rather, who possess a broad understanding of information systems, organizational behavior, and management. The lack of business and management orientation can be considered as one of the major deficiencies of current MIS curricula. In this regard, this paper proposes an MIS curriculum model which can be used to broaden MIS students' understanding of information resources.

INTRODUCTION

In the past few decades management literature has witnessed a tremendous increase in the volume of writings about Management Information Systems (MIS). MIS concepts, applications, problems, and future potential have been discussed and assessed, both by researchers and practitioners. Each group has recognized and valued the importance of MIS for the achievement of success in this very competitive business world. At no time has the need for effective management of information resources been so important as in recent years, due to the fact that more and more firms are realizing the true power of computer-based information systems in providing information and assistance to decision-makers at all levels of an organization. In response to the aggressive growth in information requirements, many firms have been searching for more effective ways of managing their information resources, and concluding that better management of MIS systems requires better managers - people who possess strong managerial, rather than strictly technical skills (Khosrowpour, 1985, p. 18).(1) Unfortunately, this change in the perceptive of top managers regarding the skill profile of MIS managers has not been reflected in the current MIS curricula used to train the future MIS managers of the corporate world. This paper explores the reasons for the lack of managerial orientation in MIS curricula currently used by universities, and proposes a model MIS curriculum to overcome some of these deficiencies.

MIS MANAGEMENT

As information systems technology matures, providing more advanced equipment for information processing and facilitating the operation of computer-based information systems, more attention is given to the managerial aspects of MIS personnel. It was noted by Ferreira and Collins (1979)(2) that the great advances in computer technology have relieved MIS managers, to a large extent, of technically related responsibilities. The concern over technical feasibility, which dominated systems developed during the 60s and early 70s, has been replaced by a concern over general managerial feasibility, as pointed out by Dickson and Wetherbe (1985): "...at the end of the 1960s, organizations were disappointed. They found that technical feasibility was not enough to warrant multimillion dollar expenditures" (Dickson and Wetherbe, 1985 p. 4).(3)

Further, the orientation of MIS management in many organizations is changing in the direction of greater user involvement. So the current MIS manager, instead of serving as the technical custodian of computer hardware entities, now functions more like an agent between MIS resources and end users. This change in role was predicted by Gilbert (1978), who concluded that the role of information processing within the organization would be changed, and with it, the role of the MIS manager. Many MIS managers have already found themselves to be incapable of coping with the behavioral issues which arise in MIS management. This problem was noted by Gupta (1974)(4), who found that most information managers do not have adequate training in managerial jobs, and thus lack those skills which are so essential for developing a successful MIS. Perhaps this lack of expertise on the part of MIS managers can be blamed on the MIS curricula adopted by colleges and universities around the country, which were designed to satisfy the previous needs of the industry, but which have not kept pace with changing personnel requirements.

THE MIS MANAGER

During the 1960s, most companies employed staff with strong technical backgrounds to manage their computer centers. The major requirement for the manager of a computer center, better known as Data Processing (DP), was technical competence, particularly the ability to cope with hardware maintenance and operations. Further, the DP management position was viewed as purely technical, not managerial. The job was conceived as a strictly "line" position, concerned with supervision and operations of computer systems, and preparation of computer-generated outputs to be used predominantly by accountants, controllers, and inventory managers. In this era of computer systems operations, the primary feasibility concern was technical, according to Dickson and Wetherbe (1985); economic feasibility was a secondary concern for most companies.

As the use of computer systems leveled off, many firms began to apply newly introduced MIS concepts and applications to various business functions by developing computer-based information systems. Consequently, the former data processing center manager became the new MIS manager. Unfortunately, this change of status did not reflect additional skills, nor did it indicate a change in the organizational structure of the computer center. One rationale for the promotion of DP manager to MIS manager is given by Tomeski and Sadek (1980)[5], who argued that many organizations promoted the technical DP staff person to a management position in order to utilize the person's technical talent and, at the same time, keep the person happy.

Therefore, while the new concept of MIS was intended to be utilized by management at all levels, the custodian of this entity remained a totally technically oriented person, rather than true manager. The lack of managerial expertise of MIS managers in the past has been the main cause of suspicions about the value of MIS, and Schlosser (1974)[6] expresses a great concern over the behavioral implications of MIS applications. Along the same line, Carper (1977)[7] states that "...the MIS problem is largely a people problem caused by an inadequate understanding of and consideration for the associative behavioral implications inherent in any MIS" (Carper, 1977, p. 48)

THE MIS CURRICULUM

In the early 60s, while companies were starting to form their new data processing organizations, many colleges and universities likewise began to develop new programs in DP. Soon, standard curricula were introduced for these DP programs. Traditionally, these programs were housed in either Accounting or Mathematics Departments. The rationale for this placement was based mainly on the general perception that computers were geared toward use in business functions by business managers. At the time, computers were viewed as 'number crunching' tools, which could perform millions of arithmetic operations in a millionth of a second. Since most accounting and engineering applications do, in fact, deal with large volumes of numerical data, it was quite natural to locate a DP program in either of these two departments. The typical DP program offered a variety of quantitative courses, in addition to many computer hardware- and software-related courses. The main intention of these programs was to train future DP personnel in matters that would enable them to know the ins and outs of computers and to manipulate the programming languages necessary to utilize the power of these machines. It is no surprise to anyone that the majority of graduates from these programs were technically oriented; Dickson and Wetherbe (1985) concluded that the majority of people in charge of information resources started out in a technical computer related field.

By the late 60s, many schools had begun to establish new programs in Management Information Systems to satisfy an expanding demand for computer systems specialists with expertise beyond that of programming languages and operating systems. The main objective of these newly formed MIS programs was to train students to program, analyze, and manage computer-based information systems. The establishment of MIS programs in business inevitably influenced these new MIS curricula. The majority of these curricula were developed by the Data Processing Management Association (DPMA), and the Association of Computing Machinery (ACM). Cotterman (1983)[8] reviewed the model curricula recommended for information systems studies in the period of 1968 through 1982, and listed ten different curricula either designed from scratch or revised from a previously existing curriculum. He concludes that the differences among the curricula are substantial.

Table 1

History of MIS Curricula

Year	Title	Organization
1968	Curriculum 68	ACM
1972	Curriculum Recommendations for Graduate Professional Programs in Information Systems	ACM
1973	Curriculum Recommendations for Undergraduate Programs in Information Systems	ACM
1974	An International Curriculum for Information Systems Designers	IFIP
1979	Curriculum 78	ACM
1981	DPMA Model Curriculum for Undergraduate Computer Information Systems Education	DPMA
1981	ACM Masters Curriculum in Computer Science	ACM
1981	Educational Programs in Information Systems (Survey)	ACM
1982	Curriculum Recommendation for Software Engineering	IEEE
1982	Information Systems Curriculum Recommendation for the '80s: Undergraduate and Graduate programs	ACM

NOTE: Adopted from Cotterman, William, 'A Comparative Analysis of Information Systems Curricula,' ISECON83.

PROBLEMS WITH THE CURRENT MIS CURRICULA

A quick look at the curricula which have been used in MIS programs in colleges and universities in the past two decades clearly indicates that students were encouraged to take as many analytical and technical courses as possible. This overemphasis on the technical components of computer-based information systems was noticed by Lusa, Iscoff, and McCartney (1975) who, in fact, clearly state that "... in the past, many MIS curricula have over stressed the technical or operational aspect of information systems" (Lusa, Iscoff, McCartney, 1975, p. 41).[9] Perhaps the lack of managerial concentration in most MIS curricula designed in the late 60s and still in use now has been considered the greatest shortcoming observed by MIS managers. One such manager explains, "...we in information services need to get business-oriented" (--Datamation, 1979, p. 60).[10]

In recent years, many articles have been written about MIS curricula and their shortcomings. Most of the authors of these articles agree that the training that MIS students are receiving is not adequate for real life jobs in information systems. Lusa, Iscoff, and McCartney (1975) report that many systems staff are not trained properly in human relations, which, in effect, makes them unable to communicate with other business people. Along the same line, Mandt (1983)[11] argues that schools are graduating students with an overabundance of technical knowledge which rapidly becomes outdated in our changing world. Further, Martine (1985)[12] concludes that MIS programs are teaching obsolete methodologies and irrelevant computer programming languages. This lack of preparation of students graduating from current MIS programs has raised the concern of some researchers, such as Archer (1983)[13], who notes that students graduating from these curricula rarely conform to the expectations of the business world. In summary, one could argue that the major shortcoming of most MIS education in the past has been too much concentration on the technical aspects of computer information systems, and little or none on the behavioral implications on broader issues relevant to MIS's role in business organizations.

Perhaps the major task facing MIS educators today is to review the MIS curricula of the past and present, acknowledge the new developments in information processing technology, anticipate future trends of MIS applications in the corporate world, and, finally, modify MIS education programs to meet the real current and future demands of the workplace.

AN MIS CURRICULUM MODEL

Traditionally, the components of a computer information system have been identified as hardware and software, and systems staff have been mainly oriented toward these two components, ignoring humanistic concerns. The heavy emphasis given to hardware and software, without consideration of the human components of the systems, led to serious management problems, and led many companies to conclude that their information systems efforts had failed. It should be mentioned that the human component of these systems is not limited to people within the systems, but extends to all end-users and anyone within the organization who is affected by the information system. One must consider the organizational impact of information systems, how these systems fit into the organizational structure of the firm, and how the products of information systems can be utilized by managers across the company. Unfortunately, most systems staff are strangers to organizational concerns that cannot be classified as either hardware or software.

A more accurate, contemporary definition of a computer-based information system is that it is a collection of hardware, software, peopleware, and procedures/techniques, surrounded by end-users from all other functional units. Further, there are external factors which have a direct effect on the system as a whole, such as technology, governmental regulations, the economy, other information systems, and available education. Figure 1 illustrates the overall organizational structure of an information system.

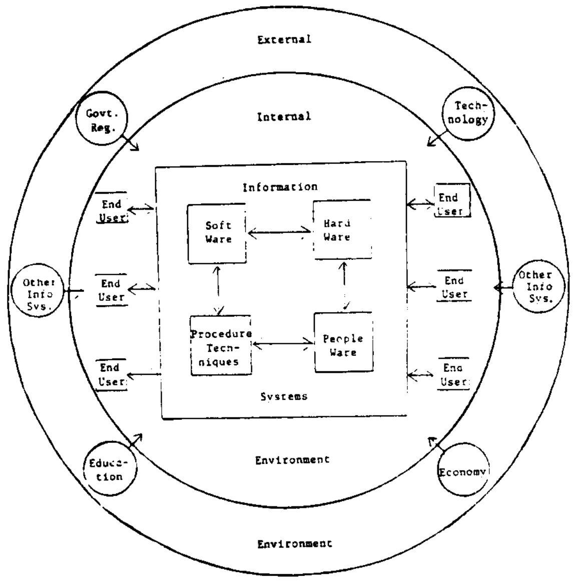


Figure 1: A Framework of Information Systems

Generally speaking, the model MIS curricula of the past has been based on the narrow definition of the computer-based information system as a collection of hardware and software. Now it is time to include other relevant training in the MIS curriculum and offer students a broader definition of these systems. Figure 2 outlines an MIS curriculum which would acquaint students with the broader concerns they will encounter in the systems-related jobs of the future.

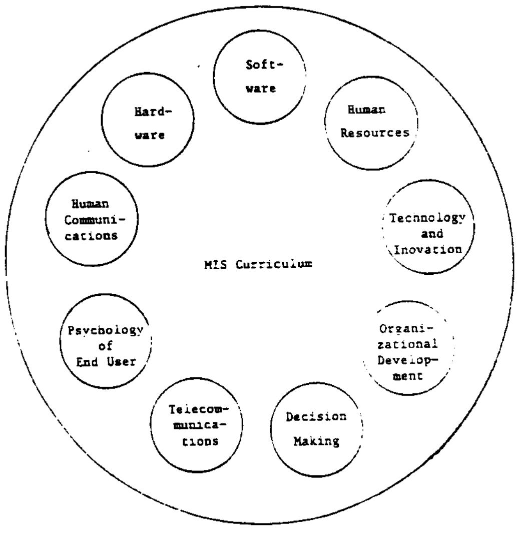


Figure 2: An MIS Model Curricul

Further description of these categories follows:

Hardware

A course or courses offered in this area would introduce students to the various hardware components of the systems, and of computer organizations in general.

Software

This category of the curriculum would mainly relate to computer programming languages, systems analysis and design, databases, office automation, decision support systems, artificial intelligence, and integration of these elements in information systems.

Human Resources

In this area, a class or classes would introduce students to the human aspects of information systems and how they might be managed in more productive ways.

Decision-Making Concepts

In this course, students would study concepts such as the stages involved in decision-making, types of decisions, and use of information systems in decision-making.

Organizational Development

Training in this area would teach students about the organizational structure of information systems, their relationship to other functional areas, and how various units of an organization communicate with each other.

Human Communication

This class would introduce MIS students to various human communication concepts - types of communication channels, methods of communication (e.g., verbal or writ-ten) - and how to communicate with end-users of information systems in effective ways.

Technology and Innovation

These studies would make students familiar with currently available technology, give them a vision of future potential, and teach them how to respond effectively to technological innovations.

Psychology

By studying different concepts in the field of psychology, primarily in the areas of human interaction and human behavior, MIS students would become familiar with the issues involved in the humanistic side of the systems, and it is hoped, learn that people cannot be treated like hardware or software.

Telecommunications

A course or courses under this category would introudce students to the techniques and procedures of telecommunications and networking, and how information systems both within and outside of the firm can be integrated.

CONCLUSION

The fact that many systems staff lack a solid educational background in humanistic and organizational concerns points to one of the major reasons for the past and present deficiencies in management of information resources. This inadequate education can be blamed, in large part, on universities and colleges and their MIS curricula. Most of these curricula are exclusively comprised of courses in quantitative analysis, and hardware- and software-oriented courses, and offer nothing on the organizational implications of information systems. This is, perhaps, a result of the way in which MIS educators have viewed information systems, with an overemphasis on the computer's role in general, and a downplaying of other elements of information resources.

Given the rapid change in information processing technology, and the increasing demands for better utilization and management of information resources, MIS curricula cannot afford to train technicians, operators, or controllers for future information resources management. It is important to make the necessary changes in current MIS curricula, and to train students to cope with all of the components of information resources, in order to satisfy the real MIS staffing needs of corporations - the future employers of today's MIS students. As one MIS executive noted, "...the MIS manager has the technical background; he now must develop management responsibility" (Ferreira and Collins, 1979, p. 60).

REFERENCES

1. Khosrowpour, M. "MIS Leadership in Transition," _Journal of Systems Management_ (November, 1985), pp. 18-21.

2. Ferreira, J. and F. J. Collins, "The Changing Role of the MIS Executive," _Data Management_ (November, 1979), pp. 26-32.

3. Dickson, G. W. and J. C. Wetherbe, _The Management of Information Systems_, New York: McGraw-Hill, 1985.

4. Gupta, R. "Information Manager: His Role in Corporate Management," _Data Management_ (July, 1974), pp. 26-29.

5. Tomeski, E. A. and K. E. Sadek "Job Satisfaction and the Systems Professional," _Journal of Systems Management_ (June, 1980), pp. 6-10.

6. Schlosser, R. D. "Psychology for the Systems Analyst," _Management Services_ (November-December, 1974), pp. 29-36.

7. Carper, B. W. "Human Factors in MIS," _Journal of Systems Management_ (November, 1977), pp. 48-50.

8. Cotterman, W. "A Comparative Analysis of Information Systems Curricula," _ISECON83_, (1983).

9. Lusa, J., R. Iscoff, and L. McCartney "Industry, Colleges Speak to Each Other," _Infosystems_, Vol. 22, No. 10 (October, 1975), pp. 39-42.

10. ________. "Calling a Spade a Spade ...a Chat with MIS Executives," _Datamation_ (November, 1979), pp. 59-62.

11. Mandt, E. J. "The Failure of Business Education and What to do About it," _Management Review_ (August, 1982).

12. Martin, J. "An Information Systems Manifesto," _Communications of the ACM_ (March, 1985).

13. Archer, C. B. "What Does Business and Industry Expect from Computer Science Graduates Today?" _ACM SIGSCE Bulletin_, Vol. 15, No. 1, (February, 1983), pp. 82-84.

INTRODUCTION TO PROGRAMMING AND ALGORITHM DESIGN

Associate CIS Curriculum Track

L. GAIL LEFEVRE

Florida Junior College at Jacksonville
4501 Capper Road
Jacksonville, FL 32218
(904) 757-6307

ABSTRACT

Introduction to Programming and Algorithm Design is an essential first course for CIS majors, which introduces the fundamentals of computer use, including editor functions, file system use, and program compilation and execution. Students are taught algorithm design using structured flowcharts and pseudocode, and the language Pascal is used throughout to demonstrate the theory being taught. This course meets the guidelines of the DPMA Associate-Level Model Curriculum course, Program Design and Development. Since Fall Semester, 1984, thirty sections have been taught to approximately six hundred students by five professors. The CIS faculty has begun to see a measurable improvement in the logical thinking of major students, particularly as they enter the study of COBOL.

TOPICAL COURSE OUTLINE

COURSE TOPICS	CONTACT HOURS
I. Use of the Computer	11 -13
A. Login/Logout Procedure	
B. Editor Functions	
C. File System and File Maintenance	
D. Compiling, Loading and Running Programs	
II. Programming Logic	8 - 10
A. The Concept of an Algorithm	
B. Problem Definition	
C. Top-Down Design	
D. Algorithm Development	
E. Testing and Debugging	
III. Algorithm Design Using Pascal	
A. Introduction	10 - 12
1. Variables	
2. Simple Data Types	
3. Arithmetic Expressions	
4. Assignment Statements	
5. Intrinsic Functions	
6. Input and Output	
B. Selection	5 - 7
1. IF-THEN-ELSE	
2. Logical Operators	
3. Type Boolean	
4. CASE Statement	
C. Repetition	7 - 9
1. FOR-DO	
2. WHILE - DO	
a. Header Technique	
b. Trailer Technique	
3. REPEAT-UNTIL	
D. Modular Design	
1. Functions	
2. Procedures	
E. Data Structures	
1. Arrays	
2. Files	

60 Contact Hours
45 hours Lecture/Discussion
15 hours Laboratory

INTRODUCTION

In Fall of 1984, Florida Junior College at Jacksonville began offering a new required course in the CIS major programs, Introduction to Programming and Algorithm Design. For a number of years before, these topics had been covered in the course Data Processing Math. The CIS faculty had become aware that the content which continually switched between numbering systems, bits and bytes, etc. and problem solving and flowcharts, should be divided into two separate courses. Also, in this original course, program design was demonstrated with the language BASIC, which did not help to teach students concepts of structured programming, top-down design and modularity. Another problem we encountered was that professors of Beginning COBOL were required to spend too much of the semester instructing students in editing, compiling and file maintenance on the Prime 400 minicomputer, our student system.

The DPMA Associate-Level Model Curriculum in CIS, Comp.2: Program Design and Development course description states, "Under modern methods, analysis, design, and development of program logic all take place independently of program coding... In this course, emphasis is on identification and solution of business problems through systems of computer programs. Programs are described and designed through such tools as structure charts and pseudocode. These tools are both logical and universal in that they promote communication between computer professionals and system users." We have developed a COBOL prerequisite course to meet these DPMA guidelines.

Projects

Students complete eight projects which demonstrate the constructs that are universal to computer programming. These projects solve "real world" problems using structured flowcharting, pseudocode and Pascal, and they require the student to edit, compile and manipulate files using the Prime 400 minicomputer. Students are given most of the Pascal code for these projects; the emphasis is on understanding design and structure, not on programming in Pascal.

Project 1, Figure 1, has the goal of introducing the editor, compiler and the concept of a COMO file. Students receive a student manual to the Prime 400 at this time. An analysis of the code of this project is used later in the term, when loops and strings are introduced. Project 2 continues the emphasis on the use of the computer, particularly how to get several files on one hardcopy with the COMO file. Project 3 emphasizes the analysis of variables, simple data types and program elements. Figure 2 is the program analysis assignment sheet which accompanies Project 3.

Projects 4-8 are each assigned in two separate parts. Part 1 states a problem in English, with hints of how to design a structured algorithm to solve the problem. Figure 3 shows the Part 1 assignment for Projects 4 and 6. Part 2 gives most of the Pascal code with requirements for the printout, including the code and run as well as an alphabetized listing of the UFD, and the flowcharts to be analyzed and/or drawn. Figure 4 shows Part 2 for Project 4.

Flowcharts and Pseudocode

Structured flowcharts are used throughout the course to design algorithms. We are using a rigid system of flowcharting which, when learned, allows the student to draw and read flowcharts readily. This ease in flowcharting is a result of control structures which have a different shape for each construct and are pictured in the same manner every time. See Figure 5 for the general form of these flowcharts. Students have been resistant to this formality in the beginning, but most recognize the advantages rather quickly.

It has taken almost two years, but most professors teaching high level languages are now using these flowcharting techniques. This is a decided benefit for students, because it provides consistency as they move through the program. We tell students that not everyone flowcharts in this way, but to aid in learning structure and logical thinking this technique will be required in this course.

Pseudocode is also used throughout the course, but with this tool we encourage individuality and flexibility. We stress that pseudocode is really informal English, and that there are many different ways to express the same idea, assuming of course, that a step-by-step process is defined and that control structures are clear.

Texts

Textbook selection has been difficult, and we are still searching. During the 1984-85 academic year we used Hartling, Druffel and Hilbing, with limited success. For 1985-86 we purposely chose a text without flowcharts, to avoid the problem of re-drawing flow-charts to fit our convention. We are using Wetzel and Bulgren, which is better, but does not completely meet our needs. The topic of Algorithm Design and Problem Solving at the Community College level seems to be "hot" in the Spring of 1986. We are considering the Solow text, but there may be others available before the adoption deadline.

Conclusion

Professors and students have begun to report the positive results of Introduction to Programming and Algorithm Design. Because our college policy states that students graduate under the requirements of the year of admission, it is only in Winter, 1986 that we are seeing classes of Beginning COBOL in which approximately 90% of the students have taken this prerequisite course. The COBOL instructors report that the classes are several weeks ahead of where they would normally be at mid-term. Students are also realizing that the logical thinking practiced in the course helps them to "tie together" concepts being taught in advanced courses such as Data Management and Utilities, and Information Systems.

Plans for the future include offering this course on all four campuses of Florida Junior College, as well as at several off-campus sites. We plan to offer some sections on microcomputers, using Turbo Pascal, beginning in Fall, 1986.

COP 1000 Project 1

LOGIN (follow separate instructions)
Get into the Editor (Input mode) by typing EDIT PASCAL. Enter the following lines of code as given, EXCEPT, in the third line, change 4, to the number of letters in your given name.

```
PROGRAM GREET;
VAR
  NAME:  ARRAY [1..4] OF CHAR;
  COUNT: INTEGER;
BEGIN
WRITE('WHAT IS YOUR NAME? ');
READLN(NAME);
COUNT := 1;
WHILE COUNT  <= 20 DO
    BEGIN
    WRITELN('HELLO ',NAME);
    COUNT := COUNT + 1
    END
END.
```

In Edit mode, type GREET.PASCAL
Make any necessary corrections, in Edit mode.

COMPILE AND SEG (follow separate instructions)

When you have successfully compiled and run the program, create a COMO file.

```
OK, COMO COP
OK, SLIST GREET.PASCAL
(your Pascal code will be listed
here)

OK, SEG GREET
(the program run will be printed
here)

OK, COMO -END
OK, SPOOL COP
```

Pick up your printout from the table in front of the printer. Submit the printout, which includes the code and the run of your program.

DUE DATE______________________________

Figure 1

NAME:____________________
DUE DATE:____________________

COP 1000 Project 3

Use the line numbers beside the lines of Pascal code to answer the following:

Which lines contain internal documentation?____________________
Which statements are assignments?____________________
Which statements contain string constants?____________________
Which are input statements?____________________
Which are output statements?____________________
Which statements contain arithmetic expressions?____________________
List variables and their types:

OUTPUT	INTERMEDIATE	INPUT

Submit:
1) A printout of the COMO file.
2) A flowchart.
3) This sheet with questions answered.

Figure 2

COP 1000 - PROJECT 4

Design an algorithm to accept a student number and three scores from a terminal, and output the student number, grade average, and letter grade. Use a ten point scale for grades 'A' through 'D', and assign 'F' to any average below 60. Allow a user to input values for a class of any size. Output the class average. Hints:
1. Use a FOR-DO loop to handle an entire class.
2. Construct a separate module to determine letter grade.

List variables and their types:

OUTPUT	INTERMEDIATE	INPUT

COP 1000 - PROJECT 6

Design an algorithm to accept four integers from a terminal and, depending on the user's choice, compute the AVERAGE, SUM OF THE SQUARES,or SUM OF THE CUBES of the four integers. Make your design user-friendly. If the choice entered is unavailable, give your user a chance to try again. Hint: Design a separate module for each of the following:
- a. to display a menu of choices.
- b. to input four integers.
- c. to select a course of action based on the user's choice.
- d. to execute each of the choices.
- e. to output the results.

List variables and their types:

OUTPUT	INTERMEDIATE	INPUT

Figure 3

COP 1000 PROJECT 4

Nine lines have been omitted from the following Pascal program. Using the flowchart given, and your code from Project 3, reconstruct the missing lines of code. Enter the program, including your lines.

```
(*PROGRAMMER        name              *)
(*DATE              date              *)
PROGRAM LETGRADE;
(*THIS PROGRAM COMPUTES AVERAGE AND
ASSIGNS LETTER GRADE*)
VAR
  SCORE1, SCORE2, SCORE3, N, STUNUM,
  I: INTEGER;
  AVG, CLASSTOTAL, AVERAGE: REAL;
(***********************************)
FUNCTION GRADE(AVG: REAL): CHAR;
BEGIN
IF AVG >= 90 THEN
   GRADE := 'A'
ELSE IF AVG >= 80 THEN
   GRADE := 'B'
ELSE IF AVG >= 70 THEN
   GRADE := 'C'
ELSE IF AVG >= 60 THEN
   GRADE := 'D'
ELSE
   GRADE := 'F'
END;  (*GRADE*)
(***********************************)
BEGIN (*MAIN*)
WRITELN;
(*nine lines have been deleted here*)
WRITELN('STUDENT # ', STUNUM:1,
'AVERAGE',AVG:7:2,'GRADE',GRADE(AVG));
CLASSTOTAL := CLASSTOTAL + AVG
END;  (*FOR*)
AVERAGE := CLASSTOTAL/N;
WRITELN;
WRITELN('THE CLASS AVERAGE IS ',
AVERAGE:6:2);
WRITELN
END.(*MAIN*)
```

File your program as LETGRADE.PASCAL, compile and test run it.

Create a COMO file to include:
1) the Pascal code
2) one run, using the data given for Project 3
3) an alphabetized list of all your files

Fill in the symbols of the Main flowchart. Copy the function flowchart, as given.

SUBMIT:
1) a printout of your COMO file
2) two flowcharts

Figure 4

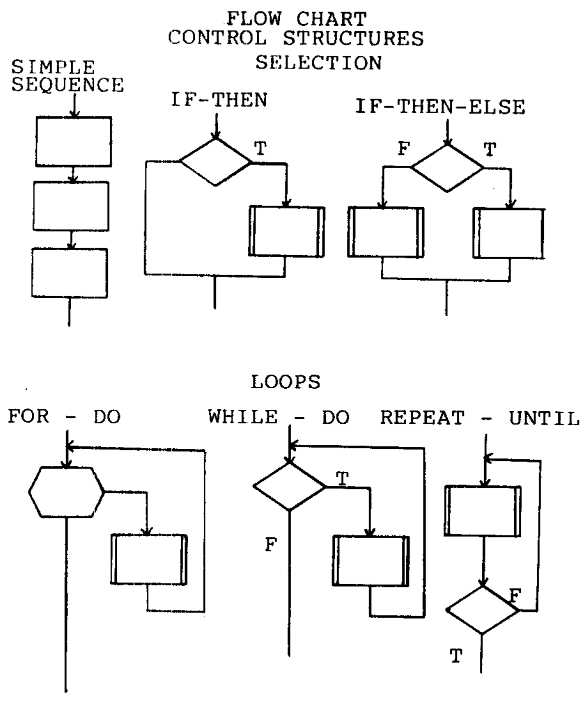


Figure 5

REFERENCES

Hartling, J., Druffel, L., & Hilbing, F. Introduction to Computer Programming: A Problem Solving Approach. Bedford, MS: Digital Press, 1983

Solow, D. Reasoning With a Computer in Pascal. Reading, MS: Addison Wesley Publishing Company, 1986.

Wetzel, G. & Bulgren, W. The Algorithmic Process: An Introduction to Problem Solving. Chicago, IL: Science Research Associates, Inc., 1985.

THE DPMA'S CURRICULUM '86 VERSUS THE ACM'S INFORMATION SYSTEMS CURRICULUM RECOMMENDATIONS FOR THE 80S: HOW DO THEY COMPARE?

Richard Discenza and Fred R. McFadden
University of Colorado at Colorado Springs

This paper compares the new DPMA model curriculum with ACM curriculum recommendations for the 80s. Both recommendations assure schools follow business programs accredited by AACSB. The new DPMA recommendations offer flexibility in terms of employment positioning and a wide variety of courses for non-CIS majors.

Last fall the Data Processing Management Association Education Foundation released an updated version of a curriculum designed to establish standard course offerings to help colleges and universities impart what can be considered the foundational knowledge in the discipline of computer information systems. DPMA's latest curriculum is the culmination of nearly five years of effort on the part of numerous participants from academia and industry. This document called "CIS Curriculum '86" was intended to reflect many of the dynamic changes that have occurred in the area of information systems since the original document was issued in 1981. In this rapidly growing profession there has been a phenomenal growth in the technology of microcomputers, resulting in the installation and applications of millions of desktop computer systems in business education and governmental organizations. Besides the microcomputer revolution, numerous computer-oriented organizations have seen the birth and development of several new concepts in the preparation and use of computer software. These powerful software tools have added new dimensions to the fields of system development and information management. Given this environment, the Data Processing Management Association (DPMA) responded by updating and expanding their undergraduate CIS curriculum guidelines so that educational institutions will continue to provide graduates who can operate and manage this revolution in information systems.

This paper summarizes the DPMA model curriculum for undergraduate CIS education and compares this curriculum with the information systems curriculum recommended by the Association for Computing Machinery (ACM) curriculum committee on information systems. While both organizations offer guidelines for other academic programs (e.g., ACM has graduate guidelines and DPMA has guidelines for secondary curricula and associate level programs in information systems), in this paper we will examine only the four-year baccalaureate programs. For complete details of these guidelines and recommendations, see Nunamaker (2) and the DPMA (3).

THE EVOLUTION OF CURRICULUM GUIDELINES

Curriculum guidelines came about when technological advances in the computer field created a communication gap between business managers and data processing personnel. Educational institutions prolonged the problem by their resistance to change, maintaining that computer science programs rather than business data processing programs should be offered to students. Computer science programs were initially outgrowths of engineering programs, the first area of computer applications. These programs emphasized a scientific orientation and thus had a tendency to misrepresent to students the type of skills which were in tremendous demand throughout the industry. Early attempts by both the DPMA and ACM curriculum guidelines were aimed at reducing the communication problem between managers and information systems personnel. The objective of these programs was to graduate individuals who were knowledgeable in the subject area they would be automating, and at the same time better able to communicate their needs in data processing terms.

Although both organizations began developing curriculum guidelines in the early 1970s for information systems curricula, it wasn't clear whether these guidelines were modifications of computer science programs or whether the programs were aimed at good business data processing practices (1). In 1972 and 1973, the ACM Curriculum Committee on Computer Education for Management sent a survey to 205 business schools that met the American Assembly of Collegiate Schools of Business (AACSB) accreditation standards, to 149 computer science department heads, and to 159 collegiate chapters of ACM. After holding several workshops in which the survey results were discussed with representatives of both business and government, ACM offered the original guidelines of 1972 and 1973. Approximately eight years later, in 1980, a similar survey was carried out and the original ACM Recommendations were revised and updated in a report entitled "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs" (2).

During the same period of time, the DPMA model was developed using a similar type of methodology. This resulted in the CIS Curriculum '81. Early in 1984, DPMA set out to revise the guidelines presented in 1981. The DPMA Education Foundation prepared drafts for the new curriculum and mailed them to approximately 1,000 persons involved in the CIS field and in CIS education during 1985. After incorporating responses to the initial drafts, the

Foundation sent out a revised draft, for further review to approximately 1,400 persons in education and in industry. The responses from these individuals served as a basis for Curriculum '86. As in earlier revisions, both the DPMA model curriculum and the ACM curriculum guidelines were designed to be adopted by colleges of business who are accredited by AACSB (American Assembly of Collegiate Schools of Business) (4, 5).

THE CURRICULUM GUIDELINE USERS

Both the DPMA and the ACM curriculum guidelines have been widely used by the majority of the AACSB-accredited institutions. Pierson, et al. reported on a number of surveys to AACSB institutions showing that both curricula are in widespread use. In a survey sent to 236 schools accredited by the American Assembly of Collegiate Schools of Business in January 1984, eighteen percent of the respondents indicated they used the DPMA model and thirteen percent indicated they used the ACM model; another twelve percent indicated they used a modified DPMA and ACM combination. In addition, almost four percent of the schools used a modified DPMA model, and an additional 2.6 percent indicated that they used a modified ACM computer information systems curriculum (6). In another survey, more than 90% of the four-year academic institutions indicated that they either utilized or plan to utilize the DPMA model curriculum for undergraduate computer information systems education. In this survey, 441 usable questionnaires were returned; this represented 103 four-year public institutions, 85 private four-year institutions, and 253 community and junior colleges (7).

Both the ACM and the DPMA curriculum guidelines help to focus data processing education in programming and in the analysis and design of information systems. Just as programs that stress strong technical but weak organizational skills end up providing inadequate background for analysis and design, those programs which have strong organizational and weak technical skills leave students under prepared to handle the complexities of systems analysis, design and implementation. Both the DPMA and the ACM curricula are designed to prepare students for entry-level jobs in application computer programming and in the analysis and design of information systems, with the aim of insuring a sufficient foundation to advance to any of a wide variety of career paths. When employers are aware that a college they deal with has this type of curriculum, then they know they have a valuable resource to tap. In addition, programs with these guidelines can offer students the opportunity to become more successful in their careers because of the concentration of study. Furthermore, these graduates' training period at any installation will be much shorter because they are already familiar with a good deal of what's going on and will really have to learn only a particular organization's equipment.

DIFFERENCES AND SIMILARITIES IN THE COMPUTER INFORMATION SYSTEMS CURRICULA

Keeping in mind that the DPMA model curriculum represents the most recent version of an approach to undergraduate computer information systems, the ACM model will be at a slight disadvantage because it originally came out in the early 1980s. Table I provides a list of similar courses that are offered in both the DPMA and ACM curricula. The ACM curriculum of core courses contains 8 courses, beginning with a computer concepts course and ending with an information systems project course. Table I points out that there is a great deal of similarity between the DPMA approach to computer information systems undergraduate program and ACM's. The real differences in the programs can be seen in Table II, which shows some additional courses which are offered in the curriculum '86 of DPMA. In addition to the core courses listed in Table I, there are two additional courses in the DPMA model that stress today's type of information processing: microcomputer applications in business and the information center functions. The big difference in the two model curricula is shown in Table II, in the number of elective courses in the DPMA model curriculum. Again, these courses deal with concepts, ideas, and techniques that have largely emerged since the early 1980s and have proven themselves to be very much a part of the field as it stands today.

TABLE I

List of Similar Courses

DPMA		ACM	
No.	Description	No.	Description
CIS/86-1	Introduction to Computer I.S.	IS1	Comp. Concepts & Software Sys.
CIS/86-3	Intro. to Business Applications Progr.	IS2	Program, Data, & File Structures
CIS/86-13	Info. Resource Planning & Management	IS3	Info. Systems in Organizations
CIS/86-6	Data Files & Data Bases	IS4	Data Base Management Systems
CIS/86-5	System Development Methodologies: A Survey	IS5	Information Analysis
CIS/86-9	Advanced Office Systems	IS6	Data Communication System Networks
CIS/86-8	Systems Development Project	IS8	System Design Project
CIS/86-19	Systems Development Project with Information Center Techniques	IS10	Information Systems Project

TABLE II

Additional Courses in the DPMA Curriculum '86

	Core Courses
CIS/86-2	Microcomputer Applications in Business
CIS/86-4	Intermediate Business Application Programming
CIS/86-7	Information Center Functions
	Elective Courses
CIS/86-9	Advanced Office Systems
CIS/86-10	Computer Graphics in Business
CIS/86-11	Decision Support and Expert Systems
CIS/86-12	Artificial Intelligence in Decision Making
CIS/86-13	Advanced Business Applications Programming
CIS/86-14	Computer Control and Audit
CIS/86-15	Distributed Intelligence and Communication Systems
CIS/86-16	Programming Languages: Procedural, Nonprocedural, and Fourth Generation
CIS/86-17	Computer Hardware, System Software, and Architecture
CIS/86-20	CIS Communication, Reporting, and Documentation Techniques

As indicated earlier, there has been a tremendous increase in the use of microcomputers in computer information systems, and to reflect this movement a number of courses have been incorporated into the DPMA model curriculum. Table III lists the courses that would make direct use of a microcomputer and its related equipment. Currently, the ACM model curriculum has very few complete courses which allow incorporation of microcomputers into a CIS program. However, for schools that are using the ACM model curriculum, there are plenty of opportunities in current courses to incorporate micro applications, concepts, and technology into a student's program of study.

TABLE III

Courses Which Specifically Include Microcomputers

	DPMA '86 Model Curriculum		ACM Model Curriculum
CIS/86-1	Introduction to Computer Information Systems	IS6	Data Communications Systems and Networks
CIS/86-2	Microcomputer Applications in Business	IS10	Information System Projects
CIS/86-9	Advanced Office Systems		
CIS/86-10	Computer Graphics in Business		
CIS/86-19	System Development Project with Information Center Techniques		

Table IV points out that the ACM curriculum may also be somewhat lacking in the utilization of 4th-generation software packages. Again, because of the recent use of these very productive software packages, the DPMA '86 model curriculum has a much greater emphasis on their inclusion. While the ACM model specifies three courses where 4th-generation software can be readily used, the DPMA '86 model curriculum has ten courses that specifically utilize 4th-generation software, from both a user and a professional information systems specialist point of view.

TABLE IV

Courses Which Specifically Include Fourth Generation Software

	DPMA '86 Model Curriculum		ACM Model Curriculum
CIS/86-1	Intro. to Computer Information Systems	IS4	Data Base Management Systems
CIS/86-2	Microcomputer Applications in Business	IS6	Data Communications Systems and Networks
CIS/86-7	Information Center Functions	IS10	Information Systems Projects
CIS/86-9	Advanced Office Systems		
CIS/86-10	Computer Graphics in Business		
CIS/86-11	Decision Support and Expert Systems		
CIS/86-12	Artificial Intelligence in Decision Making		
CIS/86-13	Distributed Intelligence and Communication Systems		
CIS/86-16	Programming Languages: Procedural, Non-Procedural, and Fourth Generation		
CIS/86-19	Systems Development Project With Information Center Techniques		

CONCLUSION

The comparison of the two model curricula shows them not to be on an equal footing, because of the recent DPMA revision. Some trends in the DPMA model seem to be appearing, and these seem to provide a robust set of program guidelines. While earlier DPMA recommendations centered on producing graduates who were strong in entry-level position skills, the latest revision offers the flexibility of providing graduates with the skills important for more advanced CIS. This has been a traditional strength of the ACM model curriculum. The DPMA '86 guidelines now offer concepts, tools, and techniques important to the sysems analyst and other higher level CIS positions.

A second characteristic of the DPMA model is not only the flexibility that it offers in terms of employment positioning, but the flexibility of offering courses useful to a wide variety of majors on a college campus. While many of the courses call for prerequisites specific and necessary to CIS majors, other courses have appeal to non-CIS majors in the College of Business and to majors scattered through a university. These include individuals who may wish to take courses on a non-degree basis and to graduates who wish to maintain and improve their CIS skills. One will have to wait and see whether ACM's revised curriculum will offer the flexibility of having more entry level orientation, or whether they will continue with the upper level orientations, stressing the systems analyst orientation.

REFERENCES

1. Rhodes, Wayne L. Jr., "Grappling With Winds of Change," Infosystems, Vol. 27, No. 6, (June 1980), p. 72.
2. Nunamaker, J. F. (ed.), "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs," Communications of the ACM, Vol. 25, No. 11, (November, 1982), pp. 781-805.
3. DPMA Model Curriculum for Undergraduate CIS Education (Curriculum '86), 1st ed., (October, 1985), Park Ridge, IL: Data Processing Management Association, 56 pp.
4. Lorents, Alden C., "The DPMA Model Curriculum and AACSB Accreditation," Proceedings of the Fourth Annual Information Systems Education Conference, Houston, TX, (October 26-27, 1985), pp. 33-35.
5. Couger, J. D. (ed.), "Updated Information System Curriculum Guidelines," Computing Newsletter, Vol. XV, No. 8, (April, 1982), p. 1.
6. Pierson, Joan K., Jack Shorter, Jeretta A. Horn, and G. Daryl Nord, "Profile of Introductory Data Base Courses Offered at AACSB-Accredited Institutions," Proceedings of the Fourth Annual Information Systems Education Conference, Houston, TX, (October 26-27, 1985), pp. 29-32.
7. Souder, H. Ray and D. R. Adams, "Survey Reveals Model Curriculum Impact in Academia" Data Management, Vol. 22, No. 2, (February, 1984), pp. 40-41.

DPMA AND END-USER EDUCATION

Donald L. Dawley, Department of Decision Sciences, Miami University of Ohio

ABSTRACT

End-user computing is important today and will become increasingly more important in future years. Businesses that extensively use this technology must face major decisions and problems that can adversely impact the information resources of the firm. This paper provides a discussion of end-user computing, its associated problems, and suggests that DPMA should establish guidelines for end-user programs at all educational levels. A guideline is provided for the four-year undergraduate program.

INTRODUCTION

The major purpose of this paper is to support the need for guidelines for end-user educational programs at all levels. The paper also proposes an end-user minor or concentration for four-year undergraduate programs. The need for educational guidelines for end-user programs is supported by the pervasive business use of end-user computing and the major problems confronting the businesses extensively using this technology. In last year's proceedings, Dr. Doris Duncan proposed that DPMA provide a guideline for an MIS emphasis within the Master of Business Administration (MBA) program or an in-depth Master of Science (MS) in management information systems (MIS) (1). The term MIS is used to distinguish end-user computing from the existing computer information system (CIS) notation used for the existing DPMA programs. The discussion that follows supports and expands Dr. Duncan's thesis. Briefly, the paper outlines the technological trends that have made end-user computing possible, problems associated with end-user computing, and the educational program that are needed to support the effective continued development of end-user computing.

THE UP SIDE OF END-USER COMPUTING

The trends in computing technology that have led to end-user computing have been surprisingly consistent over a number of years. The trends have been evolution ary and have in large part come about as a result of business needs. Examples include:

1. reduction in the cost of processing a given number of transactions
2. increased memory capacity of internal and external storage
3. reduction of power requirements, footprints, heat generated, and environmental constraints e.g. temperature, and humidity
4. improved hardware functionality, reliability and maintainability
5. higher level (more English like) languages for DP and end-users
6. reduced technical knowledge required to use effectively a large range of hardware and software (user friendly)
7. greater amounts of functional knowledge built into languages, hardware, and software
8. dramatic increases in software choices, including sources, integration, and specialization
9. improved communications among equipment, software, and people.

These trends have positioned computer technology so that it is easily within the grasp of businesses of all sizes and useful to professionals in most, if not all, business functional areas. It is common knowledge that businesses of all sizes are increasingly using this technology to improve their productivity, competitive edge, and image. It is difficult to envision a data processing (DP) professional who would not readily acknowledge the growing importance of end-user computing. James Martin and others have made a strong case for end-user computing professionals to become as important as DP professionals by the end of this decade (2). The trend toward end-user computing is expected to continue for as far as can be seen into the future.

THE POTENTIAL DOWN SIDE OF END-USER COMPUTING

The growth of end-user computing by itself may not justify the need for educational programs or DPMA involvement. However, the potential adverse impact of end-user computing upon the business MIS strongly supports the need for formal computing education and the need for computer professionals to be involved in end-user computing. Many businesses have already recognized the need to involve MIS with end-user computing. In many companies, MIS has already stepped in to provide end-user training, technical support, guidance, and trouble shooting. Special organizations such as the information center have been established for this purpose. Many businesses have also recognized the need to hire functional managers who have substantial computer expertise in addition to their functional specialties. Businesses then use these people in several ways. One, they use them in their respective functional areas as a local source of computer expertise. Two, they serve as facilitators or liaison between their respective functional areas and DP. Three, they are hired to work in information centers set up to provide assistance to end-users.

There are compelling reasons for businesses to seek computer competent professionals to oversee, guide, assist or manage end-user computing. First, end-user computing is expensive and can result in the needless waste of money and valuable human resources if not managed by knowledgeable people. Recent industry-wide surveys indicate that businesses are spending about half as much on end-user computing as they are on mainframes (3). Second, the business data base and computer systems are vital to a business's daily operation, survival, and success. These vital resources, therefore, must be protected from intentional or accidental alteration or destruction. Finally, end-user computing raises a variety of new problems as well as recasts some existing problems into a new setting. A few of these problems are listed in abbreviated question form.

1. What applications should be relegated to end-users? Who decides? What procedures? Who enforces the policies?

2. Does a business try to minimize duplication? If so, how?

3. End-user hardware, software, and data security? Who controls access, back-up, and documentation of data, in-house programs, and purchased software. What about unauthorized files?

4. What standardization is necessary? Who makes the decisions and what procedures should be followed? This relates to equipment, commercial software, data files, record formats, data element definitions, and end-user developed applications (testing, documentation, maintenance, user interfaces, and quality).

5. Who is responsible for purchasing, warranties, and licensing?

6. Is it economical and feasible for firms to train all of their computer users end-user computing skills? Why not hire needed functional professionals from schools that have effective end-user educational programs in place?

The foregoing questions are serious and have substantial cost and operational consequences. The questions are of particular significance when fourth generation languages (4GL) are extensively used. The technical dimensions of these questions, along with their potential impact upon the business MIS, the growing investment in end-user computing, and the future prognosis should lead managers, DP professionals, and educators to some very definite conclusions:

1. DP management must understand end-user computing, the trends, and play an active role in facilitating and managing it if the best interests of the firm are to be served.

2. The CIS curricula must address the role, benefits, and problems associated with end-user computing.

3. Business professionals need to be competent in end-user computing.

4. Businesses need a critical mass of professionals within each functional area who have significant computer skills and an understanding of how computer technology is used within their respective areas e.g., spreadsheets, data base management systems (DBMS), decision support systems (DSS), and specialized software such as IFPS, SAS, and GPSS.

5. Business schools must provide educational programs that assure that all functional professionals are computer competent with respect to the use of computers within their major field. To achieve this level of competence, business schools need programs at the undergraduate and graduate levels that require MIS courses taught by computer professionals as well as the use of computers in functional courses as appropriate.

6. DPMA should continue to take the leadership role and provide guidelines for the MIS courses at the undergraduate and graduate levels.

END-USER GENERAL PROGRAM GUIDELINES

Since the end-user bodies of knowledge are business and CIS, it would seem logical that the programs should fall into the domain of schools of business administration. At this point it should be evident that graduates from any major business school program should be competent in the end-user computing areas. Major programs would include undergraduate two and four year programs, masters of business administration, and doctoral programs. The following general program guidelines would seem appropriate.

1. The program should be in the business school since the relevant bodies of knowledge are business and CIS.

2. The educational programs must be consistent with the overall university and business school philosophy and rigor.

3. Program content should include mainframe and microcomputer knowledge and experience.

4. Ideally, the programs should be designed to require significant computer usage during every educational term.

5. Programs must have general business, functional, and end-user computing credibility. Therefore, the program

must be designed to provide substantial knowledge and skills in business generally, a selected functional major, and in end-user computing.

6. MIS program should be developed, administered, and taught by highly qualified CIS faculty and emphasize:

   A. concepts that have long-term functional management implications.
   B. the development of useful end-user skills.
   C. courses current with technology which provide instruction on trends.

7. Non-MIS courses having computer content should be taught by highly qualified professionals who have recognized expertise in the use of computers within their respective functional areas.

8. Programs must assure continued currency of graduates through emphasis on professionalism.

   A. Require knowledge of and consistent use of professional literature.
   B. Require knowledge of and encourage participation in professional organizations.

SOME SPECIFIC PROGRAM SUGGESTIONS

It is understood that the specific programs implemented by a school will vary based on educational philosophy (liberal arts vs. professional program), established programs, the educational term (semester vs. quarter), their market, their ability to attract qualified faculty and students, and how they are organized and operate. However, the programs must address the computing body of knowledge, relevant technologies, and developments within end-user computing. As stated earlier, specific programs are required for two and four year undergraduate programs, MBA programs, and doctoral programs for non-CIS majors. This paper will address only MIS (end-user) educational curricula for the four-year undergraduate programs. Due to space limitations, the course content suggested for the four-year program is brief. The course timing is indicated by the following notations:

100	level courses can be taken by freshmen or higher level students
200	level by sophomores or higher
300	level by juniors or higher
400	level by seniors
400/500	level by seniors and MBA students. There would normally be extra project and research work expected of MBA students.

All of the courses listed are assumed to be three semester hours, unless otherwise indicated. Adjustments would have to be made for schools using different academic terms.

Courses can be dual-listed between MIS and CIS.

THE FOUR-YEAR BACCALAUREATE PROGRAM

The four year program is designed for professionals in all of the business disciplines. The program is designed to complement the functional curricula and also is designed to have sufficient scheduling flexibility so that the major and MIS minor or concentration can be completed within the four-year period. The program courses are time-phased so that students can apply their functional knowledge within the MIS courses e.g. DBMS, systems development, and 4GL. This planned use of functional knowledge within the MIS courses strengthens the students' competence in computing and their other courses in a number of ways. One, the students are required to apply their functional knowledge in MIS courses. The result is an increased knowledge and understanding of both subject areas. Two, MIS faculty can use more realistic business examples in applications and projects without the need to take valuable classroom time to explain the business concepts involved in the examples. Three, non-MIS faculty teaching courses requiring computer content can spend more time teaching the function course content and not the details of how to use computers and software. Last, this approach to end-user education further distinguishes the MIS program from the more technical oriented computer science programs.

Required Courses for the Undergraduate MIS Minor

1. MIS 101, Introduction to Computers. This is a four semester hour course. The concepts portion of the course would be nearly identical to CIS/86-1 but would be oriented toward end-user computing (4). The emphasis would be on topics that have user impact e.g., to access data. The course consists of:
   A. 32 hours of MIS concepts
   B. 32 hours of hands-on experience with word processing, DOS, BASIC programming, spreadsheets, and DBMS. The focus of the hands-on experience is to develop skills in the user microcomputers.

2. CIS/86-3 Introduction to Business Application Programming. The course should be taken the last half of the freshman year. See the DPMA model curriculum for course content (4).
   Prerequisite: MIS 101 or equivalent.

3. MIS 201, End-User Data Base Management Systems. The course addresses microcomputer and mainframe DBMSs with an emphasis on the management implications and end-user capabilities. No compiler code is written.
   A. 32 hours of concepts of importance to end-users.
   B. 16 hours of hands-on use of microcomputer and mainframe DBMS software such as dBASEIII, RBASE 5000, and SQL.
   Prerequisite: MIS 101 or equivalent.

4. MIS 301, Management Information and Decision Support Systems. This course introduces the student to the business information systems environment. It addresses the nature of the environment, the structure and characteristics of the MIS, and the responsibilities of managers and end-users including policy formulation, DP planning, and requirements specification. It also addresses DSSs and emerging technologies that have an impact on the business computing environment.
   A. 32 hours of concepts
   B. 16 hours of hands-on DSS using IFPS and other DSS packages.
   Prerequisite: MIS 101 or equivalent.

The last two required courses must be taken in sequence during the senior year. The sequence requires team projects that span a full year. The projects require that each team use a 4GL such as Focus or Mantis to develop an operational system, document it, and present it to top management for implementation. The content of the courses included major parts of CIS/86-5 Systems Development Methodologies, CIS/86-7 Information Center Functions, CIS/86-8 Systems Development Project, and CIS/86-19 Systems Development Project with Information Center Techniques. The emphasis is on 4GL, project management, analysis and design methodologies, and on gaining proficiency with a microcomputer and mainframe based 4GL.

5. MIS 400, Business Systems Analysis Using 4GL.
   Prerequisite: MIS 101, CIS/86-3, MIS 201, and MIS 301.

6. MIS 401, Business Systems Design Using 4GL.
   Prerequisite: MIS 400.

Elective Courses for the MIS Undergraduate Minor

1. CIS/86-2 Microcomputer Applications in Business. See the DPMA model curriculum for content (4).

2. Presentation Graphics in Business. This course is similar to CIS/86-10 but uses commercial software and requires no programming.

3. CIS/86-9, Advanced Office Systems.

4. CIS/86-11, Design Support and Expert Systems.

5. CIS/86-18, Information Resource Planning and Management.

6. MIS 350 Communications and Distributed Systems. This course is similar to CIS/86-15 but at a lower technical level. Greater emphasis is given to LANs including projects that require students to determine requirements, survey available hardware and software, and work with an existing LAN lab.

SUMMARY AND CONCLUSIONS

There is little doubt that end-user computing is important today and will become increasingly more important in the foreseeable future. The 1985 revised Model Curriculum has addressed end-user education from the CIS point of view and a dialogue has been started for an MIS program at the master's level. It is important that DPMA continue to stay abreast with computer technology and provide the needed leadership in the end-user computing area. Educational guidelines are needed at all educational levels if universities are to continue to meet the computing needs of businesses.

SELECTED REFERENCES

(1) Duncan, Doris G. DPMA Graduate Model Curriculum in Management Information Systems: a preliminary report. Proceedings of the Fourth Annual Information Systems Education Conference, Houston, Texas, October, 1985, pp. 8-10.

(2) Application Development Without Programmers, James Martin, Prentice Hall, Inc., Englewood Cliffs, N.J., 1982.

(3) Beaver, Jennifer E. "Bend or Be Broke," Computer Decisions, December, 1984, pp. 131-137.

(4) DPMA, The DPMA Model Curriculum for Undergraduate Computer Information Systems, Park Ridge, Illinois, October 1985.

ETHICS EDUCATION IN INFORMATION SYSTEMS

Robert N. Linebarger
Computer Science Department
Brigham Young University
Provo, Utah

Gary L. Blatter
Vision Quest, Inc.
Orem, Utah

ABSTRACT

Computer based information has become a valuable commodity. Its lack of physical form, and it's growing importance as a valuable product invites misuse and abuse. The legal framework for addressing the rights and obligations associated with information is considered. The concepts of ethics and professional conduct for information system professionals are introduced and profiled. The ethical guidelines and practices of other professions are presented for comparative reference. Specific recommendations for ethics education in the Information Systems profession are presented including curricula, integration and programs. The impact of an ethics education is treated. The paper concludes with a summary of existing and future ethics problems and a prediction of the consequences of the continued lack of ethics education in the information systems profession.

PROBLEM DESCRIPTION

Information as a commodity

Within the past thirty years computer based information has developed into a useful and valuable product providing great advantage to it's possessor, i.e. it has become a commodity. After the materials of information (unstructured data) have been assembled they can be manufactured into valuable information products. These information products are then brought to market like other commodities. However, unlike other commodities in our society, information can exist without physical form. This intrinsic characteristic makes information appear to the user as if it is a product without an associated cost of development, manufacturing or marketing. Hence, society fails to treat information commodities in the same fashion it treats other commodities--giving rise to the abuse of an owners rights in his or her information product.

This abuse and illegality, as many abuses of commodities are illegal, has taken on epidemic proportions in our new information society. Reports appear with an increasing frequency of violations of software copyrights, illegal entry into computer systems, interception of data transmissions, and the misuse of data stored on a computer. These problems are not confined to the narrow segment of our society which we call the computer hacker, but they appear to be widespread both within and without the information industry. Each year information abuses receive greater visibility as ways in which information is abused become more serious.

Legal framework

The problem of curbing information abuse grows tougher in the absence of legal protections. Although legal protections for software and other forms of information not packaged in a traditional form have begun to surface; the law always lags technology. This lag time is due to the inability of legislatures and courts to establish standards before they obtain a clear understanding of the technology and the issues involved. While the legal system lumbers through the technology problem, it denies protection from abuse, theft, monitoring, copyright violations, and illegal entry to an expanding information society. Thus, protection under the law for illegal acts involving computer based information is either slow to action, unenforceable, or unavailable.

While this situation is changing, the rate is exceedingly slow - resulting in the potential for increasing abuse of and illegal acts involving information. This coupled with rapid changes in the computer industry creates a vacuum of protection for computer based information and poses a very serious threat to continued growth in all sectors of the information systems industry.

Even though it is convenient to lay the blame for the lack of legal protections on the legal community it is unjustly delivered. Every technologist who has ever dealt with a representative of the legal community can appreciate their lack of experience with technology. If the legal community is to set standards they must be properly informed about the problems and the impact of potential solutions. Who will inform the legal community? The next section explores ways in which other professionals attempt to solve the particular problems that arise in their fields of expertise.

PROFESSIONAL TRAINING IN ETHICS

Each profession is entrusted by society with an area of expertise. Part of this trust includes the obligation that professionals regulate the conduct of those practicing their profession. This section identifies ways in which other professions deal with problems that arise in their areas of expertise.

Law

The legal community faces troublesome questions in the practice of its profession. For example, a serious question of appropriate conduct arises when a client confides to his attorney that he is guilty of the crime for which he is charged. The attorney's response to this situation affects not only the individuals involved but society in general. The treatment of ethics and professional conduct is a cornerstone of legal education. From the very first introductory course in law, the student is immersed in questions of ethics and professional conduct. As the education process proceeds, the student of law is presented with broader and stronger frames of reference involving professional ethics and conduct. Training in ethics is presented in both informal and formal formats--including a required ethics examination before admission is granted to the practice law in many states.

Medicine

Physicians also face problems in the practice of their profession. Realizing that society has entrusted medical professionals with a sacred trust, medical students are constantly presented with questions of ethics and professional conduct as they are taught the details of medical science. As the educational vehicle turns from formal classroom instruction to clinical experience, the visibility of professional ethics and conduct is strengthened by both informal training as well as formal course work.

Accounting

Both undergraduate and graduate training in accounting contains a strong component of ethics education. The practice of accountancy involves a high degree of ethics and professional conduct in both the procedures of accounting (acceptable methods) as well as the handling of client information, resulting in a strict code of ethics for accounting professionals. The privacy of client information is considered a cornerstone of accounting practice and accounting students are introduced to these concepts early in their education.

Management

Education in business management has always contained a strong component of training in ethics. In recent surveys of business school faculties, many professors have expressed the need for additional training for business students in ethics. Increased emphasis is now being given to an integrated frame of reference for ethics training in capstone courses at both the undergraduate and graduate level.

ETHICS IN INFORMATION SYSTEMS

Codes of Ethics

Codes of ethics for the information industry have been generated by various domestic and international organizations. In the United States, several professional societies that been active in generating codes of ethics for their members, including: The Association for Computing Machinery, The Institute of Electrical and Electronic Engineers, The Data Processing Management Association, The Institute for Certification of Computer Professionals, and The Independent Computer Consultants Association.

Each of these organizations has identified cannons of ethics and professional conduct for their members. These codes of ethics specifically outline the conditions for ethical procedures and professional conduct. The concept of the treatment of information as a commodity of value is intrinsic to the rules and principles of each of these codes. In addition to general ethical considerations, these codes contain references to the principles of professional conduct in handling valuable information property.

Education

While the information industry has followed the lead of the other professions in the development of codes of ethics it has failed to take the next step: Education of its members in the content and application of these codes. In order to instill in the information profession, a frame of reference similar to other professionals specific ethics training must become a part of the general curricula in Information Systems. This educational focus should include traditional in-class training of at least one semester hour early in a student's study program. The course should be required as part of the student's fundamentals training and it should be taught by senior faculty members familiar with the application of ethics within the information profession.

Curricula

The curricula should be composed of modules which cover the breadth of problems currently appearing as well as frames of reference which can be used to deal with problems not yet encountered. The content should include at least the following modules:

1) Information as a commodity - value, development, treatment, marketability.

2) Legal treatment of information - copyrights, trade secrets, non-disclosure agreements, patents.

3) Intellectual content - authorship, ownership, employee/employer agreements, concept portability.

4) Responsibility of handling information - secrecy, security, protection, visibility, procedures.

5) Obligations - responsibilities, liabilities, group and individual negligence.

6) Privacy issues - data sensitivity, privileged information, modification, data havens, file linkage, errors.

7) Authentication issues - authorship, source recognition, identity of sender, identity of recipient.

8) Theft and electronic burglary - software duplication, illegal access to files, network break-in.

9) Information exchange - transborder data flows, exportation, customs, duties, smuggling, value added, taxes.

Delivery vehicles should include both formal lectures on principles as well as group discussion of prepared case studies and reference readings. It is particularly important that outside guest speakers be included as part of the instructional delivery in order to provide real life role models for ethical handling of information and professional conduct.

It is recommended that the general principles of ethics, the proper handling of computer based information and professional conduct be integrated into all Information Systems courses taught after the required ethics course. The issue of ethics and professional conduct must be integrated into the general information systems curricula as well as taught as a specific topic in order to generate the correct attitude toward ethics by the student.

CONSEQUENCES

Stabilization

The inclusion of ethics education in the study of Information Systems can have a powerful and positive impact. In the absence of a stable legal frame of reference, a strong code of ethics is needed by all segments of the information profession. Information workers are often placed in a position of confusion regarding the proper proceedures for handling information. Since the legal framework for handing information is not presently well defined, proper procedures may depend upon one's ethics rather than a set of legal rules.

Presently we find the information systems professional in the following environment: rapid change in both standards and technology, little structured to the legal framework for handling information, a high value in information commodities, and little consideration given to ethics or conduct. In this environment the opportunity for abuse of information is excerbated - with the end result strongly affecting the image of the industry profession and the computer based information worker. Ethics and ethics education should be the stabilizing force in this environment.

If the information worker is confused by the lack of the frames of references for the handling of information then the non-information worker must be in an oblivion about the treatment of information. Providing and teaching these frames of reference is part of the obligation placed on the information profession by society.

Professional Recognition

Another advanatages or distrastrous consequence of ethics education is the recognition by society of the information industry as a profession. Occupations in data processing are treated increasingly as separate from other occupations. A hallmark of an occupation being treated as a profession is the establishment of professional associations and standards of conduct for their members. These standards of conduct are not only important for those to whom they apply but also to the whole of society. Society is interested in insuring that those exercising power as a professional use it without succumbing to the temptation to abuse that power. The inclusion of ethics education in Information Systems can significantly aid the establishment and recognition of Information Systems as a profession in our technical society.

Outside Constraints

The absence of professional conduct in the handling, treatment and recognition of information as a valued commodity runs counter to well established principles of ownership, risk, reward, and fairness in our democratic society. Continued abuse of computer based information, coupled with an absence of professional conduct education will surely result in raising the visibility of the problem to society--who eventually pays the bill. Society at this point must of necessity assume the role of regulator of the information profession which would result in the imposition of outside constraints and procedures upon Information System professionals--either through litigation, regulation, or economics (or all three combined!).

In the absence of self imposed standards for ethics and professional conduct other forces, uninformed about the technology considerations, i.e. legal, economic, and political, will impose their standards of operation. These constraints may involve the impostion of licensing, bonding, insurance, labor relations, and bargaining requirements upon the information systems specialist.

SUMMARY

Attitude

The consequences of the present trends appear only too obvious. Since the computer industry is relatively new in our society there is a attitude of indifference toward ethics and professional conduct among computer professionals. The pace of change in the industry, coupled with the intensity of technical training required for an individual to remain competent has created a myopic attitude toward other important aspects of our profession. The degree of time, effort, and emphasis given to mastering the procedures of Information Systems completely overshadows any consideration of the ethical principles involved in such procedures.

Training

What is missing is a formal treatment of ethics education within the Information Systems curricula. Such training will generate frames of reference for ethical conduct, responsibility for actions, and professional treatment of information as a commodity. It is as important a component of our professional training as any technical concept. Indeed, it may yet prove to be the most important component of our training - one which allows us to practice our profession in a continued environment of unrestrained challenge and application.

Ethics education is conspicuously absent in Information Systems education, and its absence invites control of our profession from other forces in society. We therefore call for a concerted effort by educators and information professionals at all levels to install ethics and professional conduct as a cornerstone of Information Systems education and the Information Profession.

BIBLIOGRAPHY

1. Aiken, R."Reflections on Teaching Computer Ethics," ACM SIGCSE Bulletin (Vol. 15 No. 3), Sept 1983. pp. 8-12

2. Adams, D.R. and T.H. Athey, editors; "DPMA's Model Curriculum for Undergraduate Information Systems Education", Data Processing Management Association, Park Ridge, IL Sept 1981

3. Constitution of the Association for Computing Machinery, Communications of the ACM, July 1980, pg. 425

4. Friedman, L.W. and H.H. Friedman. "An Effective Method for Teaching Ethics in a Computer Course," Interface, Spring 1984, pp. 62-64

5. Parker, Donn B.; "Ethical Conflicts in Computer Science and Technology," AFIPS Press, Arlington, VA 1978

6. Abshire, G.M. "A philosophical Discussion on the Ethics of Computer Professionals," Computers and Society, July 1983

7. Hoffman, W.M., and J. Mills Moore (ed.), "Ethics and the Management of Computer Technology," Cambridge, MA 1981

8. Akers, J.F. "Keynote Address," 1984 National Computer Conference, Las Vegas, Nevada. IBM Corporation, Armonk, NY 1984

9. Heide, D. and J.K Hightower, "Organizations, Ethics and the Computing Professional," Journal of Systems Management, Nov 1983, pp. 38-42

10. Code of Ethics, Independent Computer Consultants Association, St. Louis, MO 1984

11. Johnson, D. G., Computer Ethics, Prentice-Hall Inc., 1985

12. Nunamaker, Jay F., et. al., (ed.), "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs," Communications of the ACM, Nov 82, pp. 781-805

13. "Code of Professional Ethics." AICPA Professional Standards (Volume 2), American Institute of Certified Public Accountants, New York City, NY 1985

14. "Code of Ethics." Data Processing Management Association, Chicago, IL. 1986

15. "Principles of Medical Ethics." American Medical Association, Chicago, IL 1984

16. "Model Rules of Professional Conduct," American Bar Association Journal (Vol. 69), November 1983, p. 1671 f.

Revised Information Systems Curriculum and Training for a Small Graduate Business School

James B. Pick

Graduate School of Management,
University of California,
Riverside, California 92521

ABSTRACT

This paper discusses a recent revision in the information systems curriculum in a small graduate business school, not offering an MIS concentration. The school has a high proportion of micros per student. The curriculum revision, being fully instituted in 1986, includes a revised core IS course, an expanded set of advanced IS courses, expanded computer internships, required micro labs, and optional micro training seminars. The changes are based on the philosophy that IS curriculum and training in a smaller school should provide a basic set of IS concepts to all students, with a limited number of more advanced IS courses available; that the IS curriculum should instill hands-on skills for micros and minis; that micro skills are best taught through a multi-faceted approach; and that the curriculum should be carefully integrated with the computer facility resource.

I. INTRODUCTION

The Graduate School of Management (GSM), University of California Riverside (UCR), consists of 18 faculty and about 100 students. It supports a 92 unit M.B.A. curriculum, offering concentrations in finance, marketing, and general areas. Students are required to write a thesis. In Fall 1987, the School will commence a Ph.D. program in financial economics. In addition, the school administers an undergraduate business administration major to over 300 students. The school plans to grow by more than 50 percent over the next five years.

GSM does not offer an MIS concentration. The MIS curriculum supports all students with a required core course and optionally offers the following advanced courses: MIS, Modelling and Simulation, Forecasting for Management, and Accounting Systems and Control. In addition, an undergraduate Systems Analysis and Design course is an eligible one for M.B.A. students to take. Currently, a student with strong IS interests can develop an "informal" MIS concentration, by selecting the general concentration and focusing it on MIS. The general concentration is an option in which a student, in consultation with his/her advisor, specially constructs his/her 2nd year courses in an agreed-upon area of interest. Such a plan enables an MIS-oriented student to write a thesis on an MIS topic. Over the past two years there have been about 10 students in such "informal" MIS concentrations.

Another formal curricular approach to MIS is for students to take field internships in computer-related job roles. These offer the student academic credit, as well as valuable training experience in the real world. Among firms sponsoring computer internships have been IBM, Anheuser-Busch, American Diversified Services, Sunkist Growers, Computerland, the Riverside Symphony, Riverside County, and UCR's top administration.

The school has long included computer tools in its curriculum, but moved heavily towards micros starting in summer, 1984. Prior to 1984 the school used a Prime mini and IBM mainframe to support a variety of courses, emphasizing statistical and business game packages. In summer 1984, the school received funding from UCR's top administration to institute numerous micros for teaching and research purposes. Presently there are 35 micros in the school, 14 for teaching, 14 for faculty offices, and 7 for administrative use. All except four are members of the IBM PC family (the others are two Macintoshes and two Apple IIe's). The student-to-micro ratio of 7.1 is among the top 15 in the U.S., when compared with results of the Frand/McLean study of U. S. business schools.(1) The micros are not networked, and eight are hard disk units. Concurrent with the advent of micros, the school also established a Computer Facility, consisting of a 2000 square foot specially-designed space, and a staff of a director and five student computer consultants.

II. INFORMATION SYSTEMS CURRICULUM REVISION

The IS curriculum was formulated in a

series of plans from fall 1984 to fall 1985, and is presently largely implemented, the remaining pieces to be implemented by fall, 1986. Its major parts consist of (1) revision of the core IS course, (2) some enhancement and change to the advanced IS curriculum, (3) expanded computer internships, and (4) required micro labs.

REVISION OF THE CORE IS COURSE

Although GSM is not yet accredited by the AACSB, it has prepared for accreditation by offering a 11-course core curriculum, including an IS course, Computer Systems for Management. This course, prior to revision, emphasized the following major topics: introduction to MIS, computer functioning, BASIC, the SPSS-X standard statistical package, and business information systems. Weekly required, hands-on-labs served to develop the following skills: interactive use of the Prime mini, the Prime editor, BASIC, and SPSS-X. The course was well-accepted by students, and served effectively to insure a modicum of skills for non-IS students and to serve as a springboard into advanced IS courses for the IS-oriented.

Under the guidance of a faculty committee, this course has been revised as follows:
1. It has dropped all statistical content and broadened its coverage of business IS. The dropped statistical material and labs will be added to a revised core course in business statistics. Topics now covered are the following: introduction, functioning of computers, MIS, programming and languages, BASIC (3 weeks), systems analysis and design, data-bases and distributed systems, office automation, and telecommunications/the future. The revised course conforms essentially to AACSB standards, as well as to the content of most textbooks for the first course in computers for management. If the topics seem too few, it should be noted that UCR is on the quarter system. Another change in the lecture portion is inclusion of class discussion of four case studies, spaced at two week intervals.

The lab part has been revised to train students in the following skills: introduction to the GSM Computer Facility, introduction to the PC and DOS, presentation graphics, BASIC (IBM PC), dBASEIII, interactive use of the VAX mini, VAX editor, micro-mini uploading and downloading, and use of the GSM Value mini data-base. The revised lab will continue under the format of hands-on graded weekly exercises. The major changes are more emphasis on micros, data-basis, and micro-mini interaction.

The student is required to purchase a textbook in business IS (which includes BASIC), a micromanual (which includes material on dBASEIII), locally-printed lab exercises, and the GSM Computer Facility Users Guide. Most of this material, especially the latter three items, can be retained for use throughout the student's M.B.A. career.

ADVANCED MIS COURSES

Two MIS courses, Data-Bases for Management and Accounting Systems and Control, have been added, and several others are being revised, in order to incorporate recent material as well as to create synchrony in the IS curriculum, avoiding overlap of material. For example, the topic of functional systems (accounting, marketing, etc.) is being dropped from the core IS course in order to avoid overlap with the advanced course on Accounting Systems and Control. The overall result is an enhancement in the IS curriculum, laying the groundwork for a planned MIS concentration in a year or two.

EXPANDED COMPUTER INTERNSHIPS

The computer internship program will be expanded. Internships will be more numerous and better planned beforehand with managers in companies, governmental, or non-profit organizations. Also all internships will be required to be on a paid basis. This tends to improve a student's attitude towards the internship and to raise the importance in the eyes of the outside sponsoring organization.

REQUIRED MICRO LABS

The school wrestled with how to institute required micro training in the curriculum. For a quarter-system school, the core course would be overloaded, if all micro-training were placed there. It was also felt that adding a separate "micro course" to the core curriculum would burden students too much. Other business schools such as Harvard have implemented a 3-4 week non-credit short course on micros preceding the student's first term, but September in Riverside is very smoggy. After deliberating, the committee decided to attach seven required micro labs to a variety of non-IS courses in the core curriculum. For instance, a 1 1/2 hour lab on spreadsheets has been attached to the Financial Analysis core course. The lab teaches the Lotus 1-2-3 spreadsheet, emphasizing financial subject matter directly related to the course. The full set of labs and associated micro software is shown below.

Micro Lab	Software
(1) Introd. to IBM PC	DOS
(2) Spreadsheet	Lotus 1-2-3

(3) Data-Base	dBase III
(4) Word Proc I	WordStar 2000
(5) Word Proc. II	WordStar 2000
(6) Simulation	IIE,IMSL
(7) Organiz. Tools	Mgt. Edge

These labs are associated with the following core courses: (1) Workshops in Administration, (2) and (3) Financial Analysis, (4) and (5) Marketing, (6) Management Science, and (7) Organizational Tools.

The Management Edge package is an expert systems package which asks students extensive questions about their managerial style and experience and then produces a written report criticizing the student as a manager. Although not ideal for computer training, this lab has served as a spur for spirited class discussions and reports, and instructors consider the lab pedagogically sound. Starting in Fall 1986, this lab will be based instead on Harvard Business School case study software.

In summary, the IS curriculum revision has been extensive, and required the cooperation of nearly the entire GSM faculty, as well as student body and computer facility staff, which supervises the micro lab sessions.

III. MICRO TRAINING SEMINARS

Another avenue for a business school to institute computer skills training is through informal seminars. In GSM, extensive informal seminars have been instituted by the GSM Computer Facility. Each seminar is 1 1/2 hours in length and taught by the facility staff. In the 1985-86 academic year seminars were taught on IBM PC/DOS, WordStar 2000, Lotus 1-2-3, dBASE III, VAX/SAS, CRSP data-base, ValueLine data-base, Advanced WordStar 2000, Advanced Lotus 1-2-3, Keychart presentation graphics, Uploading/Downloading/File Transfer, Advanced Primos, and Word Processing for Faculty. All seminars except the latter were open to GSM students, faculty, and staff, as well as to persons invited from other departments. The last lab was restricted to and tailored to GSM faculty to encourage greater faculty use of micros.

IV. FUTURE PLANS AND CONCLUSION

In the future GSM's strategic plan calls for instituting a MIS concentration. Starting from the base of the present curriculum, outlined in Section II, the planned concentration will require the introduction of only three new required courses, File Processing with COBOL, Decision Support Systems, and MIS/Industry Seminar. Thus the present curriculum both builds on the past and is keyed to the school's future.

The following conclusions may be made drawn from GSM's experiences with designing and implementing an IS curriculum in a small business school. First, requiring student knowledge of both micros and minis maximizes use of computers in a 2-year M.B.A. program and supports broad computer skills preparation for today's business world. Second, micro skills are taught most effectively using a multifaceted approach, through formal required courses, informal seminars, and consulting contacts. For such skills, some learning repetition greatly improves overall results. Third, there is great advantage in requiring a standardized set of computer skills. Each GSM student is required to have working knowledge, for minis, of VAX/VMS, VAX/EDT editor, BASIC, and SPSSX, and, for micros, of DOS, WordStar 2000, Lotus 1-2-3, dBASEIII, IIE/IMSL, Management Edge, Keychart, and Persoft TE-125 (for terminal emulation and downloading/uploading). Since this set of skills is understood as a school requirement by students and faculty alike, misunderstandings about skills are avoided, and faculty can plan course content and lab assignments accordingly. Fourth, moderate knowledge of a programming language, BASIC, is required. Fifth, computer aspects of the curriculum, both IS and non-IS, are carefully integrated with the operation of the school's computer facility. For example, student computer consultants working in the facility supervise the required micro labs; the facility director teaches the core IS course. Finally, the emerging IS curriculum is based on a hands-on approach. Although limited in scope, it is keyed to the present demands and resources of a small business school.

REFERENCE

(1) Frand, Jason and Ephraim McLean. 1985. Second Annual UCLA Survey of Business School Computer Usage. Los Angeles. UCLA Graduate School of Management.

The Use of Externally Directed Projects in Software Engineering Courses

Scott N. Woodfield
Douglas M. Campbell

Brigham Young University
Provo, Utah 84602

ABSTRACT

The number of software engineering classes has greatly increased over the last five years. Experience gained from teaching these classes supports the use of some type of group project. Two general types of projects which have been incorporated are internally and externally directed projects. Three software engineering courses at BYU recently switched from internally directed projects to externally directed projects. This paper describes why externally directed projects have provided a much better experience for our students than internally directed projects.

INTRODUCTION

The number of software engineering classes has significantly increased during the last five years. Such classes can be one semester undergraduate type introductions [Kant81, Carv84], multi-semester in-depth graduate instruction [Coll82], or fully developed master's degrees [IEEE80, Stuc78]. A survey conducted by Petricig and Freeman [Petr84] confirms that many institutions use a software development project to reinforce software engineering principles. At some schools the project is part of the software engineering class; while at others the project is administered as a separate three credit lab.

We have taught software engineering classes at Arizona State University and Brigham Young University for the last five years using both externally and internally directed projects. In an internally directed project the instructor acts as the customer [Sath84, Henr83]. In an externally directed project the customer is someone outside the classroom [Carv85, Perk80]. While the internally directed project helps teach basic software engineering concepts, we have found the externally directed project superior in several aspects. This paper explains why.

The remaining sections of this paper present evidence to support our claims. In section 2 we describe the software engineering instructional environment at BYU. In section 3 we present the advantages of an externally directed project. In section 4 we describe some problems and solutions associated with externally directed projects. In section 5 we summarize experiences gained while using externally directed projects.

SOFTWARE ENGINEERING AT BYU

Three software engineering classes at BYU use team projects: a standard system analysis course, a general software engineering course emphasizing software design, coding, and testing, and a software testing course. Each class provides graduating students with experiences to make them competitive in the market place. Our graduate software engineering research courses do not employ projects.

The structure of the classes and the topics taught stress principles rather than mere techniques and memorization of methods. A student can't properly use a technique until he or she understands the underlying principles. We find that by incorporating a project in the class (as opposed to creating a projects class or a separate lab) students more consistently apply software engineering principles and techniques. Students doing projects as part of an independent lab too often revert to ad hoc development techniques used before entering the software engineering classes.

In the past we created the student project teams; currently we allow students to form their own teams and to choose their own leaders. Student created teams have performed no better or worse than teams formed by us.

Sixty percent of a person's grade is based upon the team project; this does not mean that all team members receive the same grade. We require all team members to fill out a confidential evaluation of their teammates which we use to adjust individual team member's grades.

The externally directed projects are large and complex; consequently most students are unable to experience every aspect of software development on the team project. Unfortunately, this allows some students to focus on only one aspect of software development and ignore other aspects, contrary to our desire to encourage generalists rather than specialists.

To make sure students experience the application of all principles and techniques taught in a software engineering class we assign each student the same internally directed individual project. The internally directed project is much smaller than the externally directed team project. The student must apply all the principles and techniques on the individual project that the different team members performed on the team project.

FIVE ADVANTAGES OF EXTERNALLY DIRECTED PROJECTS

Both internally and externally directed projects force the student to deal with such unfamiliar problems as requirements, specification, and quality assurance. Nevertheless, there are five specific advantages of externally directed projects over internally directed projects.

Realistic Customers

First, an externally directed project offers a degree of realism surpassing an internally controlled project. Indeed, an internally directed project is not as faithful to reality as a teacher might believe. In a real world situation the customer often does not know what the problem really is, let alone know how to solve it. In an internally directed project the instructor as customer has a fairly clear idea as to what the problem is and what should be done to solve it. Consequently, a student interacting with the teacher doesn't learn how to develop requirements and specifications from an unknowledgeable and confused customer.

No Premeditated Solution

Second, not only does the teacher as customer have a fairly clear idea of the problem, but the teacher has probably already designed a partial solution. When interacting with the student the instructor will unconsciously give hints and guidelines. A student soon learns that it is easier to interact with the teacher and determine what he or she is thinking than develop an independent design.

Realistic Testing Environment

Third, internally directed projects may not teach testing well. In an internally directed project, time and grading constraints lead a student to believe that testing consists of surviving a few test cases provided by the instructor to see if the software runs. The student knows the software will probably not be used after the semester is over. On the other hand, students on externally directed projects know that a real customer will use their project after the semester is over; the student therefore tests more extensively. If the student plans to use their customer as a job reference, then the testing is even more thorough.

Non-Academic Problems

Fourth, externally directed projects are 'real'; they represent a wide variety of business and application type problems similar to those encountered after graduation. No matter what an instructor tries to do, internally directed projects are not 'real'; reliance on computer systems programming, software engineering problems, and games fail to provide a student with the software development experience he or she hopes to be involved with after graduation.

Development of Communication Skills

Fifth, the communication and management skills developed by externally directed projects are superior to those developed by internally directed projects. Although it is true that both types of projects require a student to interact with the customer several times a week, the environment of the externally directed project is quite different from the class room. Students realize they must be well prepared, well groomed, and able to communicate well. In an externally directed project students learn to start a project without a teacher's prodding and learn to define a problem which must be solved by the end of the semester to a real customer's satisfaction. Recruiters are impressed by a student's involvement in an externally directed project. We have had students offered a job on the spot after presenting and explaining the documents that their project produced. Students from externally directed projects are more experienced in communicating with people and managing a project, a decided advantage in the business world.

SIX PROBLEMS AND SOLUTIONS

In this section we discuss solutions to six problems associated with externally directed projects. We do not claim the list is definitive nor our solutions unique.

Instructor's Feeling A Loss of Control

It is certainly true that internally directed projects give the instructor more direct control than externally directed projects. In the internally directed project the teacher as customer is directly involved in the actions of the various team members. He knows which teams are interacting, which are asking the right questions, and which are making progress. This interaction makes it easier to know when to teach and demonstrate various software engineering concepts in class.

When the instructor is not the customer there is often a feeling of loss of control. One way to regain control is to become the quality assurance manager. Wearing the quality assurance hat allows the teacher to: insure that things are done correctly, require the correct type of documents, and establish proper standards and quality assurance methods. A teacher can require a weekly activity chart which shows who is responsible for what activity, when the activity is to be finished, and the activity's current status. This allows the instructor to retain control while teaching the student to be responsible for the quality of their own work. The teacher can devote class time to teaching concepts and principles which can be used on any project rather than talking about the details of a specific project.

Variant Job Sizes

With internally directed projects the instructor can give the same size problem to all teams. But externally directed projects usually are not the same size; different teams therefore do different amounts of work.

Our solution has been to put the responsibility of sizing the project upon the shoulders of the students. At the beginning of the semester we tell all students that they are responsible for finishing their team's project. The first week we offer the names of several external customers from which students may choose; more often than not they find their own customer and choose their own project. If they choose a project which is too small, then we reject it the first week of class. If the project is too large, then we let them suffer the consequences. Putting the responsibility for project size on the students reduces variation in project size.

Growth of Project Size

When an internally directed project gets too large the instructor can reduce the scope of the project. Customers of externally directed projects tend to want more than can be accomplished in a semester. Once an agreement has been reached between the customer and the team, the customer may be unwilling to reduce the scope of the project even if it puts undue pressure on a team.

We have minimized this problem by interviewing potential customers. We explain that if their problem grows too big to be solved in a semester, then we expect them to reduce the size of the project. We monitor the team's progress and encourage teams to tell us when they feel a project has become too big. If we agree that the project is too big, then we negotiate with the customer. To date customers have been cooperative. In most cases the unfinished part of the project becomes a candidate project for the next semester.

Customer Will Not Perform

Occasionally an external customer may decide that interacting with "a bunch of college kids" is just not worth the trouble and withdraw participation leaving the students without a project. Our solution depends on when the disaster occurs. If it is early in the semester, then we split up the team and assign its members to other teams. If it is late in the semester, then the team is asked to continue as if the customer were still interacting. If it is in the middle of the semester, then we find little difficulty in acting as an unknowledgeable and confused customer.

Evaluation and Feedback

Fair and accurate evaluation of student performance makes better software engineers. Since an instructor is not directly involved with externally directed projects it is easy to be ignorant of a team's progress and quality of work. Unless an instructor provides feedback, students quickly regress to producing software by ad hoc software development techniques instead of going through the effort of doing it correctly. This is especially true near the end of the semester.

The instructor must monitor, evaluate, and provide feedback for externally directed projects since the real customer lacks proper training for these functions. The instructor's feedback reinforces the student's proper software development methods. To provide feedback we require teams to produce documents associated with milestones. These documents must meet stated quality standards and must use appropriate quality assurance methodologies [Coll85]. Requiring milestone documents significantly increases the quality and the number of completed projects. The review of these documents allows us to easily evaluate each teams progress. The entire team grade is determined by the quality of the documents and whether or not they were completed on time. Documents are evaluated twice. If the team does not like the grade they receive the first time, then they are given a chance to revise the document and resubmit it for evaluation at the end of the semester.

At one time, believing that teams would learn more by re-submitting documentation until they got it right, we allowed students to resubmit a document as often as the team desired. Unfortunately this policy promoted the badgering of teaching assistants with constant re-submissions, essentially forcing the teaching assistants to write the document. Teams that seriously try need only one re-submission.

Getting Customers

Gathering external projects is difficult and time consuming .The instructor must invest time each semester to find organizations which require software engineering services. Managers must be convinced that even though students may not finish a project, the results are worth the effort and risk. Switching to an externally directed project system requires an initial investment. The teacher must contact enough customers to provide sufficient projects for the students. After the first semester of externally directed projects, many of the customers request another group of students to work for them. Repeat customers appreciate the high quality professional work that our students have done for them, especially since the service is free. Once the credibility of the students' work has been established it is easier to recruit new customers. We give a hesitant prospective customer a list of previous customers. After contacting these references, the prospective customer quickly calls us. The instructor need not do most of the recruitment. Our students know that the class requires an externally directed project; many have already formed teams and selected a customer before enrolling. About 80% of the teams find their own project.

EXPERIENCES

Externally directed projects produce insights into software engineering education and real world software development. When compared to custom software produced by their employees or by consultant groups many customers felt that student software is completed faster and is of higher quality. A properly educated and guided software engineer can produce software of higher quality than the average developer without significantly increasing the cost.

Methods used for teaching software engineering students vary from teacher to teacher and institution to institution. Some teaching techniques seem superior to other techniques we have used. At one time we concentrated on teaching a set of tools which could be used for various aspects of software creation. While intelligent students used the tools well, far too many students were confused and perplexed. We have switched our emphasis from tools to principles of good software engineering. Tool use is now taught in the context of sound principles. In some instances we have taught principles in class and required the students to learn about the tools outside of class. When we de-emphasize tools students seem more confidant about what they are doing and why . They are better able to handle unusual situations. In the past, when a team encountered a situation which was different from a scenario presented in class they would descend upon the instructor to find out how to solve the problem; their fears prevented them from creating solutions that went beyond their tools and techniques. Because our current students are taught to understand why they are performing various tasks, we give them much more freedom. If they believe that a certain design technique or quality assurance document is not suitable for their application, they are allowed to make the necessary modifications as long as they can justify the changes. The increased student responsibility and authority has increased the project's quality.

Two support mechanisms are critical for a class which uses externally directed projects: a qualified teaching assistant, and a set of well defined and easy to understand documents.

A teaching assistant allows an instructor to fulfill his other responsibilities while handling thirty to sixty students. The teaching assistant has usually finished the class the previous year and understands the problems facing the students. Our teaching assistant monitors the progress of the teams, evaluates the quality of the documents, and answers technical questions. The instructor handles students appealing the decision of the assistant and questions concerning principles or method. An assistant is necessary if the class is to be monitored properly.

The set of necessary documents is class dependent. In our systems analysis class we require: a request for proposal, a proposal, a description of the current system, a constraints document, a description of the proposed system, and a feasibility study. An individual project requires additional documents. It is not the specific set of documents which is important; it is giving the students something to organize their actions and provide suggested milestones. It helps them feel as if they are in control.

A restructuring of our grading policy was student initiated. In the past, project documents were graded at the end of the semester. Upon receiving their grades some would exclaim that if they had only known how poorly they had been doing they would have been able to do better. They felt our one-shot grading system degenerated to grading the papers without communicating what was wrong. Although they realized that real life software projects only get one chance, they felt that the goal of software education would be better served by being able to resubmit documents for grading. We therefore allow a team to submit a document twice. This has reduced student complaints and produced higher quality documents. We feel that the next time the students are given a project they will be able to produce high quality documents the first time. The lower quality documents were not produced because the teams were incapable; they were produced because the team lacked the experience of knowing what was wanted, why it was wanted, and how to produce it.

CONCLUSION

Software engineering teachers face the question of whether a software development project should be associated with a software engineering class, and if so, what type of project. It is always difficult for a new discipline to judge which teaching approach will provide the best education. As we have struggled with these questions we have constantly modified our classes. We have been most successful when we have used externally directed projects. The problems which must be addressed for externally directed projects can in most instances be overcome; the benefits certainly seem to outweigh the problems.

REFERENCES

[Carv84] Doris L. Carver, "Software Engineering for Undergraduates," **SIGCSE Bulletin,** vol. 16, no. 3, pp. 23-25, SIGCSE, New York, New York, September 1984.

[Carv85] Doris L. Carver, "Comparison of Techniques in Project Based Courses," **SIGCSE Bulletin,** vol. 17, no. 1, pp. 9-12, SIGCSE, New York, New York, February 1985.

[Coll82] James S. Collofello and Scott N. Woodfield, "A Project Unified Software Engineering Course Sequence," **SIGCSE Bulletin,** vol. 14, no. 1, pp. 13-19, SIGCSE, New York, New York, February 1982.

[Coll85] James S. Collofello, "Monitoring and Evaluating Individual Team Members in a Software Engineering Course," **SIGCSE Bulletin,** vol. 17, no. 1, pp. 6-8, SIGCSE, New York, New York, March 1985.

[Henr83] Sallie Henry, "A Project Oriented Course on Software Engineering," **SIGCSE Bulletin,** vol. 15, no. 1, pp. 57-61, SIGCSE, New York, New York, February 1983.

[IEEE80] IEEE Computer Society Education Committee, **Draft Report on MSE-80: A Graduate Program in Software Engineering,** IEEE Computer Society, 1980.

[Kant81] Elaine Kant, "A Semester Course in Software Engineering," **Software Engineering Notes,** vol. 6, no. 4, pp. 52-76, ACM SIGSOFT, New York, New York, August 1981.

[Perk80] Thomas E. Perkins and Leland L. Beck, "A Project-Oriented Undergraduate Course Sequence in Software Engineering," **SIGCSE Bulletin,** vol. 12, no. 1, pp. 32-39, SIGCSE , New York, New York, February 1980.

[Petr84] Michael Petricig and Peter Freeman, "Software Engineering Education: A Survey," **SIGCSE Bulletin,** vol. 16, no. 4, pp. 18-22, SIGCSE, New York, New York, December 1984.

[Sath84] Harbans L. Sathi, "Project-Oriented Course For Software Systems Development," **SIGCSE Bulletin,** vol. 16, no. 3, pp. 2-4, SIGCSE, New York, New York, September 1984.

[Stuc78] L. G. Stucki and L. J. Peters, "Software Engineering Graduate Curriculum," **Proceedings of the 1978 ACM National Conference,** pp. 63-67, ACM, 1978.

"DATA FILES AND DATABASES" AND THE UNDERGRADUATE COMPUTER INFORMATION SYSTEMS CURRICULUM

Kirk L. Malmrose
Robert P. Burton
Department of Computer Science
Brigham Young University
Provo, UT 84602

Abstract

In October of 1985, the Data Processing Management Association (DPMA) Education Foundation published the model curriculum for undergraduate Computer Information Systems: "Curriculum '86." "Curriculum '86" recommends a course entitled "Data Files and Databases," CIS/86-6, as an undergraduate core requirement. Findings show that with the advent of "Curriculum '86" more than 80 percent of universities offering undergraduate degrees in Computer Information Systems do not require a course like CIS/86-6. The controversial role of CIS/86-6, "Data Files and Databases," in the undergraduate Computer Information Systems curriculum is discussed.

I. QUESTIONS AND ISSUES

To understand CIS/86-6 and its role as part of the undergraduate curriculum, the authors have sought answers to three questions:

(1) What factors influenced the DPMA Education Foundation to include "Data Files and Databases" as a core course requirement in "Curriculum '86?"

(2) With the publication of "Curriculum '86," what is the current incorporation of CIS/86-6 and its concepts? For what reasons do universities include or exclude CIS/86-6 and its concepts?

(3) What is the future of CIS/86-6 as an undergraduate course? Will "Data Files and Databases" lose its relevance to the CIS curriculum or will an understanding of these concepts be necessary?

These questions are discussed in sections III, IV and V, respectively.

II. CIS/86-6 OBJECTIVES AND CONTENT

The objectives and content of CIS/86-6 are extensive. The appendix provides a complete description.

III. FACTORS INFLUENCING THE DPMA EDUCATION FOUNDATION TO INCLUDE CIS/86-6 AS AN UNDERGRADUATE CORE REQUIREMENT

The DPMA Education Foundation Curriculum Committee included "Data Files and Databases" as a core requirement in "Curriculum '86" in part for the following reasons:

(1) A technical as well as a business base is important for graduates.

(2) Basic database concepts are essential to CIS graduates joining the work force.

(3) Indicators show that data files and databases will be used extensively in the future, particularly in microcomputer office implementations. (2,3)

IV. CIS/86-6 IN THE UNDERGRADUATE COMPUTER INFORMATION SYSTEMS CURRICULUM

Current Incorporation of CIS/86-6

To determine the extent of CIS/86-6's incorporation in undergraduate curricula, 108 (27%) out of 391 four-year institutions were selected randomly from the "Directory of DPMA Undergraduate Curriculum for Computer Information Systems Adopting Colleges and Universities." The Directory indicates the extent to which the model curriculum has been implemented. 65 percent (70) of the sampling carry a "Level 2" rating (the highest level of implementation). (4) Specific department titles vary from "Business Computer Systems" to "Information Systems and Quantitative Analysis." Each university's program, as presented in its general catalog, has been examined to determine if the university offers or requires a course similar in content to CIS/86-6. (5)

Of the 108 schools, only 16 percent (17) of the selected schools <u>require</u> a course similar to CIS/86-6. 25 percent (27) <u>offer</u> a course resembling the DPMA's "Data Files and Databases." These course titles range from "Computer File Structures" to "Introduction to Databases." In most instances, these courses appear to ignore or treat lightly most CIS/86-6 concepts while placing considerable focus on either data files or database concepts. 36 percent (39) implement the equivalent of CIS/86-6 as a multiple semester course sequence. The first semester usually emphasizes file structures; subsequent semesters target database concepts. The remaining 23 percent (25) offer only part or none of CIS/86-6. Typically, data file concepts are bypassed (see Figure 1).

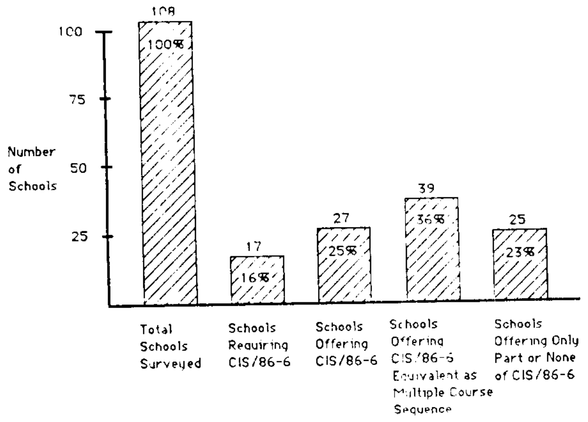


Figure 1. Incorporation of CIS/86-6

General Findings

Computer Information Systems curricula in applications or technically oriented colleges typically offer courses closely related to CIS/86-6. File structures, database concepts and programming methods are stressed heavily.

By contrast, Computer Information Systems curricula in management or business colleges typically do not contain a course similar to CIS/86-6. Stressing business aspects, such colleges teach database concepts without reference to the internal structure of data or programming methods used to implement databases.

Perceptions of CIS/86-6 as an Undergraduate Requirement

Universities offer a data files and database course in part for the following reasons:

(1) A course covering data files and databases adds credibility to a university's CIS curriculum.

(2) Industry is moving toward very large database systems. Therefore, theoretical and practical knowledge of database concepts is essential to the CIS student. (2,3,6,7)

Universities exclude CIS/86-6 in part for the following reasons:

(1) CIS/86-6 is difficult and expensive to implement because it is very dependent on existing facilities.

(2) Business faculty are not interested in teaching an applications programming course.

(3) Faculty are not sufficiently familiar with course material.

(4) Technical areas of CIS/86-6 are not considered necessary for business oriented curricula.

(5) CIS/86-6 contains too much material to be covered adequately in a single semester. A two or three semester sequence is more appropriate, but increases credit hour requirements. (2,7,8,9)

"Business Knowledge" vs. "Technical Knowledge"

Attitudes toward "Data Files and Databases" vary widely. CIS/86-6, covering the foundations of database concepts, is thought by most business faculty to be extraneous to a model Computer Information Systems curriculum. Business concepts and computer literacy are considered paramount; technical database concepts can be relegated to computer science and computer engineering. The debate continues as CIS technical faculty maintain that data files and databases are central to Computer Information Systems.

The authors' paper "File Processing and the Undergraduate Computer Science Curriculum" discusses a similar attitude among computer science and computer engineering faculty. (10) These faculty see the Association for Computing Machinery's recommended "Introduction to File Processing" primarily as a computer information systems, data processing course, and not as an appropriate course in a computer science or computer engineering curriculum. Ironically, each discipline views data files and file processing as subjects to be taught by the other.

V. THE FUTURE OF CIS/86-6 AS AN UNDERGRADUATE COURSE

The future of CIS/86-6 is clouded. Faculty that view internal structures and technical areas of CIS/86-6 as superfluous to the CIS curriculum do not seem likely to abandon their business oriented stance. On the other hand, faculty who consider all concepts of CIS/86-6 to be necessary in the CIS curriculum maintain that data files and database concepts cannot be taught satisfactorily in a single semester. Considering these strongly held opposing views, further implementation of CIS/86-6 in its present form does not appear likely.

VI. CONCLUSIONS AND RECOMMENDATIONS

With the advent of "Curriculum '86," only 25 percent of the nation's universities and colleges <u>offer</u> a course similar to the recommended CIS/86-6; only 17 percent <u>require</u> such a course. It is doubtful that all CIS/86-6 concepts can be covered adequately in a single semester. Valid reasons for inclusion and exclusion are noted in section IV of this paper. The authors propose two alternatives to accommodate the relevant concepts from CIS/86-6 as described in the DPMA model curriculum:

A. If studies show that a nontechnical familiarity of data files and databases is sufficient for the CIS student, the course content of CIS/86-6 ought to be modified accordingly. Technical concepts from CIS/86-6 might be offered in an elective course.

B. If studies show that a technical knowledge of data files and databases is necessary for the CIS student, the feasibility of CIS/86-6 as an individual course needs to be questioned. A multiple semester sequence covering the concepts of CIS/86-6 may be warranted.

Until such an action is taken, CIS/86-6 will remain a stepchild in the DPMA model curriculum.

APPENDIX

<u>**CIS/86-6 Data Files and Databases**</u>

<u>Course Objectives</u>

Upon completion of the prescribed work for this course, the student should be able to:

* Define the components and terms used in database management systems (DBMS).
* Describe the main features of the types of data models used in building databases.
* Compare and contrast the advantages and disadvantages of linked-list, hierarchical, network, inverted, and relational models.
* Evaluate backup and recovery techniques.
* Discuss the importance and responsibilities of the database administrator.
* Discuss the relevance of data dictionary.
* Given a sample application and data, normalize the data, create a logical model, and draft appropriate schema and subschema.
* Discuss the critical elements for implementing and maintaining a DBMS.
* Evaluate the implications for integrity and redundancy of content of distributed databases and databases implemented on microcomputers.

<u>Course Content</u>

1. <u>Overview</u>
   A. Data and reality
   B. Concept of an information structure:
      a. Entities
      b. Attributes
      c. Values
   C. Views of data:
      a. End-user's view
      b. Application programmer's view
      c. Database technician's view
      d. Database administrator's view
   D. Database development and the system life cycle
   E. Database issues:
      a. Security
      b. Integrity
      c. Privacy
      d. Data dependence
   F. Files vs. databases: advantages and disadvantages
2. <u>Direct Access Storage Device Characteristics</u>
   A. Sequential files and media:
      a. Blocking
      b. Spanned records
      c. Variable length records
   B. Random access files and media:
      a. Sectors, tracks and cylinders
      b. Volumes
      c. Blocking
      d. Spanned records
      e. Spanned files
      f. Variable length records
3. <u>Modeling the Domain of Information</u>
   A. Data element analysis
   B. User views
   C. Policy
   D. Storage and access design
4. <u>Indexed Files</u>
   A. Index levels:
      a. Full
      b. Block
      c. Hierarchical
      d. Multiple or alternate keys
   B. Access types:
      a. Sequential
      b. Random
      c. Dynamic
   C. Overflow handling
5. <u>Direct File Organization</u>
   A. Bucketing
   B. Relative addressing
   C. Hash addressing
   D. Packing density
   E. Overflow handling
6. <u>Applied Data Structures</u>
   A. Linked structures:
      a. Linked lists
      b. Pointer chains
      c. Rings
      d. Unidirectional linking strategies
      e. Bidirectional linking strategies
      f. Multiple linking strategies
   B. Chain processing:
      a. Insertion
      b. Deletion
      c. Traversal
   C. Trees:
      a. Balanced
      b. Unbalanced
      c. Binary traversal
      d. Preorder traversal
   D. Networks
   E. Data structure diagrams:
      a. 1:1 relationships
      b. 1:N relationships
      c. M:N relationships
      d. Cycles
      e. Loops
7. <u>Data Model Overview</u>
   A. Definition of data model
   B. Data definition languages (DDL)
   C. Data manipulation languages (DML)
   D. Representation of entities, attributes, and relationships
   E. Prominent data models:
      a. Hierarchies
      b. Networks
      c. Relationships
   F. Physical vs. logical database records
   G. Evaluation framework for data models
8. <u>Hierarchical Data Model</u>
   A. DDL Terminology and concepts:

a. Field
b. Segment
c. Database record
d. Database

B. DML Terminology and concepts:
a. Get Unique
b. Get Next
c. Get Next with same Parent
d. Get Hold

C. Database processing activities:
a. Replace record
b. Delete record
c. Insert record

D. Description of an actual hierarchical DBMS

E. Advantages and disadvantages of the hierarchical model

9. <u>Network Data Model</u>

A. DDL terminology and concepts:
a. Data item
b. Vector
c. Repeating group
d. Record
e. Set (Member and Owner)
f. Realm and area

10. <u>Database Administration Overview</u>

A. Roles:
a. Database administrator
b. Database technician
c. Application programmer
d. Systems analyst designer

B. Administrative concerns:
a. Security
b. Privacy
c. Integrity
d. Backup and recovery
e. Concurrency control
f. Auditability
g. Distributed processing

11. <u>Data Analysis, Design, and Implementation Life-Cycle</u>

A. Requirements analysis, reviewed
B. Logical object analysis
C. Relational analysis
D. Aggregate design
E. Storage and access design
F. External schema
G. Schema:
a. Conceptual
b. Logical
c. Physical
d. External

12. <u>Database Technology</u>

A. Hardware:
a. Storage devices
b. Back-end machines
c. Database machines

B. Software:
a. Operating system interface
b. Physical vs. logical I/O
c. On-line system interface
d. DBMS ulilities

C. Performance monitoring

13. <u>Selection and Acquisition Criteria for DBMS</u>

A. Correspondence between DBMS model and data model of organization
B. Matching a DBMS to the application mix that will access data
C. Compatibility of DBMS query language and structures with programs and existing data files
D. Future application development plans and appropriateness of DBMS and data model to support planned work
E. Readiness of the organization to adopt fourth or fifth-generation techniques for building applications and handling data resources

NOTES AND REFERENCES

1. The DPMA Model Curriculum for Undergraduate Computer Information Systems, "Curriculum '86", Data Processing Management Association, First Edition, October 1985, p.20-23.

2. Telephone interview of Robert M. Babb, DPMA/EF Curriculum Committee Chairman, Computer Information Systems Department Marshall University, WV, by Kirk L. Malmrose (March 10, 1986).

3. Telephone interview of David R. Adams, DPMA/EF Technical Advisor for "Curriculum '86," Department of Information Systems, Northern Kentucky University, KY, by Kirk L. Malmrose (March 10, 1986).

4. "1985 Directory DPMA Undergraduate Curriculum for Computer Information Systems Adopting Colleges and Universities," Data Processing Management Association Education Foundation, p. 1-38.

5. Each of the 108 universities' general catalogs of courses and curriculum specifications has been considered in gathering data for the report; this research has provided information to determine if a course in data files and databases is available or required in a university's Computer Information Systems program.

6. Telephone interview of Donald D. Medley, DPMA/EF Technical Advisor for "Curriculum '86", Computer Information Systems Department, California State Polytechnic University, CA, by Kirk L. Malmrose (March 12, 1986).

7. Telephone interview of Karen Gardner, DPMA/EF Content Contributor for "Curriculum '86", Information Systems, Golden Gate University, CA, by Kirk L. Malmrose (March 10, 1986).

8. Telephone interview of Gary Hansen, Department of Information Management, Brigham Young University, UT, by Kirk L. Malmrose (March 10, 1986).

9. Telephone interview of Helen Wolfe, DPMA/EF Content Contributor for "Curriculum '86", Business, Post College, CT, by Kirk L. Malmrose (March 10, 1986).

10. "File Processing and the Undergraduate Computer Science Curriculum", Kirk L. Malmrose and Robert P. Burton, Department of Computer Science, Brigham Young University, UT. In preparation.

"TEACHING THE BALANCE LINE ALGORITHM METHOD FOR SEQUENTIAL FILE UPDATE USING A MODEL"

RON TEEMLEY

DeVRY INSTITUTE OF TECHNOLOGY
4250 North Beltline Road
Irving, Texas 75038
(214) 258-6767

Abstract: This paper presents a method of teaching the Balance Line Algorithm design for sequential update of a master file with one or more transaction files. The Balance Line is one of the truly beautiful pieces of design work in structured programming. The complexity of this design makes it very hard for students to grasp at their first exposure. Here the Balance Line is broken into easy-to-learn pieces and the teacher reconstructs the design piece-by-piece with the students working from a MODEL program design.

INTRODUCTION

This paper describes a method of teaching the very elegant and powerful Balance Line Algorithm (hereafter referred to as the Balance Line) design used for sequential file update. The teaching method is based on the structured programming model (hereafter called the MODEL introduced in [5] . This material is presented to sophomore students in their second course in COBOL programming at DeVRY Institute of Technology, Irving (Dallas), Texas.

THE MODEL DESIGN

The idea of the structured programming MODEL shown in Figure 1 is to provide a framework for beginning programming students to use in designing their programs. A vast number of programs can be designed simply by adding modules to or deleting modules from the MODEL. Then, as students gain a little experience and confidence, they begin to work with more sophisticated designs that also require manipulation of portions of the MODEL. This method of designing structured programs was developed as an easy-to-use, enjoyable alternative for students to use in place of pseudocode and flowcharts and **it really works!**

There are two features of the MODEL that I make heavy use of in this paper:

(1) The MODEL and all designs derived from the MODEL can (and should) be used to follow the logical flow of the program reading the chart row-by-row, left-to-right, top-to-bottom. The symbol '↻' above a module indicates that module is to be processed until the condition next to the symbol is true and ' ◆ ' above a module indicates the module is to be executed only if the condition next to the symbol is true.

(2) Each module chart is a detailed module chart: each module performs only one task and each contains only a few lines of code. Hence the logic inside each module is trivial. Stated another way, all the interesting logic of the program is given on the module chart.

For more information on teaching COBOL from this MODEL see [5].

THE BALANCE LINE DESIGN

The Balance Line Design is one of the truly beautiful pieces of work in structured program design. Its purpose is to sequentially update a master file from one or more files of add, change, and delete transactions via a methodology that "lays bare" the logical ideas of such a matching process. The Balance Line Design is the work of Cooper, Cowan, Dirksen and Graham of Waterloo, Ontario [1]. This book is unfortunately no longer in print, but you may read about the method in [2] and [4].

The interesting thing in the logic of a sequential file update is matching the key fields on the old master and transaction records as you are reading from these two (or more) files and then processing the record (or records) containing the key that comes first in sequential order while deferring the processing of records that have been read but come later in the sequential ordering.

Cooper, Cowan, Dirksen and Graham handle this in a way that is at the same time remarkably simple yet marvilously elegant. They establish a hold area called the ACTIVE KEY: the keys of the records in the read-in areas are compared and the key(s) that comes first in sequential order is moved to the ACTIVE KEY.

Each time before entering the processing loop (C level activity on the module chart) the ACTIVE KEY is chosen so that the record(s) whose key is ACTIVE is the one(s) that is processed that time through the "loop".

A second device to help regulate the detail processing is a switch which we will call MASTER-BEING-PROCESSED-SWITCH which is set to 'YES' if we have an active master record, set to 'NO' otherwise. This switch is very important and very simple to use: at each task in our detail processing we check the setting of this switch to see whether or not it is appropriate to perform the task on this particular trip through the processing loop.

The teamwork of the ACTIVE KEY and the MASTER-BEING-PROCESSED-SWITCH is wonderful. Under the control of this pair the update moves through the modules of the processing loop like an efficient, well-oiled machine.

Why is this design called "the BALANCE LINE"? It originates from the physical appearance of the so-called structure diagram for this design as used by Cooper and associates in [1]. Another interpretation is in terms of the "balance" that the program is constantly trying to maintain between the master and transaction files so that they will stay in sequential order.

TEACHING THE BALANCE LINE USING THE MODEL

By the time DeVRY students are introduced to the Balance Line design they have had one and one-half semesters of practice designing COBOL programs using the MODEL (Figure 1). Now, once again, we begin by putting a copy of the MODEL in front of us as we proceed to construct the Balance Line design from the MODEL, step-by-step.

We break the Balance Line design into easy-to-handle pieces or cases and put it back together again piece by piece as we discuss it with the students.

CASE 1 TASK: Create the original Master File from a transaction file of ADD transactions only.

MODULE CHART: Figure 2

NARRATIVE: Starting with the MODEL we design the program for CASE 1 by simply deleting modules B20, B50, and C10 (and adjusting the terminology in C20 and C30).

CASE 2 TASK: Update the Master File with ADD transactions only - Errors are not considered.

MODULE CHART: Figure 3

NARRATIVE: The instructor must first make sure the students are fully aware of the logical challenge presented by this file matching procedure. Only then should the instructor reveal how the Balance Line design beautifully and successfully meets this challenge with the ACTIVE KEY which he then proceeds to explain. After digesting the ACTIVE KEY logic the students will be able to understand how the CASE 2 design is derived from CASE 1: first add another READ module for the old master file (B level and C level); the only other addition to the processing loop (C level logic) is a second level FORMAT module, C10, necessary now that our new master record may come from either an old master record or a transaction record, depending on which one is ACTIVE.

CASE 3 TASK: Update the Old Master File with ADD transactions only. Errors are considered.

MODULE CHART: Figure 4, revised in Figure 5

NARRATIVE: First ensure that the students understand the two types of errors possible here. (1) More than one ADD transaction with the same key, and (2) An ADD transaction for a record that already exists on the Master File. In our processing loop (C level logic) there are some modules we perform if the Old Master record is ACTIVE AND OTHERS WE PERFORM if the Transaction record is ACTIVE. With the possibility of errors another "level" of logic is required. Because, once we get to the appropriate set of modules to process an ACTIVE Master and/or to process an ACTIVE Transaction we must decide again between two alternate paths: one if our transaction record is VALID and another if it is INVALID.

The Balance Line design handles this complicated VALID/INVALID logic (the E level logic of Figure 4) very elegantly with the MASTER-BEING-PROCESSED-SWITCH introduced in the previous section.

Now let's see how we obtain the design for CASE 3 from CASE 2 very simply by using this switch. First we introduce a new module C10-SET-MASTER-BEING-PROCESSED-SWITCH at the beginning of the processing loop to initialize the switch: if OLD-MASTER-KEY = ACTIVE KEY set the switch to 'YES', otherwise set the switch to 'NO'. Comparing the C level logic of CASE 2 and CASE 3, the only other differences are that the FORMAT-NEW-MASTER-FROM-TRANSACTION module has been replaced by PROCESS-ACTIVE-TRANSACTION and the READ-TRANSACTION-RECORD module is no longer on the C level. We reason like this: with the introduction of error processing and its alternate VALID/INVALID, processing tasks putting all of the modules at the C level would make it too cluttered.

Hence all transaction processing tasks are grouped under the control module C30 and delegated to the D level where we see: if the switch is set to 'YES' we already have a master being processed and an error is displayed; if the switch is set to 'NO' we do not currently have an ACTIVE master record so we format a new one from the ADD transaction and set the switch to 'YES'. This will alert all modules due to be executed later in the loop that we now have an active master we are processing.

NOTE: Figure 5 shows a revised version of the CASE 3 program. Figure 4 leaves the designer a little uncomfortable since some of the master record tasks are on the left of the transaction routine and others are on the right.

In addition, the details of transaction processing are now on the D level while those of master processing remain on the C level. For the sake of beauty and balance (no pun intended) we have dropped all master detail processing to the D level. The master and transaction routines now appear balanced in Figure 5.

CASE 4: TASK: Update the Old Master File with ADD and change transactions.

MODULE CHART: Figure 6

NARRATIVE: From this point on the design proceeds quite smoothly. We now expand a design technique described in CASE 3 when too many detail tasks accumulate on one level of the module chart, arrange the tasks into subgroups that seem to naturally go together and assign a control module to each group. Then only the control modules remain on the original level while the detail groups of tasks are pushed down one level.

In our situation, trying to line up all the tasks to be performed for both VALID and INVALID ADD and CHANGE transactions would overcrowd the D level. Hence we place only the control modules on the D level: D20-PROCESS-ADD-TRANSACTION and D30-PROCESS-CHANGE-TRANSACTION while the detail tasks all appear on the E level (Figure 6). Making changes to an old master record is quite simple. If the MASTER-BEING-PROCESSED-SWITCH has a 'YES' setting we know there actually exists an active old master record that we can change. Each non-blank field of data on the change transaction is moved to the corresponding field on the master record causing the "change". If the switch is set to 'NO' we display the error message: 'NO MASTER EXISTS' with this transaction.

CASE 5: TASK: Update the Old Master File with ADD, CHANGE, and DELETE Transactions.

MODULE CHART: Figure 7

NARRATIVE: With this module chart we will have the complete Balance Line design. To go from the design of CASE 4 to that of CASE 5 two steps are required, one obvious, one not so obvious. The obvious step: Add a control module PROCESS-DELETE-TRANSACTION at D40 and line up the delete transaction detail tasks under D40 on the E level - If the MASTER-BEING-PROCESSED-SWITCH has a 'YES' setting then our delete transaction can be applied and the deletion is indicated by setting the switch to 'NO' (this is the Balance Line program's way of telling all modules that follow later in the loop that an active master no longer exists). On the other hand if we find the switch set to 'NO' as we start our deletion processing there is no master to delete and we display the transaction with the message 'NO MASTER EXISTS'.

The not-so-obvious step: C40-WRITE-NEW-MASTER-RECORD now becomes a conditional module! If we set the MASTER-BEING-PROCESSED-SWITCH to 'NO' in the deletion routine we don't want to write to the new master file. In this way the record is deleted. (Figure 7)

EPILOGUE

It is possible to rework the Balance Line design so it will perform random file updating as in [4] Chapter 7.

In my opinion this is not appropriate. Sequential update involves rather complicated logic that requires the sophisticated Balance Line design while the logic involved in a random update is extremely straight forward and does not require the power of the Balance Line design. Figure 7 does not disclose the full beauty and usefulness of the Balance Line design. There are several extensions and applications of considerable interest. These will be given in my next paper.

BIBLIOGRAPHY

1. Cooper, R.H., D.D. Cowan, P.H. Dirksen and J.W. Graham, File Processing with COBOL/WATBOL, WATFAC, Waterloo, Ontario, 1973.

2. Cooper, R.H. and L.F. Johnson, File Techniques for Data Base Organization in COBOL, Chapter 3 and 5, Prentice-Hall, Englewood Cliffs, N.J., 1981.

3. Dwyer, Barry, "One More Time - How to Update a Master File", Communications of the ACM (January 1981).

4. Grauer, Robert T., Structured Methods Through COBOL, Chapters 6 and 7, Prentice-Hall, Englewood Cliffs, N.J., 1983.

5. Teemley, Ron "Teaching Structure COBOL Programming Using a MODEL", Fourth Annual Proceedings of the Information Systems Education Conference (1985).

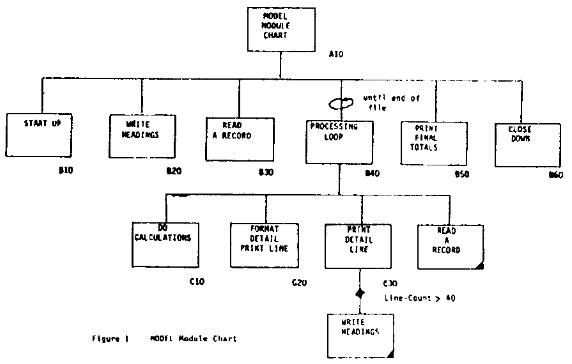


Figure 1 MODEL Module Chart

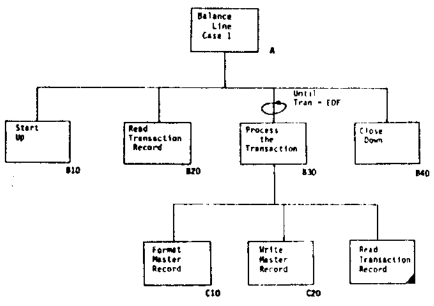


Figure 2 Case 1 Create the Original Master File from a File of Add Transactions

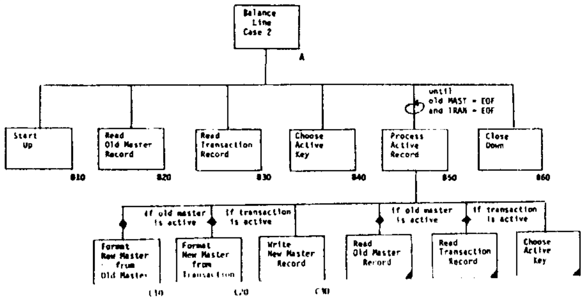


Figure 3 Case 2 Update the Master File with Add Transactions Only Do Not Consider Errors

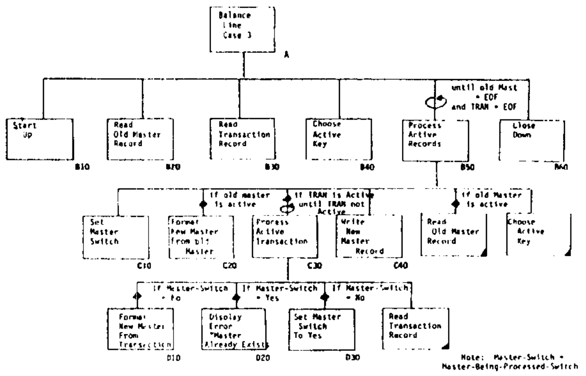


Figure 4 Case 3 Update the Master File with Add Transactions Only - Do consider errors

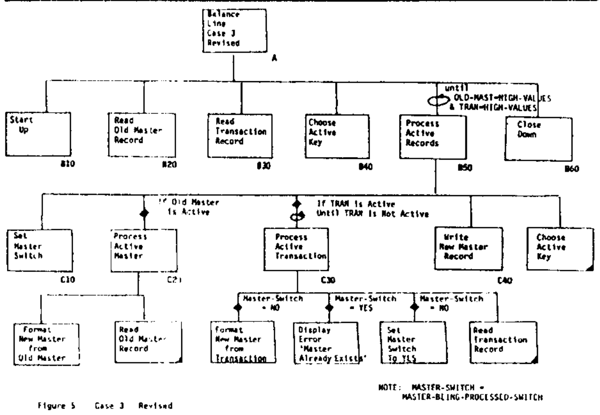


Figure 5 Case 3 Revised

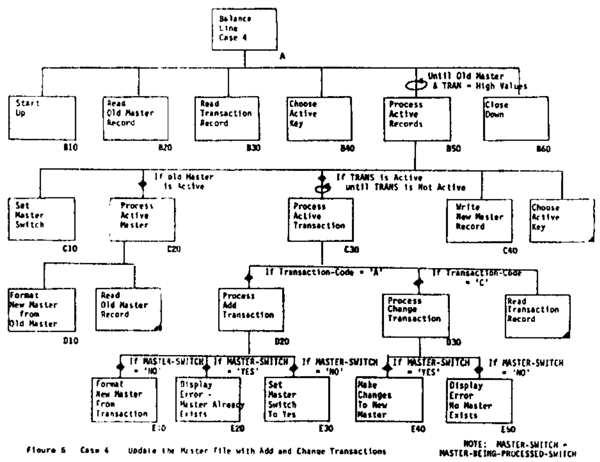


Figure 6 Case 4 Update the Master File with Add and Change Transactions

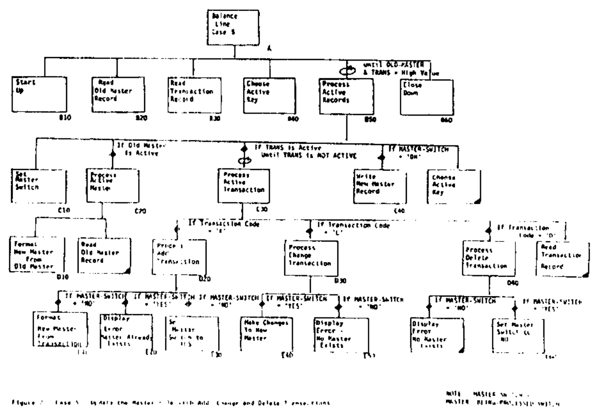


Figure 7 Case 5 Update the Master File with Add Change and Delete Transactions

BEHAVIORALLY BASED TEACHING METHODOLOGIES FOR CIS COURSES

Richard Leifer
School of Management
Rensselaer Polytechnic Institute
Troy, New York 12180
518 266-6831

Captain Michael J. DeHaemer, USN
Naval Science Department
Rensselaer Polytechnic Institute
Troy, New York 12180
518 266-8004

ABSTRACT

Due to the impact of behavioral issues on systems effectiveness, exposing students to these issues is an essential aspect of the CIS course. Unfortunately, there are very few experientially based behavioral activities available for CIS curricula. The majority of CIS course instructors are neither trained in nor comfortable teaching behavioral experiential activities. This paper presents six behavioral activities that have been used with great success in CIS courses. Hopefully this will begin a dialogue between technically trained instructors and those with more behavioral backgrounds on how to transfer behavioral methodologies to the CIS context.

INTRODUCTION

The rationale for this paper evolves from two concerns. The first is the premise that experiential materials are often more effective for conveying course materials than are lectures. Students need the involvement that lectures do not supply. A recent paper by McLeod (1) reports on the results of a survey of 113 schools of which 62 had an MIS core course in the undergraduate program. It was found that faculty teaching the MIS course often stress the need for experiential activity. "This activity is especially important when students have a modest appreciation for difficulties of managerial decision making and the role that MIS or DSS plays" (p.77).

The experiential activities reported varied from solving non-computer based case problems to writing programs, to working on a term project in a real organization, to using statistical packages or prewritten software. Aside from the use of case problems, none of the other experiential activities give students a feel for the role of MIS or DSS in managerial situations since there is little managerial component in those exercises. While these activities emphasize rational decision making they do not adequately address behaviorally oriented issues.

The second concern derives from the extensive literature in information systems pointing to the importance of behaviorally oriented components of successful information systems (2). A few such components might be the cognitive component of users, the involvement of stakeholders in the information systems design process and the need to communicate in order to be able to obtain good information in the systems analysis process. All of these behavioral issues are inextricably tied up with the technical aspects of information systems and essential to the success of those systems. So, although many respondents to McLeod's survey stressed experiential activities, it appears that there is a lack of attention to behaviorally oriented experiential exercises.

There are plausible reasons for this which all stem from a common theme: most people who teach MIS do not have a behavioral background and so may not feel competent/comfortable/experienced with behaviorally based activities or issues. This is neither surprising nor upsetting since we would not expect technically trained instructors to have a background in behavioral issues. And yet, the increasing importance of behavioral concerns would seem to auger a dialogue between technical and behavioral disciplines.

We hope to add to that dialogue by offering six "sure fire" behaviorally oriented experiential exercises for use in CIS courses. These are exercises that we have used with great success to effectively demonstrate behavioral issues involved in information systems.

Presented below are:

a. Name and objective of each exercise.

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

c. The discussions and learning experience which follow.

The following exercises are grouped in two sets of three. The first two exercises provides awareness of human differences and sensitizes students to the individuality that may be encountered among design team colleagues and clients. Finally, the effectiveness of positive listening in overcoming interpersonal communication blocks is experientially taught.

The second set gives the student insight into the process as well as the content of group problem solving activities in a way which will make the student more effective in group problem solving contexts. Both underlying destructive forces as well as constructive forces for group operations are aptly demonstrated.

Each of the exercises requires the use of some materials which can be obtained from the senior author.

Set One: Interpersonal Experience.

1. Exercise: Myers-Briggs Type Indicator Questionnaire (short-form).

   Objective: students are shown how they prefer to learn about and decide things in comparison to others.

   Method: students complete the questionnaire and are asked to identify the likely cognitive types in different occupations. The cognitive types among the class members are obtained. People with similar cognitive types are put together and asked to solve a brief systems design problem. The various group's solutions are discussed.

This exercise is usually a real eye opener for the technically trained students, who tends to think they and all their colleagues have similar problem solving preferences. In the course of discussion the student learns the existence of a continuum of problem solving style preferences, which are not stereotypable to sex, academic or professional discipline per se. This experience is usually the first one which motivates the student to deal with others on an individual basis.

Although the Myers-Briggs has been used extensively in IS research, it is not usually employed in the classroom. A short form of the Myers-Briggs takes about 10 minutes to complete. The use of this instrument provides a concrete example of different cognitive styles and the impact that different styles have on the design of computer based systems.

Important discussion of the implications for IS design follows logically from the classroom experience. Because of individual problem solving preferences, the same IS support will not be expected to have the same appeal for all clients.

2. Exercise: Active Listening.

   Objective: students are motivated toward effective communication through experience in the major factors which facilitate or block person-to-person communication.

   Method: students form groups of four--one speaker with three listeners--for each of four sub-exercises. The listeners are instructed to respond to the speaker in specified ways calculated to be destructive or facilitative of communication. Short discussion on the responses of the speakers to each type of communication highlights the exercise.

Many information systems are not effectively developed because of the poor communication between designers and users. Good communication is especially difficult between groups such as these with differing backgrounds and experiences. Since it behooves a system designer to become an expert in communication, the active listening experience in the exercise will provide a cornerstone for such expertise. Students leave this experience with improved awareness and real motivation for the listener's impact on improving communications.

3. Exercise: Ranking Job Characteristics.

   Objective: students learn differences between their motivational needs and others' involved in the information system such as programmers, managers, and users.

   Method: a job characteristics rating form consisting of 18 job outcomes (i.e., responsibility, salary, promotion, etc.) are rank ordered for different professional groups and oneself. Perceptions of the motivating potential for the different groups from the class are obtained and compared.

In order for people to perform effectively they need to be appropriately rewarded. Motivation to use a system, motivation of professionals to participate in the design of systems or motivation to maintain systems requires appreciation of the differing needs and requirements of the people involved in the various aspects of systems.

This exercise reinforces and adds emphasis to the individual differences first developed in Exercise 1.

Set Two: Teamwork Experience.

4. Exercise: The Fishbowl Exercise.

Objective: students gain experience in observing a group doing problem solving and learn to distinguish between process and content issues in group decision making.

Method: pairs of groups of 5 or 6 people each are required. One group sits in a circle inside the other group. The inside group (in the fishbowl) works on a group problem solving task while the outer group observes the process by which the task is solved (the process) using process observation worksheets. When the inside group has finished, feedback is given to the group and to individuals on their role in the process. The two groups then reverse roles.

Generally students do not consider the methodology, or process, for group problem solving. The tendency is for groups to jump into the middle of problems and begin discussion without any thoughts of how final closure on the solution will be achieved. Process observation reveals a number of attributes of successful group action, as well as a number of blockages to successful group decision making, such as individual dominance of outcomes, inappropriate discounting of some solutions, fractionalizing, etc. With recognition comes improvement which is recognizeable in follow-on exercises and group projects.

Attention to group process is critical to effective group functioning. Much of the management of systems as well as the useage, training and implementation of system requires effective group functioning. Designers and users, managers and programmers, and systems professionals, all must be able to work well together. This exercise is critical for understanding how to ensure good group management.

5. Exercise: Win as Much as You Can.

Objective: Students experience the effects of destructive competition in a multigroup environment. The payoff for successful conflict resolution is demonstrated.

Method: The context is a set of 4 teams. Each team must decide if it will compete or cooperate with the other 3 teams in a 10 round game. Points are accumulated that demonstrates the effectiveness of co-operation or competition. This is an especially lively and exciting experience that usually transcends being thought of as a classroom exercise.

Many organizations are prevented from being as effective as they could be due to dysfunctional competition between different groups, such as systems designers and systems users, between hardware and software engineers or between programmers and analysts. Quite obviously, effective systems require cooperation between quite different constituencies. This exericse works as a springboard for discussing these potentially competitive situations.

Class discussion following the exercise reveals that our culture contributes to a natural competitiveness which must be subordinated to group achievement overall. The need for cooperative conflict resolution for effective groups is made clear.

6. Exercise: Survival Exercise.

Objective: students learn by experience that the synergy of group solutions to complex problems are usually better than those made by individuals. Motivation to work effectively as a team in the process of problem solving is heightened.

Method: a survival situation is given, such as a plane crash in the Arctic or the Sahara, or a shipwreck on a desert island. First the individuals rank order items for survival and then these individuals work on a team ranking. The two lists are scored against a solution provided by appropriate survival experts. Individual and team scores are compared.

The value and synergism of team functioning is validated by experience and by the statistics brought out in the discussion of the exercise. Rarely does any individual's score exceed his team's score in this exercise, and, in fact, the team score is usually a dramatic improvement over that of individual's. This exercise emphasizes the need for attention to process and group communications themes, all leading to enhanced group performance. Since information systems are analyzed, designed, developed, implemented and used by and with groups of people, enhancing group functioning can only lead to more effective information systems. This exercise is a powerful one for demonstrating group functioning and is remembered by students for quite a while.

CONCLUSION

CIS has a strong behavioral component since we are talking about the management of systems which are used by people to support human knowledge and decision making. Nevertheless, there are few experiential and behavioral exercises in CIS courses that support more effective design of IS. By paying more attention to these issues, the probability that IS professionals will design better and more appropriate IS systems, get their designs implemented by clients and accepted by the end user is increased - which, after all, is the prime effectiveness measures of information systems.

REFERENCES

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

2. Leifer, R. and K. White, Designing Information System Task Teams, Proceedings of the Twenty-first Annual Computer Personnel Research Conference, May 1985, p. 112-118.

COMPUTER INFORMATION SYSTEMS DEVELOPMENT: AN INTERDISCIPLINARY APPROACH TO DESIGN AND IMPLEMENTATION

Judith L. Solano and Jack E. Leitner
Division of Computer and Information Sciences
University of North Florida, Jacksonville, Florida 32216

and

Kathaleen C. Bloom
Division of Nursing
University of North Florida, Jacksonville, Florida 32216

ABSTRACT

The Division of Computer and Information Sciences at the University of North Florida has implemented a two-course sequence designed to introduce the tools and techniques employed in systems development and to provide students with an opportunity to apply those tools and techniques in a systems development effort. Presenting the two courses has become an interdisciplinary effort with nursing faculty actively involved in the project portion of the courses as experts with knowledge of the processes involved in the problem area to be addressed by the system and knowledge of potential solutions to the problem. As expected, the students had an opportunity to put into practice the tools and techniques of systems development. More important, however, were the unexpected results due to the interdisciplinary approach. The typically insular lines of the University community were cut across, paving the way for cooperation in research activities as well as continued cooperation in the classroom.

INTRODUCTION

The Division of Computer and Information Sciences at the University of North Florida has implemented a two-course undergraduate sequence in computer information systems development. The courses are designed to introduce the tools and techniques employed in systems development and to provide the students with an opportunity to apply those tools and techniques in a systems development effort.

The content of the two courses reflects the traditional approach to the systems development life-cycle. The concept of splitting the content into a two semester sequence follows the curricular recommendations of the Data Processing Management Association. (1) The interdisciplinary approach to presenting the two courses represents, however, a significant departure from the more traditional approach.

SYSTEMS ANALYSIS AND DESIGN

The first course in the sequence, a course entitled Systems Analysis and Design, is intended to introduce students to the traditional systems development life-cycle and to the tools and techniques used throughout that cycle. Using Gore and Stubbe's text, Elements of Systems Analysis, the content of the course is divided into a treatment of the study, design, development, and operation phases of the system life-cycle. (2)

The study phase is that phase in which a problem is defined, alternate solutions evaluated, and a system representing the most feasible solution recommended. Instruction centers on fact-finding and analysis activities that can be used to gather information about the problem and to develop a definitive statement of the problem. The course also includes extensive discussion of procedures for identifying and evaluating alternate systems that will satisfy the specific objectives of the primary user, resulting in a solution.

In the design phase it is expected that a detailed technical design specification be produced for the system that was selected at the conclusion of the study phase. Students are instructed in the various flowcharting methodologies, including system and data flowcharting, for identifying and allocating all the processing functions of the system. Emphasis is given to input and output design principles and techniques. And, considerable attention is given to file design concepts, including data formats, record formats, file access methods, and file organizations.

The principal activities of the development phase are the design, coding, debugging, and testing of the computer programs required for the system.

Since program development is the subject of several other courses in the curriculum, this course undertakes to provide students only with a general overview, or review of program development activities.

In the operation phase changeover to and routine operation of the new system takes place. The course addresses the necessity for periodic performance evaluation and presents a method for managing requests for changes and enhancements to the system.

To give the students an opportunity to put into practice some of the techniques presented, a project is included as a part of the course. Given a problem, students are expected to conduct the typical activities of the study phase and to produce a comprehensive report, in which the problem is defined, alternatives are evaluated, and a solution is recommended. They are expected to follow-up the study phase with a design phase, producing the detailed technical design specification for the system solution that they recommended.

Students are divided into small groups of three to five people, to simulate the team approach to system development often used in many businesses. The group approach requires that the students use selected project management techniques to keep everyone working toward the same ends and to keep the project on schedule. Individuals are forced, many for the first time, to learn to work with others. They must learn to compromise and must learn the skills necessary to elicit compromise in others. They must also learn to conform to standards used to insure uniformity in the work they contribute to the end product.

Traditionally, the problems that the students have been assigned have come from the local community. Churches, schools, small businesses, banks, and government agencies have been among those who have participated in the course in the past. Each has had its own particular problem or need. To fulfill the requirements of the course, students have been expected to go out into the community to meet with their assigned client, study the problem, and produce the design specifications for the solution.

SYSTEMS IMPLEMENTATION

The second course in the sequence, a course entitled Systems Implementation, is designed to give students the opportunity to develop a functioning system. The principal activity in this course is the coding and testing of the computer programs that will make up the automated portion of a system.

It is expected that students enrolled in this course will have successfully completed the Systems Analysis and Design course and will have produced the technical design specifications for the solution to a data processing problem. The computer programs developed in this course will be developed to conform to the design specifications prepared in the Systems Analysis and Design course.

The students are again assigned to small groups, as they were in the Systems Analysis and Design course. If possible the group assignments from the Systems Analysis and Design course are retained, but more often than not the vagarious nature of student scheduling makes this impossible.

The first task of the groups, particularly those that have not been retained in tact from the Systems Analysis and Design course, is to do a detailed review of the design specifications. Often this involves further meetings with the client, to insure that the group and the client thoroughly understands the specifications as they were originally written. It is quite common for the group to have to incorporate changes to the specifications at this stage in the project.

The tasks of program design and development are begun in earnest as soon as the design specifications have been completed. The instructor works with each group supplementing their knowledge of program development with design and development considerations appropriate to their particular problem. Particular emphasis is placed on the application of design techniques and tools in the development of the program logic specifications. The groups are also encouraged to apply various code validation methods, as well as program optimization techniques.

The expectation is that, at the end of the semester, each group will be able to show their client a functioning system. In addition, it is expected that appropriate reference manuals for the people who will work with the system have been prepared. The course stresses the need for documentation and requires that, at a minimum, the students submit a programmers technical reference manual complete with source code and a users guide for their system.

THE INTERDISCIPLINARY APPROACH

Typically the data processing problems that have served as the basis for the student projects in this two course sequence have come, as previously mentioned, from the local community. The students have interacted with end-users, in studying their problems and developing solutions to them.

In a significant departure from the more traditional approach to the project, the Division elected to take a page out of the texts on expert system development. Of primary importance in the development of expert systems are those individuals with a particular expertise in the problem area to be addressed by the system. These experts serve as a resource to computer personnel by sharing their knowledge of the processes involved in the problem area and their knowledge of potential solutions to the problems. The use of these experts as resources minimizes the need for computer personnel to interact with end-users. (3)

During the last couple of semesters when this two course sequence has been offered the Division has used, as a basis for the project, a data processing problem that is of particular interest to the faculty of the University's Division of Nursing. The courses have become an interdisciplinary effort with nursing faculty actively involved in the project portion of the courses as the area experts.

Without ever having to set foot in a hospital and interview end-user nurses, students have developed a sophisticated system designed to analyze clinical data and generate nursing diagnoses that are specific to each individual patient's health state. The system uses these diagnoses to generate individualized care plans for each patient. The nursing faculty, working closely with the course instructors, have provided the students with the benefit of their training and experience in the development of the system.

RESULTS

As expected, the students had an opportunity to put into practice the various tools and techniques of systems development, producing a functioning system complete with detailed documentation. More important, however, are some of the unexpected results that are directly related to the use of an interdisciplinary approach rather than the traditional approach.

From the students' point-of-view, they were relieved of the requirement of trying to arrange to get down town for meetings with users, during business hours that often conflicted with their own work and class schedules. Their expert was right on campus and being a faculty member was more receptive to scheduling meetings at times typically convenient for students.

From the instructors' point-of-view, dealing with another faculty member made it far easier to manage the project phase of the courses. While faculty who are not familiar with computers have many of the same unrealistic expectations as other noncomputer users, they are far more sensitive to the limitations of student projects that must be completed within the timeframe of a semester. It was, therefore, easier to define and manage the scope of the project, insuring that there was enough work to provide the students with a realistic experience, but not more than could be reasonably completed.

From a broader perspective, the interdisciplinary approach to these two courses has played a part in accomplishing a major objective of the University for its faculty. It has been a priority of the academic wing of the University to provide for the computer literacy of the faculty. The close assciation, through these courses, of computer and information sciences students and faculty with nursing faculty has made a contribution to the nursing faculty's knowledge of computers and their applications in nursing. As a by-product, nursing faculty are now, with little help from computer and information sciences faculty, able to offer their own credit and continuing education courses on computers in nursing.

The association between the two faculties has also had a reciprocal effect, with nursing faculty contributing to the computer and information sciences faculties' understanding and appreciation of the field of nursing. This has laid the foundation for a long term relationship in conducting cooperative research projects in the medical information area.

The interdisciplinary approach to the Systems Analysis and Design and the Systems Implementation courses has served to cut across the typically insular lines of the University community with some positive results. The Division plans, in the future, to include other academic units as well as the Division of Nursing in the courses, broadening the experiences of our students and extending the benefits of the interdisciplinary approach.

REFERENCES

1. Data Processing Management Association. The DPMA Model Curriculum for Undergraduate Computer Information Systems. (1986).

2. Gore, M. and Stubbe, J. Elements of Systems Analysis. Wm. C. Brown Co. Dubuque. (1983).

3. Waterman, D. A. A Guide to Expert Systems. Addison-Wesley Publishing Company. Reading. (1986).

4. Ziegler, C. A. Programming System Methodologies. Prentice-Hall, Inc. Englewood Cliffs. (1983).

METHODOLOGIES FOR TEACHING CIS COURSES

Wita Wojtkowski, Ph.D., Boise State University, Boise, Idaho 83725, (208)385-1372
Susan Brender, Ph.D., Boise State University, Boise, Idaho 83725, (208)385-1327
W. Gregory Wojtkowski, Ph.D., Boise State University, Boise, Idaho 83725, (208)385-1227

ABSTRACT

This paper outlines several successful teaching methodologies which were applied in CIS courses in the College of Business at Boise State University. We cover techniques which are applicable in introductory CIS courses and techniques for teaching upper-division courses in System Analysis and Design and Software Design.

In each of these methods we emphasize the importance of the use of real-life projects. This keeps student interest high and makes the learning process highly relevant for their future careers. We also incorporate into CIS courses the study of efficient writing of system documentation.

INTRODUCTION

With the rapid changes in computer technology, CIS course content and instruction methods must also undergo significant alterations. One of the changes that has occurred in teaching methodology is the current trend toward "term project" oriented courses [18].

The introduction of the Fourth-Generation languages makes it possible to use real-life projects of considerable complexity [14]. Currently, System Analysis and Design (IS420) and Software Design (IS430) involve semester projects around which much of each course is structured. We believe that students gain significant advantages in courses of this type. For example, they gain an obvious conceptual advantage because term projects reduce many abstract CIS concepts to an understandable level via an actual concrete implementation. In addition, the real-life project gives students a hightened sense of accomplishment since they know that what they do will actually be used by others. This, most of the time, motivates them to learn more, so that their work will be good and really useful.

We believe that documentation skills need to be taught to CIS students at every level. This involves teaching students what documentation is, the various types of documentation and how to distinguish between good and poor documentation. Beginning students prepare limited program documentation. Intermediate students are taught to prepare extensive program documentation, and advanced students are taught how to prepare systems and procedures documentation which meets standard criteria for quality--readable, understandable, and usable.

One way to do this task is to incorporate the techniques for good writing into the process of teaching recognition and preparation of good documentation. With the increasing importance of end-user computing, procedures documentation is becoming much more critical. In the past, we had programmers writing documentation for other programmers. Now, the users of documentation, especially procedures documentation, often have very few if any computer skills. Good writing is becoming critical for productive use of documentation and of computer systems.

The purpose of this paper is to present the teaching methodology used in introductory and advanced CIS courses. We show how projects relate to the course content and how they are phased in during the semester.

NEW TEACHING METHODOLOGY IN INTRODUCTORY INFORMATION SYSTEMS CLASSES (CIS210 AND CIS220).

We utilize Team Learning/Informative Testing (TL/IT) teaching methodology [15] in introductory computer information systems classes. The rationale for use of this technique comes from recognition of the fact that the majority of our CIS graduates (who prepare for entry-level and mid-management positions in CIS) will invariably work in group settings [6] [12] [22]. Much information processing today is done in groups--from team programming [1] to managing information systems by committee [17].

TL/IT offers significant experiential learning about group processes and at the same time, facilitates learning of the subject matter [8]. In our classes, it offers students opportunities to learn more CIS and engage in meaningful, practical exercises of appreciable complexity even in introductory classes. Through this, we hope to enhance the likelihood that students will be able to integrate their learning into their future job situations. The TL/IT method represents practical utilization of ideas of organizational behavior [4] [5] and learning theories [8].

Briefly, a good learning situation should incorporate all three modes of learning--individualistic, competitive, and cooperative.

Individualistic learning, in which students seek outcomes that are personally beneficial and ignore the goal achievement of their classmates, is necessary for appropriate internalization of the subject matter [8]. Competitive learning, in which students compete with classmates for grades, corresponds to realities of the competitive working world outside the class [3]. Cooperative learning, in which students seek outcomes that are beneficial to all those with whom they are cooperatively linked, corresponds to appropriate use of goal structures in effective, productive teams [10]. Here students perceive that they can reach their learning goals if and only if the other students in the learning group also reach their goals. This has direct relevance to CIS; ideally, to complete any CIS project, participants should seek outcomes that are beneficial to all those with whom they are cooperatively linked. TL/IT incorporates all three modes of learning.

Instructional activity sequence which is used in TL/IT is presented schematically in Figure 1. Step 1 corresponds to the individualistic mode. Step 2 is competitive. Steps 3 and 4 utilize the cooperative mode. Formed cooperative groups are purposefully heterogeneous [6] and this heterogeneity brings about at least mild group conflict which, as organizational behaviorists point out, enhances productive group function and guards against "groupthink" [7]. Through Steps 3 and 4 of instructional activity, students are made aware of group functioning, its pitfalls, and its possibilities. We invariably note that those steps help individual students become more effective in groups as problem solvers. Moreover, the entire learning group becomes very effective and productive. This is especially evident in Step 6 where integration of all modes of learning takes place. Step 5, by the way, brings into focus the following: instructors become managers of the learning process, not dispensers of information only.

Management of learning processes also involves encouraging good writing in the process of teaching. Good writing is becoming critical for productive use of documentation and of computer systems.

These requirements make it necessary to teach good writing skills in our computer-related courses at all levels. Since writing skills are taught at almost every level of a person's formal education, one may say that this task is someone else's responsibility. Not true! These are basic communication skills which must be reinforced in every subject matter area. In our CIS courses, we must review the principles of good writing and hold the students accountable for writing.

In a survey course, a very limited amount of time is available for teaching documentation which, historically, has led us to concentrate on graphic approaches like flowcharts, structure charts, etc. However, it is possible, with the use of psuedocode, to illustrate a usable type of documentation and to show how good writing enhances documentation quality.

The characteristics of good writing, which we stress here only by example, are the same as those which are described in detail later in this paper where we illustrate the approach used to have advanced students actually prepare documentation.

The introductory courses are the only contact many of our students have with formal training in information systems from a literacy point of view. Here they get the concepts. They should learn to recognize good documentation so that later in their

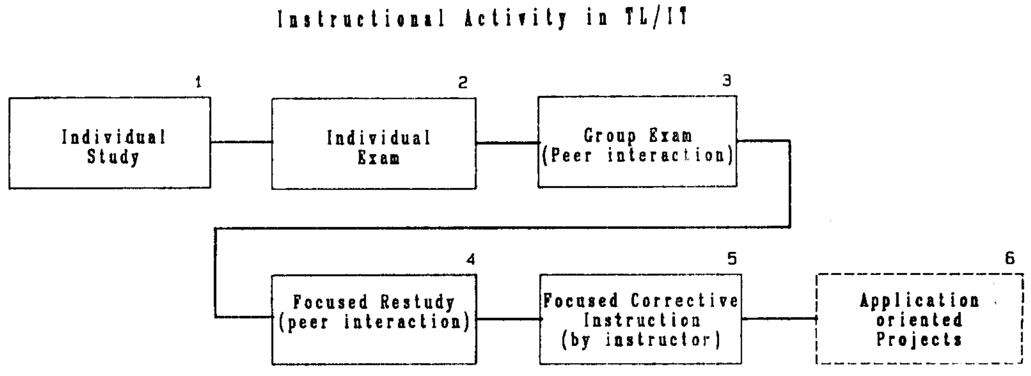

Instructional Activity in TL/IT
1
Individual Study
2
Individual Exam
3
Group Exam (Peer interaction)
4
Focused Restudy (peer interaction)
5
Focused Corrective Instruction (by instructor)
6
Application oriented Projects

Figure 1. Instructional activity sequence used in Cooperative Team Learning/Informative Testing teaching methodology.

computer-user careers they can demand quality documentation, because they will be able to recognize it.

THE SYSTEM-DESIGN PROJECT--CIS420

The System Analysis and Design class at Boise State University covers the system life cycle with the emphasis on user-oriented design [13] and introduces and applies ideas of prototyping [16] [23]. For their semester project, students are required to analyze the user's requirements and design the system. This project is a real-life application--it involves analysis and design of the budgeting system for the Idaho Legislative Budget office for the year 1986 [21].

The course begins with a general discussion of a system feasibility study and an overview of techniques and requirements of system analysis and design [11] [13]. During the sixth week of the semester, students are introduced to the users and their needs. (Students are made aware of the political nature of the user's environment and through this become cognizant of possible difficulties and needed compromises [21].) In the standard 15-week semester, roughly ten weeks are left for the completion of the project--three weeks are devoted to evaluating user's needs, three weeks to designing, three weeks to writing functional specifications, and one week to presenting the design.

Good writing of functional specifications becomes very important at this stage of the course. Here we incorporate the techniques for good writing into the learning process. Recognition of the real value of reviewing and applying good writing techniques in the system analysis and design course occurred to us during the last time this course was taught. In retrospect, we see considerable value in utilizing the same approach we used in the software course. This approach is described in the next section of this paper.

SOFTWARE DESIGN--CIS430

Software Design covers programming for the system, writing of documentation, and implementating the system for the user. Students in this class have already been exposed to the teaching methodology of our introductory courses; therefore, they work well in groups and are aware of group problems and possibilities. From the start groups are formed as business associations. That is, students enter them because they must. Groups are not social; they are business relationships directed toward goal attainment. Moreover, students worked in groups in System Design and are completely familiar with the user's project.

During the first five weeks of the semester, students are taught a Fourth Generation Language for the HP-3000, specifically 4GL POWERHOUSE. They write programs for the system in this language for the next six weeks, and the last four weeks of the semester are spent in testing and writing documentation. Such a venture is possible in this limited time period because we use a highly productive software tool [14].

WRITING DOCUMENTATION

We have been somewhat misdirected by the "chart" or "graphic" approach to documentation. These approaches make it appear as if ample and quality documentation is achievable without writing. However, we have seen problems even with the few words that are written in the symbols of "chart" documentation. With the increasing importance of end-user computing, procedures documentation is becoming much more critical.

Some people may say that psuedocode is so structured that a concern for writing technique is not relevant. We believe whenever words are put on paper, the intent to communicate exists; therefore, application of good writing principles is critically important.

Briefly, the following writing principles should be included: parallel structure, purpose, positive vs. negative words and tone, concrete vs. abstract words, conciseness vs. wordiness, clarity vs. obscurity, reader viewpoint, sentence types and variety in use, transition between and within paragraphs and sections of a document, active vs. passive voice, paragraph construction, pronoun reference, effects of redundancies, reader mindset, and writer mindset.

Students must then be required to apply these principles by writing documentation. The lack of success in much writing instruction lies in the way writing assignments are handled. Students often write an assignment, receive a grade, and go on to another assignment. The key to success is revision. With this in mind, we review the writing principles, the students prepare a section of procedures documentation, we grade this documentation for content and writing quality, and the students revise and rewrite when necessary. No final grade is given until quality documentation has been prepared.

CONCLUSIONS

We have outlined teaching methodologies applicable to effective teaching of CIS courses. Use of high productivity tools such as Fourth Generation Languages and prototyping allows us to undertake real life, complex projects which can be finished during typical 15-week semesters. We also incorporate instruction concerned with writing usable documentation for programs as well as for procedures into CIS classes.

Through all this, learning about CIS becomes directly experiential. Students truly learn by doing.

REFERENCES

1. Argyris, C., "Organizational Learning and Management Information Systems," Acc. Organ. 1985 Soc. 2, 3, 113-123.
2. Baker, F.T., Chief Programmer Team Management of Production Programming, IBM Systems Journal, V. 11 No. 1, 1972, pp. 56-73.
3. Bloom, B.S., Madus, G.F., Hastings, J. T., Evaluation to Improve Learning, New York: McGraw-Hill Book Company, 1981.
4. Couch, P.D., and Graf, A., "Diagnosing Group Climate to Improve Supervisory Effectiveness," Developments in Business Simulation and Experiential Exercises, Proceeding of the 11th Annual Conference of the Association for Business Simulation and Experiential Learning, (David M. Currie and James W. Gentry, editors), 1984, 72-75.
5. Dion, K.L., Baron, R.S., and Miller, N., "Why Do Groups Make Riskier Decisions Than Individuals?" In Advances in Experimental Social Psychology, 5, L. Berkowitz, editor, Academic Press, New York, NY, 1970.
6. French, W.L., and Bell, C.H., Organization Development: Behavioral Science Interventions for Organizational Improvement, (second edition). Englewood Cliffs, NJ: Prentice-Hall, 1985
7. Jalajah, D.S., and Sutton, R.I., "Feuds in Student Groups: Coping With Whiners, Martyrs, Sabateours, Bullies, and Deadbeats," Organizational Behavior Teaching Review, 4, 1984, 94-102.
8. Johnson, D.W., Maruyama, G., et al, "The Effects of Cooperative, Competitive, and Individualistic Goal Structures on Achievement," A Meta-Analysis Psychological Bulletin, 89, 1981, 47-62.
9. Johnson, D.W., and Johnson, R.T., (editors), Structuring Cooperative Learning: Lesson Plans for Teachers, New Brighton, MN: Interaction Book Company, 1984a.
10. Johnson, D.W., and Johnson, R.T, Cooperative Learning, New Brighton, MN: Interaction Book Company, 1984b.
11. King, W.R. "Alternative Designs in Information System Development" MIS Quarterly Vol. 6, No. 4, 1982, pp. 31-42.
12. Kolb, D.A., Rubin, I.M., and McIntyre, J.M., "Organizational Psychology: An Experiential Approach to Organizational Bahavior," Englewood Cliffs, New Jersey: Prentice-Hall, 1984
13. Lucas, H.C., Jr., The Analysis, Design and Implementation of Information Systems, (second edition), New York, McGraw-Hill, 1981.
14. Martin, J., Fourth-Generation Lanaguages, Englewood Cliffs, New Jersey: Prentice-Hall, 1985.
15. Michaelsen, L.K., Watson, W.E., and Scrader, C.B., "Informative Testing -- A Practical Approach For Tutoring With Groups," Organizational Behavior Teaching Review, 4, pp. 18-33.
16. Naumann, J.D., and Jenkins, M.A., "Prototyping: The New Paradigm for Systems Development," MIS Quarterly, Vol. 6, September 1982, pp. 28-044.
17. Nolan, R.L., "Managing Information Systems," by Committee. Harvard Business Review, 1982, 60, 4, 72-79.
18. Sathlin, H.L., "A Project Oriented Course for Software Systems Development," SIGCSE Bulletin, Vol. 16, No. 3, 1984, pp. 2-4.
19. Schneider, B.R., Jr., "Programs as Essays," Datamation, May 15, 1984, pp. 162-168.
20. Sigband, N.B., and D.N. Bateman, Communicating in Business, (second edition). Scott Foresman and Company, Glenview, Illinois, 1985.
21. Wojtkowski, W., G.W. Wojtkowski, "Design of the Budgeting System for the Idaho Legislative Budget Office," College of Business Boise State University, 1985.
22. Zmud, R.W., Information Systems in Organizations," 1983, Scott Foresman and Company, Glenview, IL.
23. Young, T.R., "Superior Prototypes," Datamation, May 15, 1984, pp. 152-158.

STRUCTURED METHODS AND THEIR PRACTICAL USE: Implications for Educators

Mary Sumner and Jerry Sitek
Department of Management Information Systems
School of Business
Southern Illinois University at Edwardsville
Edwardsville, IL 62026
618-692-2504.

ABSTRACT

The objectives of this study were to determine the extent to which structured tools and techniques are being used in actual projects during each phase of the systems development life cycle. One of the major findings was that although most of the respondents acknowledged the benefits of using structured tools in analysis and design, these tools were not being widely used in actual projects, largely because of their lack of acceptance by DP professionals and the perception that they were time-consuming to use. This situation might be remedied if educators offer inservice workshops and university-based courses for DP professionals in the uses of structured tools in systems design projects.

BACKGROUND

As educators we are often faced with the dilemma of selecting the best systems methodology to prepare our students for their future professional careers. While preparing our course outlines, we may think about what is being used in industry, but owing to the lack of time or information, fail to find out what structured analysis and design tools our students need to know. As a result, we often select the structured method we are most familiar with or prefer in doing systems work. In the end, we end up teaching tools and methodologies with little actual knowledge of the extent they are used and needed by our MIS graduates.

Among the tools and methods which can be used in teaching structured methodologies in systems courses are: the data flow techniques developed by Yourdon, DiMarco, and Gane and Sarson; the data structure techniques developed by Warnier, Orr, and Jackson; and the structured design approaches developed by Yourdon and Constantine. Tools used in detailed systems design such as HIPO (Hierarchy, plus Input, Process, Output), structured English, Pseudocode, and traditional flowcharts can also be added to this list.

In order to better develop our course content, we decided to learn about the tools and methods actually being used in systems development work. A search of the existing literature provided little insight into the current industry standard(s) for systems development techniques. What the literature search did point out was that designers are choosing from tools that are best suited to a particular design environment and then modifying them to meet their individual needs, with the tools used likely to change during the various stages of the project's life cycle. We decided the best way to gain insight into what to teach in the classroom was to conduct a survey of our own.

PROCEDURES

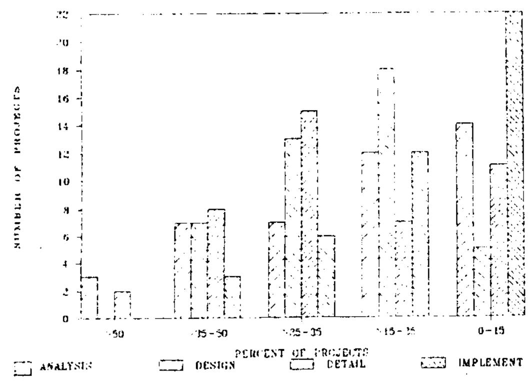
A survey questionnaire was sent to all the members of the St. Louis Chapter of the Association of Systems Managers (ASM). This ASM chapter consists primarily of systems analysts and project managers in St. Louis based organizations varying in size from small to large companies, government agencies, and non-profit organizations. In all, 172 members were surveyed and 27 percent (47 members) returned the survey. The respondents were asked to give specific information on a systems development project that they were currently involved with and to identify the tools and methods used at each phase of the project's life cycle, including requirements analysis, design, detailed design, and implementation. Specific questions addressed the organizational methods used during the project (i.e. structured walkthrough, etc.) and the strengths and/or weaknesses of the structured tools used.

FINDINGS

Most of the projects reported in the survey were new systems development projects, with 25 of the 38 projects in this category. The budgets ranged from $10,000 - 1,000,000, with fifty-eight percent (26) in the $100,000 - 500,000 category. Computer resources were categorized as mainframe, mini, and microcomputer-based. Mainframes dominated the projects with fifty-six percent (25); however, multi-computer based systems accounted for an additional thirty-two percent (14). Not at all surprising, given the projects' size, COBOL was used in fifty-five percent (27) of all projects represented; however, fourteen percent (7) of the projects involved the use of a fourth generation language to at least some degree.

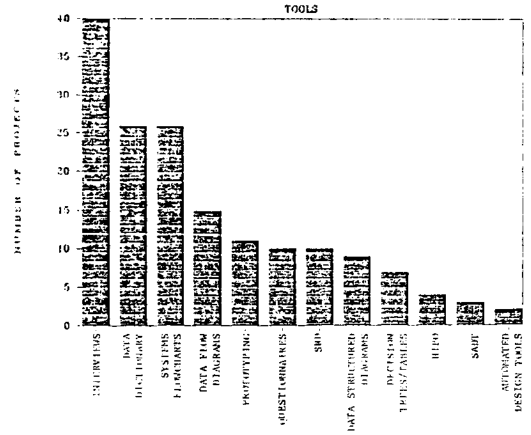
The data was analyzed for the percentage of time spent on each of the traditional life cycle phases; analysis, design, detailed design, and implementation. Analysis represented a range from zero to sixty percent, design from zero to fifty percent, detailed design from five to fifty-four percent, and implementation from four to fifty percent. Figure 1 represents the outcome.

FIGURE 1
PROJECT PHASES



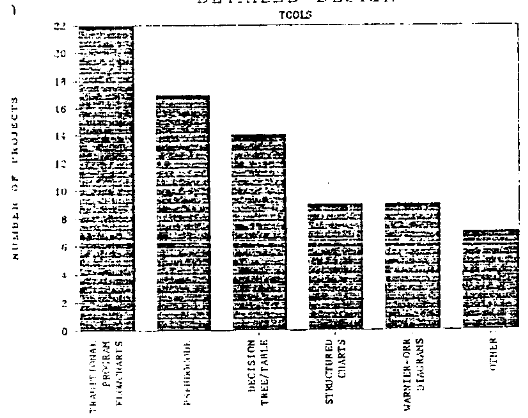
Traditional and structured methods were categorized on the survey and the respondents were asked to identify the methods used in their respective project's requirements analysis phase. Interviewing was used in eighty-nine percent (40) of the projects, followed by traditional flowcharts, used in fifty-eight percent (26). A data dictionary was used in fifty-eight percent (26) of all projects represented; data flow diagrams, thirty-four percent (10); and prototyping in twenty-five percent (11). Figure 2 represents the number of projects in which the various requirements analysis tools were used.

FIGURE 2
REQUIREMENTS ANALYSIS
TOOLS



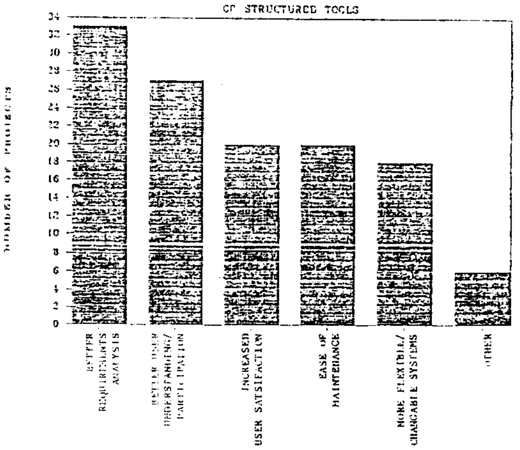
Detailed design responses also showed heavy reliance on traditional flowcharts, with forty-nine percent (22) of the projects using them during this phase of the life cycle. Pseudocode ranked second with thirty-eight percent (17), followed by decision trees/tables. Figure 3 represents the tools used during this phase.

FIGURE 3
DETAILED DESIGN
TOOLS



Although traditional systems development methodologies were used in most of the projects surveyed, many of the respondents advocated the use of structured tools during the projects life cycle. When asked to identify the major benefits associated with using structured tool(s), seventy-four percent (33) of the respondents indicated they believed the number one benefit is better requirements analysis, followed by better user understanding/participa-tion. Figure 4 represents the perceived benefits of using structured tools.

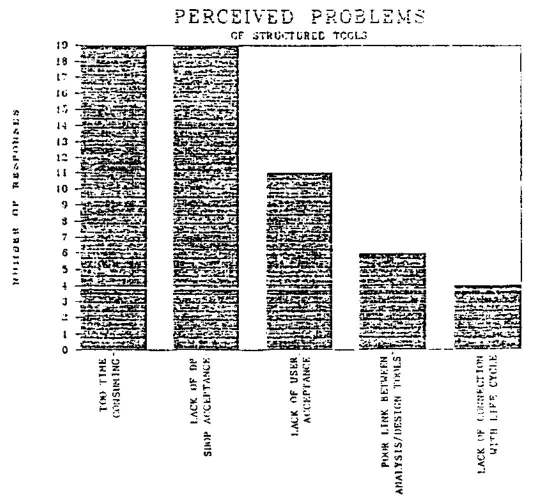
FIGURE 4
PERCEIVED BENEFITS
OF STRUCTURED TOOLS



When asked to identify the perceived problems associated with using structured methodologies, two problems were identified by an equal number of respondents (forty-three percent (19)): too time consuming and a lack of the data processing shop's acceptance. Not as surprising was the next highest perceived problem, a lack of user acceptance, with twenty-four percent (11).

Figure 5 reflects the perceived problems associated with using structured tools.

FIGURE 5



IMPLICATIONS

The study has several implications for educators. First, it appears that we cannot give up teaching traditional flowcharting techniques because they are used in over half the projects in the requirements phase and in slightly less than half of the projects during the detailed design phase. Other tools that we need to incorporate into our lectures are: data dictionaries, data flow diagrams, structured requirements definition, and structured design. Given the mixture of the methods being used, the best approach may be to provide students with an overview of the various methods and an indepth skill in at least one of them.

The results of the survey provide insight into what tools are being used in systems development projects and provide some very interesting "food for thought" with regard to future classroom instruction. It was particularly interesting to note that while almost every respondent advocated using structured tools, they did not appear to be accepted in actual projects.

As educators, we question whether this is a fault of the methodologies and tools or a lack of understanding and education on the part of data processing professionals. If data processing professionals feel undertrained and hesitant to use these tools, there are implications for inservice education. If DP professionals simply do not have the time, manpower, and budget to practice the structured methodologies, the implication is that the methodologies must be streamlined and simplified to be put to practical use.

This study points out the gap between the classroom and the work environment with regard to the structured tools. If we believe in the merits of structured system development, then we need to shift some of our efforts to educating data processing professionals on their merits. The answer may also lie in continuing to teach our students state-of-the-art structured methodologies and wait for them to apply these techniques in industry.

REFERENCES

Colter, Mel A. "Evolution of the Structured Methodologies." In Cougar, J.O., Colter, M.A. and Knapp, R.W. Advanced System Development/ Feasibility Techniques, New York: John Wiley and Sons, Inc., 1982, pp. 73-96.

Horton, John B. "Are the New Programming Techniques Being Used?" Datamation, July 1977, pp. 97-103.

Jenkins, A. Milton, Naumann, Justus D., and Wetherbe, James C. "Empirical Investigation of Systems Development Practices," Information and Management, V. 7, April 1984, pp. 73-83.

Shevlin, Jeffrey L. "Evaluating Alternative Methods of Systems Analysis," Data Management, April 1983, pp. 22-25.

Wasserman, Anthony. "Information Systems Design Methodology," Journal of the American Society for Information Science, John Wiley and Sons, 1980, pp. 43-62.

Zolnowski, Jean C. and Triney, Peter D., "An Insider's Survey of Software Development," Proceedings of the 6th International Conference on Software Engineering, Tokyo, Japan, September 13-16, pp. 178-187.

PORTFOLIO SELECTION AND MANAGEMENT: COMPUTER-ASSISTED INSTRUCTION DESIGN

JOANNE A. COLLINS
GEORGE H. JACOBSON
AND
GEORGE V. RAPTIS

California State University, Los Angeles

The authors have designed a simulation of investments for upper-division and graduate-level courses. The student assumes the role of portfolio manager and is provided general information regarding forecasts for the economy, stock market, et cetera over the next quarter. He/she then selects from a list of stock funds, bond funds, or money market fund. If the student selects a portfolio congruent with the external conditions, he/she is "rewarded". If additional information is desired for decision making, it may be purchased, the higher the probability of a "correct" decision, but the lower the potential reward. The simulation is designed for four quarters to provide students the opportunity to learn how to adjust portfolios to changing conditions.

The purpose of this paper is to design a computer simulation to supplement textbook instruction in an upper division or graduate-level course in investments.

The authors feel that simulation provides the best method of utilizing computer-assisted instruction (CAI) for the teaching of advanced financial concepts. Chambers and Sprecher (1, p. 92) concluded that "in general, a combination of strong positive reinforcement for the correct response and mild punishment for an incorrect response has been found to provide optimal support for learning." The same authors emphasize the usefulness of color, graphics and sound to strengthen learning through reinforcement (1, p. 141). Additionally, Chambers and Sprecher (1, p. 96) felt that learning is enhanced when the student "invests" something of himself/herself, even something as simple as entering the student's name. All of the above factors are included in the following simulation.

Due to the heuristic nature of decisions to be made and student motivational factors, simulation is selected by the authors as a better method of supplying instruction in a course in investments to advanced students than either drill-and-practice or tutorials. Drill-and-practice is ideal "where students need to memorize certain facts rather than to understand complex concepts" (2, p. 286). Tutorials are more difficult to adapt to subjects that require conceptual knowledge (2, p. 287).

The educational purposes of the design are to refine skills in investment selection and portfolio management by simulating the management of a simple portfolio for a corporate pension plan. The exercise is to be used to integrate material and develop practical skills. The design is planned for courses in Finance/Accounting and Portfolio Theory. Specifically, Investment courses in Finance; Continuing Education courses; Finance 332 (Investments); and Finance 534 (Seminar--Portfolio Theory) at California State University, Los Angeles.

The goal is to allow the students to develop comprehension and skills in portfolio management, specifically in the proper proportions of stocks and bonds with money market funds providing possibilities for defensive posture.

In the scenario (see below) the computer will hire the student as a manager of corporate pension portfolio, consisting of 3 types of stock funds, 3 types of bond funds, and money market fund. The types of investment are chosen to represent different levels of risk and return. The provision of a position (which may be changed based on performance), a salary, bonuses, and perks which are adjusted appropriately make this a portfolio management video game for business students. More accurately, it parodies a possible career path and this is intensely interesting to them.

The student is provided with a number of quarters of prior performance for each instrument, as well as a number of external predictors (GNP Trends, CPI Trends, Performance of Certain Stock Indices). In addition, there are objectives specified for the fund. The student has to contend with both risk and return concepts. Given a beginning fund amount, information on specific instruments, and other data, the system simulates performance of the portfolio over the quarter. The student may in making his/her selection use help messages to reinforce understanding of valuation and theories and to review prior performance.

At the end of each quarter, performance will be calculated and compared with objectives. As a result, adjustments will be made in salary, bonus, perks, or even position. The student will then repeat the process for 3 more quarters. The final objective, subject to an acceptance level of risk, is to optimize both the performance of the fund and the personal performance of the manager. These may not be congruent. (An optional index of measurement can also be developed based on the individual measures of fund and personal performance to illustrate possible problems of goal congruence.)

The simulation may also allow for purchase of specific information (such as economic trends over the next quarter). Thus, the simulation may allow for both free (instructional) and non-free help messages.

As part of the motivational strategy, the student is provided with a number of learning reinforcements. Specifically, the periodic adjustment of salary, bonus, perks, and position as well as the indices of performance (points) provides a great deal of motivation to business students. This may be further enhanced by animation and graphics that depict the rising or falling fortunes of the student. There also can be a promotion of the student from his/her initial position, signaling ultimate success (winning the game).

More specifically, graphics will be used (line and bar graphs) to provide information on specific investment instrument performance. The trends of external predictors, the representation of purchased information, and finally the comparisons of actual performance with objectives.

Also, color will be used effectively with the graphics to illustrate more clearly trends and results. Since there are several instruments and actual versus planned performance to consider, colors will be very helpful (e.g. red indicates poor performance). Another area where color will be used is in the animation of the boss talking to the student, the animational representations of salary, bonus. and perks (a stack of money, a corporate car) and occasional bursts of color to indicate significant events (stock market bust, etc.).

Finally, sound will be used to reinforce good or bad performance (rising or dropping), specific events (a crash may be appropriately reflecting stock market behavior) or achievements - the sound of coins; the honk of a new auto (a perk); appropriate music "We're in the money" at points of good performance; ruffles and flourishes announcing good performance, a promotion, and winning the game at the end.

Of course, it is important that the CAI design sustain the student's interest and even be exciting and engrossing as part of the learning process. To this end, the animation (including the boss, representations of stacks of money and perks), the motivation system which is in many ways realistic, and the video game quality of the design (with final points and the possibility of winning) will make this an engrossing experience for students.

The context of the game in the management of a corporate pension portfolio is designed to insure some conservatism. However, the obsession of society with the stock market and personal investments makes a very personal identification with such a game possible for most students and certainly nearly all business students.

Instructional Strategies

Viewed as part of the instructional process, the CAI design has several behavioral objectives. The major behavioral objectives of the simulation are to (1) develop skills in the selection and management of investments, (2) provide an opportunity for the student to perform in a relatively complex situation against pre-set but possible changing goals, (3) provide practice in operating under a relatively realistic set of incentives and penalties (i.e. the "real world" that business students are anxious to experience), (4) consider the value of information is worth its cost, and (5) link world practical experience on a personal level (e.g. the management of personal investments).

Furthermore, the design uses decision support systems that offer hints and help at strategic points in the program. These systems are contained in part in the evaluation process over four quarters. The student receives instantaneous evaluation of performance four times during the simulation, the results are reinforced by salary increases, bonuses and perks, and hopefully the student will use the results of one period to improve performance in the next period. In addition, there is the possibility of purchased information and the value it has (if any). Furthermore, there is the ability to retrieve prior information on investment instrument performance and the past performance of economic indicators. Finally, there is the possibility of the student asking for help prior to each decision on investment valuation principles and their relationship to the objectives he/she is asked to attain. Of course, the computer "hires" the student at the beginning of the game, asks for his/her name, and addresses the student by name throughout the game. For examples of how these interactions flow, please refer to the below scenario.
From a pedagogical point of view, the primary type of learning is problem solving and goal attainment. However, there will also be some attitude change as the student evaluates the value of information versus its cost and considers the potential of non-congruent behavior from the performance evaluation and reward system.

As a final point, there are several principles of learning embedded in the design. The primary principles of learning are reinforcement and repetition. The process is repeated four times, feedback and primarily positive reinforcement through attainment of objectives and personal rewards is elaborately offered, and the student has the chance to repeat the experience drawing heavily on information obtained from the preceding iteration. The relatively realistic environment of the design and the types of rewards offered add a great deal of motivation to the design. Also, the value of "learning by doing" as a means of integrating theoretical knowledge into "real world" practice, should not be under-emphasized. Finally, this is a portfolio management video game. It provides some of the fun and fascination of the arcade. Perhaps the motivational devices and the subject matter are a bit more serious and dignified, but in a real sense the design provides a game that can be "won" - the value of the aspect both on motivation and skill development are obvious.

In summary, this design teaches rather complex principles in an exciting "game" format. The solid principles of learning that underlie the design combined with the format assume its success as an effective learning tool.

REFERENCES

1. Chambers, J. A., & Sprecher, J. W. (1983). Computer-Assisted Instruction: Its Uses in the Classroom. Englewood-Cliffs, New Jersey: Prentice-Hall, Inc.

2. Stern, N., & Stern, R. (1983). Computers in Society. Englewood-Cliffs, New Jersey: Prentice-Hall, Inc.

SCENARIO

PORTFOLIO SELECTION AND MANAGEMENT

Scene on monitor: Animation of Mr. Folio, Director of Personnel for the Very Big Corporation.

Good day: You are being considered for a position of portfolio manager for the pension fund of Very Big Corporation. May we please have your name for our personnel records.

Name --------------------------

Congratulations ----------------, you got the job! As you know, retirees at Very Big (including your economic future) depend on your investment selection and management skills.

You will be on probation for one year. During this time we will evaluate your performance to decide your long-term future with Very Big. You will have the opportunity to invest an initial fund of $10,000,000 into any or all of seven investment instruments.

(1) 3 stock funds - one of these has high growth potential (greater than 25%) but also has high risk of loss, as measured by its standard deviation. Another has moderate growth with correspondingly less risk, and the third fund offers low growth (less than 8%) with correponding assurance of preservation of capital.

(2) 3 bond funds structured along the same dimensions of risk and return as the equity funds.

(3) A money market fund for a defensive position as needed.

The past eight quarters' performance of each of these seven intruments is given to assist your decisions.

		Q1................Q8
Stock Fund	#1	
	#2	
	#3	
Bond Fund	#1	
	#2	
	#3	
Money Market Fund		

In addition, data for the past eight quarters for the GNP deflator, the consumer price index and other selected/coding indicators are also given.

	Q1................Q8
Indicators	

Your objective is to attain a return of at least 10% in quarter 1, 12% in quarter 2, 12% in quarter 3, and 12% in quarter 4. These objectives may be altered if it appears that the economy will not support such return. You also must not exceed certain levels of appropriate risk for the portfolio measured by keeping the variability of returns for the equity instruments within +/- 10% and keeping at least a minimum amount of the total debt investment in higher rated bonds. A composite index for risk and the risk values assigned to various bond ratings are given below:

$$\text{Risk index} = \text{Weight}_1 \times \text{Stock risk} + \text{Weight}_2 \times \text{Bond Risk}$$

The index is not to exceed N. To the extent that the portfolio you select exceeds the target value, penalties will be assessed against your performance records.

You will start with a salary of $50,000 and a company car (a Chevrolet).

Exceeding the return objective in a quarter by at least 2% will result in salary increases, bonuses, and other perks, the total value of which will be given for your tax records. Your personal objective will, of course, involve maximizing the value of your salary and perks given that you meet corporate objectives. An outstanding performance will also earn you a promotion. A dismal performance will terminate the program.

You will make your selections quarterly. The entire fund allocated to you must be invested somewhere. You will then be evaluated on your performance for the quarter and rewarded accordingly.

You may, at each decision point, purchase information about future economic trends (i.e. the value of the indicators for the next four quarters) by paying $5,000-$10,000 from the fund balance and $5,000 from your salary to Collins Econometric Forecasting Associates (CEFA). This information is, of course, only imperfectly accurate.

Good luck ______. Let's begin. Do you wish to examine the prior performance of the investment instruments once more? Yes/No

(If Yes - Repeat table on prior performance; If No - Continue)

Do you wish to examine the prior performance of the economic indicators once again? Yes/No

(If Yes - Repeat data on prior economic activity; If No - Continue)

Do you wish to purchase information on economic trade? Yes/No

If Yes: The current value of your salary and bonus is (deduct $5,000)
The current value of the fund is (deduct $5,000)

Show table on economic indicators for next 4 quarters -

	Q1..............Q4
Indicators	

(Information can also be shown moving in ticker tape fashion across screen.) This information is redetermined stochastically each time the program is run, and updated each quarter. The actual performance of the instruments is determined from the predictions with varying (randomly determined) degrees of reliability. The degree of reliability used in determining the actual, if forcast is purchased, is printed out with the actual results.

(If the answer to "Purchase Information" is No, continue with simulation.)

The results of Q1 are as follows: (Also use graphics)

		<u>Amount Invested</u>	<u>Return</u> (Detail)
Stock	#1		
	#2		
	#3		
Bond	#1		
	#2		
	#3		
Money Market			

The current value of the fund is now ______. The weighted average return on your portfolio was ______.

Your performance against objectives has won you the following:
(If bonus or salary increases are earned, animation of growing stock of money, tune: "We're in the money"! If a bigger auto is earned, adjust animation accordingly with proper auto horn sounds.)

Repeat simulation for quarter 2, 3, 4. Show final results, animation job final stock of money, bigger car or even corporate plane (if appropriate).

A possible final scenario reflecting super performance might be:

Congratulations____________! (Appropriate music flourishes) Your performance as a portfolio manager has been superb. We are promoting you to division manager of our High Flying Division. Please check appropriate CAI software for your instructions (A CAI sequel?). The final performance of your division was a weighted average return of ______%. The final value of the fund was $_________.

Your final personal status is:
(annualized)

Salary $
Bonus $
Perks :

The final value of your salary, bonus, and perks is $_________.

PROVIDING PROJECT MANAGER EXPERIENCE IN THE COLLEGE CURRICULUM

Dr. John Cordani, Baruch College, C.U.N.Y.

ABSTRACT

Baruch College has instituted a project in its Office Administrative and Technology course of study which is designed to provide students with both live experiences in active research and in the application of their findings in live operations. The attempt is made by the faculty to integrate all that students have learned in their undergraduate studies in managing a major project in Office Automation. All the frustrations, and a few of the joys of success are experienced by students in a live environment. The demands on both the students and the participating faculty members are discussed. The focus of the experience deals with an examination and implementation of information systems.

PROVIDING PROJECT MANAGER EXPERIENCE

Study of office administration and technology is an important and growing discipline at the college level. A major in office technology should provide the student with active research skills, management skills, office technology including word processing, spreadsheeting, and data base utilization. As a culminating experience, the student should be provided with an opportunity for business systems planning in a capstone course. To be effective, the systems management experience in such a capstone course must be structured to provide for individual and group development of active research skills, for development of isolated skills and knowledges learned in previous courses, and for use of electronic teqhnology applications.

Traditionally, the capstone experience for office administration has been a simulation, perhaps prepared by a business publisher, and delivered in a microcomputer laboratory. In the simulation the students use microcomputers for wordprocessing and for rudimentary calculations. The simulation provides opportunity for students to experience a business operation, but lacks the true office challenge: project management--managing the integration of systems, people, and computers.

An improved capstone experience provides students with opportunities to experiment with management techniques learned in previous courses in the solution of real management information problems. Prerequisite courses should expose the students to techniques and strategems that utilize the capabilities of microcomputers in business. Students can only survive the rigors of such a capstone experience by coming into the course with at least a rudimentary knowledge of management skills and with a good grasp of the three cornerstone programs of office technology: wordprocessing, spread sheets, and data base programs.

The students should have access to and be able to use microcomputers as word processors, for spreadsheeting, and for database manipulation. The role of the microcomputer as a management tool is justified by the current literature.(3) (5) In this type of improved capstone experience, the students should acquire experience in using the computer as a scheduling tool, as a budgeting tool, as a graphics tool, and of course as an excellent tool for preparing reports.(4) The course should be designed to provide an opportunity for students to experience application of management theory and techniques and to integrate the tools of automation with management.

An instructor does not have to search very far to find an environment rich in need of improved communication and the implied need for improved office automation--the department, the school, and the university in which the students and faculty already reside provide an environment ripe for office automation planning, development, and implementation. As in all businesses, the department needs to plan for its acquisitions based on limited resources, both for the short and the long term. Usually no master plan exists for future computer resources. In most school environments, the lack of a master plan has often been explained away as being a conscious desire to allow each school, each department, each program, and indeed, each faculty member the academic freedom to pursue his or her research and teaching interests with whatever tools that could be acquired. This method of acquisition of resources results in chaos. A planning cycle should be undertaken, needs assessments made, and goals set.(1) (4) Even if a college has taken steps to coordinate automation within its various operations, it has probably directed that all computer related purchases be funneled through one staff officer, such as the business manager, requiring each unit of the college to prepare proposals with needs assessments and goals.

At Baruch College of the City University of New York the B.B.A. major in Office Administration and Technology prepares the graduate for numerous existing positions in business and government. The capstone course, ADVANCED OFFICE ADMINISTRATION AND TECHNOLOGY, provides opportunities for students to act as project managers planning office automation. This approach was developed to provide a structure wherein students could experience all of the achievement and frustration in creating a real office information system. Project management training in the capstone course at Baruch College has extended over three semesters. The students in each succeeding class take over the planning at the stage the previous class left it. The students in the first class set out to recommend a better information system for the School of Education, a unit of Baruch housing two departments in two different buildings, four program centers, thirty-two fulltime faculty members, and numerous staff and student employees.

The information needs of the school were determined, the present state of automation in the school was considered, ideal solutions were analyzed and adjusted to actual budgeted resources available, and recommendations were prepared for an automation plan. Along the way, the need for organizing the students and their work became evident. An overall operation scheme was developed from authoritative references. (1) (2) and (4)

The students defined the operational steps they would take and determined a realistic time frame for themselves. They used the software package, Visi-schedule, to control time and expenses and for forecasting target dates and for controlling personnel resources. That the experience was live and in the real world was brought home to the students when interviews were missed or incomplete, when misinformation was gathered, when interviewees appeared reluctant to confide to students their deepest needs as faculty or staff, or when analysis of incomplete findings clearly indicated the need for re-planning and rescheduling.

In each succeeding semester, the conduct of a live study forces the students to depend upon each other in the production of their final work in term reports. In the beginning of each term students exhibit a great deal of laisse-faire in their internal coordination. As the term progresses it becomes obvious to the students that project planning by team effort extends individual capabilities far beyond sheer time and effort. Increasingly, students become more focused on shareability of material.

Experience has shown that the use of a scheduling package should begin with the initial planning session. The use of microcomputer spread sheet programs makes calculations and tables possible and renders constant updating of the project "doable " rather than a task only discussed in the abstract. The combination of the microcomputer as a resource and the live environment of a project are an excellent match. The students are able to track, control, and redirect their efforts as required by work with real problems and with real people.

Study of the internal flow of information and knowledge of short response times highlight the advantages of standardization of report formats and the desirability of "boiler-plate" and compatible systems and programs. The "boiler-plate" technique allows students to execute a portion of the report individually and yet maintain the structure and logic common to the team report as a whole.

The most used microcomputer programs were those programs which permitted the graphic presentation of time schedules and the integration of spread sheet projections. The programs used in the course include Visi-Schedule and Visi-Plot. Once a week the projected work schedule graphic was compared with a graphic of the actual work completed to date. This became a primary management control technique used to direct the entire project. The use of the microcomputer in this role greatly enhanced the ability of the students, in their role as project managers, to use their visual senses to track progress and redirect resources when necessary. The neat appearance of the graphics further facilitated rapid understanding and, therefore, rapid progress of the project. Students did not have to understand hurried graphics and notes scrawled across a chalkboard. The students quickly grasped the impact of microcomputer graphics as an integral element of their communications with the administration, the real consumer of this project. Microcomputer graphics capabilities were exercised in the presentation of overall project plans and resource requirements. At the presentation of the preliminary report, the first semester's output, graphics proved a key to clear understanding on the part of the administration as to future requirements. Of particular note was the comment made by the Dean of the School of Education that this graphic allowed him to understand why the project must continue over an extended period. Current estimates are that two full academic years are required to complete a business systems planning project. The integration of budget with schedule allowed the students to quickly convey the interrelationships between the support requirements of the project and the speed with which the project could be accomplished, expressed in terms of cost in dollars and time.

The approach used in Advanced Office Administration and Technology allowed the students to experience the tools and problems facing managers planning for and implementing an automation project in the live environment setting. The participants come away from the study having experienced a semester of real activity in applying their developing skills to true purpose. Combined with the live environment, Advanced Administrative Office Technology is a combination of classroom development of theory, direction and motivation provided by an instructor, and integration of subject matter from previous courses. The live environment approach has led to this secondary result: The college faculty and administrators, having been interviewed by the students, are more aware of their individual requirements for access to information and automated information support.

BIBLIOGRAPHY

(1) Hussain, Khatech M., Development of Information Systems for Education, Prentice-Hall, Englewood, NJ, 1973.

(2) Martin, James, An Information System Manifesto, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

(3) McLeod, Raymond, Jr., Decision Support Software for the IBM Personal Computer, SRA, Chicago, IL, 1985.

(4) Murdock, Robert, Joel Ross, and James Claggett, Information Systems for Modern Management, McGraw-Hill Book Co., New York, 1984.

(5) Taffee, Stephen (Ed), Computers in Education, 1985-1986, The Pushkin Publishing Group, Inc., Sluce Dock, CN, 1985.

COMPUTER GRAPHICS IN THE CIS CURRICULUM

Syed V. Ahamed

Professor Computer Science
College of Staten Island
130 Stuyvesant Place
Staten Island, New York 10301

ABSTRACT

A large segment of undergraduate (UG) and graduate students with diverse backgrounds are taking courses in Computer Graphics. Interest triggers the first sparks of enthusiasm and the opportunity to be creative and integrate the new area of study with almost any subject of student interest carries the student through the follow up undergraduate and graduate courses. In this paper, we suggest topics of instruction for one or two courses of UG and graduate curriculum. As a continuation, we suggest that the topics to motivate students to do their projects be from an area of student specialization or interest. The chief area of concern is the wide variety in the background of the students who take these beginning, intermediate, or these advanced courses as the first course in computer graphics.

I. INTRODUCTION

Computer Graphics (CG) has impacted almost every phase of industry and commerce. The most dramatic impact has been in the high technology industry. Most of these industries cannot simply survive without CG. High technology demands the effective use of computer systems. By definition the processing of a large amount of data to seek certain conceptual relationships and/or causal effects demands the effective integration of the CG with simulation and design software. The numerous directions for the effective use of CG in any one aspect of high technology computer aided design can enhance the demand for CG expertise many fold, thus multiplying the need for specialized hardware, software and personnel. This is reflected in the increased enrollment in the CG courses, and most students rightfully see CG as an entry to a very fertile area of Computer Aided Design/Computer Aided Manufacture (CAD/CAM), computer vision, thus leading on to Pattern recognition, certain specialized topics in Robotics and Artificial Intelligence.

Medium and low level technologies acquire CG as an effective way towards high technology tools in their areas of expertise, thus extending their growth by increasing the product line and productivity. Both tend to hold a promise of enhanced profits for two reasons. First, the increased visualization of conceptual relationships in their own environment leads to an onset of a new wave of creativity in this rather sluggish field (e.g., the introduction of zip code scanners in Post Office environment) or when human accessibility can be dangerous (Robotic control of nuclear plants). Second the new tools and methodologies (e.g., Robotics in American Automobile and Japanese auto and steel industry), provide flexibilities in areas where the proven manufacture technique led to saturation and limitation in profits. These benefits have been discovered and incorporated in most of the modern manufacturing environments. Industries that isolate themselves from the potential impact of CG and the subsequent innovation face a grim future.

II. THE ROLE OF CG EDUCATORS

Industrial research, and CAD environments have proven use for CG and the more advanced topics in Artificial Intelligence that are subsequent to the basic CG courses. However the growth of research and innovation in CG has been exponential over the last 10 years because of increasing potential applications. Hence the role of CG educators gets intensified because of three distinct responsibilities placed upon them. First, the educators need to integrate the courses they teach in context of the possible application and usage of the material they teach in industry, research, and design. Second, the responsibility they have is to teach a very wide spectrum of students with diverse backgrounds in their undergraduate and graduate environment. Third, they have to teach courses in a subject material experiencing a very rapid growth starting with the standardization of the language to the implementational technology. The growth of specialized VLSI and specialized software components causes an additional element of flexibility on the material and the emphasis in these courses.

III. A PROPOSED UG CURRICULUM

At this level, the students are likely to be in the Associate or the Bachelor degree program. The employment would generally be at technical and/or entry level engineers and programmers. Here the principles on which the functioning of the CRT are based, the low level graphics commands and their integration into other software are suggested for the first quarter of the semester (about three weeks). During the second quarter of the first course the basis of communication between the CPU (as assembly level instructions) and any one of the established standard graphic terminals in conjunction with their incorporation in the high level languages (such as true basic, turbo pascal, Fortran, C languages, etc.) is proposed. During the third quarter, the differences in the operation of monochromic direct view storage tubes (DVST) and raster scans is suggested. An overview of other output devices such as plotters, graphic printers, color graphic devices would also be desirable. During the fourth quarter, the functional roles of the graphics interface board, the associated graphic processes in context with the standard functions such as scaling, translation, windowing, rotation, etc., would be very desirable. We also suggest a lecture for the graphic input functions for DVST with Graphics input (GIN) mode and Raster Graphics with a light pen.

In the undergraduate course a Graphics Laboratory experience to learn (and play) with microcomputers is desirable. The object of this one semester course is to provide an insight for the computer technician with an Associate degree and with the basic functioning of the hardware and for the low level programmers with some understanding of providing interfacing a graphic device and incorporating the assembly level programs with higher level application programs. The foregoing objective is satisfied by the course work. The laboratory fulfills this more fundamental objective of utilizing the inexpensive hardware and software microcomputer graphics in the highly variable applications environment. In a four credit course meeting about 3 1/4 hours a week, we suggest seven laboratory assignments starting from point, line, and box commands to an advanced project to use the microcomputer for a project of the student's choice. The typical fourth assignment consists of writing a higher level routine to translate, scale, and window any segments of a picture they may have generated in the third assignment. In this class it is desirable for the student to experiment with the special commands associated with the specialized or color graphics board in the particular hardware environment.

In a two semester sequence of four credit courses, we suggest more in-depth coverage of the same material with a more detailed laboratory assignment to include a project and a final assignment. Student interest should channel the direction of the final project. The final assignment should be reasonably complex and include rotation, translation, and color coordination. Typical of such a project would be the sun, earth, and moon depicting an annual cycle of repetitive events calculated for any calendar year. The dark and brightness period of the earth and the moon should be coded by a coloring scheme. Projects of this type are handled by the motivated students easily and the marginal students find it hard to complete this final project.

IV. A PROPOSED GRADUATE CURRICULUM

By and large the ground rules for the design of the graduate curriculum needs to address a set of different objectives and constraints. Graduate students are generally full time employees and make it to the university once or twice a week. However these students, being far more motivated and mature, can grasp concepts and need the introduction of the newer material at a consistent pace. Generally the classes meet for about 2 1/2 hours a week and a project would a typical means to provide hands-on experience on graphic devices in the University. The entire class time is generally consumed in discussing theoretical and concepts of Computer Graphics.

For the universities that do not have an established graphics laboratory an optional project to replace the lowest grade in one of the two or three examinations is proposed. In our experience about 30 percent of students do submit meaningful projects and about 10 percent do these projects even though they would get an A grade even without the project. For universities contemplating establishing graduate and undergraduate laboratories the following guidelines are suggested. In the undergraduate education where the attendance and submissions of laboratories is an integrated activity in completing the requirements, about half to three quarters graphic terminals or stand alone microcomputer graphic systems are proposed for every student enrolled. Between the time to program and individual preference, the students generally complete the assignment on time and do find terminals to do specialized projects. For the graduate students the laboratory set up needs fewer terminals. Students generally have facility to access personal and/or company computer systems. All graduate students do not use the university facility at the same time. Still, in our experience providing one terminal for every 2 to 3 students appears to be a compromise between university capital expense and student needs for graphics hardware.

IV.1 Course Organization. This is again a very delicate task for the educator. For about 85 percent of the students, the graduate course is their first exposure to any computer graphics, thus placing the responsibility on the instructor to cover about one to two semesters of the undergraduate course material in the first 4-5 weeks or about 10 hours of instruction. However the students being motivated can conceive and retain the material quite effectively. The last 9-10 weeks needs special consideration. For this reason we propose two independent graduate courses in computer graphics. The first course should be designed towards the engineers, architects, and component designers so that CG can be incorporated in this work environment in an interactive Computer Aided Design sense. The second course should be designed towards the students interested in Image processing, computer vision and/or pattern recognition. Accordingly we discuss these course contents in the subsequent section.

IV.2 CAD/CAM Oriented Graphics Course

The interactive part of the computer aided design needs special emphasis. The role of conventional man/machine communication as it is enhanced by a wide variety of input devices needs about a two-three week (5-7 hours) of instruction and discussion. Next the role of the human being as the creative component of the feedback design loop need special emphasis. The most recent trends of using the AI concepts in recognizing the negative feed components in channeling and stabilizing the design loop to its optimal configuration need expanded discussion for about two weeks. A project from industry, from the instructor's background, interest, and/or experience should take about three weeks. The remaining class time is discussing and monitoring student project through the semester. Finally the discussion of student projects plays a dominant role. From our experience, students pick problem areas to discuss which are consistent with their background in context to finishing the project in time. For the other students, this area of instruction is best exchanged between the students. The educator learns from the students concerns other than what he has taught in the class.

IV.3 The Image Processign Oriented Graphics Course. This course may or may not be placed in sequence to the CAD/CAM oriented course discussed in section 4.2. Generally universities do not have a prerequisite to either of the two courses except that the students should have a graduate standing. Once more the educator needs to cover the undergraduate material in a hurry but in context to the image processing aspects. Typical consideration would be the impact of the effect of resolution, color processor, and graphics board. The effects of the hardware components is significant. New software issues also demand three to four weeks of discussion. Material from the most recent publication, statistical, and syntactic pattern recognition, edge relaxation technique, texture recognition, effects of reflectivity, refractivity and light source orientation are also significant topics recommended for discussion at the rate of one to two topics each class. The proficiency of the educator plays an important role in the progress of the class. An excellent source of material is the recent IEEE publication dedicated to CG and also Computer Graphics World published by Penwell Corporation.

V. DISCUSSION OF STUDENT PROJECTS

The extent of creativity and the amount of work in projects for the CG courses can vary by about two orders of magnitude. At the lowest level, we have students submitting one page higher level codes for their project. Such students generally end up with an F for the project, but they do take all the examinations and some have a chance of a low B in the course. At the other extreme we have students with particular interest who turn in very significant projects. Typical of this are reported here.

V.1 Three Undergraduate Projects. The first project discussed here has been completed by a computer science student who had an inherent interest in human anatomy. The work was completed in addition to a normal completion of the laboratory work and lasted about four weeks. Figures 1a through 1c depict the functioning of the heart as it undergoes one cardiovascular cycle. The displays are in color and can be synchronized to the real time heartbeat (stethoscopic input).

The second project completed by an engineering student who had an inherent interest in automobile safety. The display package predicts the limits of the safe speed of an automobile negotiating the curvature of the road under different conditions of wetness. Figures 2a, b and c indicate typical displays when the speed and the road curvature (steering wheel position) and condition (humidity meter reading) are supplied to the program. This work was also completed during the last four weeks of the semester.

The third project completed by a graphics designer incorporated the design and coding of an entire graphics editor. Icons could be defined and reconstructed to generate novel designs and patterns. This program generated hard copy units in color and offered extreme flexibility in generating a variety of designs. Figure 3 illustrates a design be generated in the CSC 470 course.

V.2 Two Graduate Projects. As discussed in section 4, graduate students work under different constraints. These projects become slightly more encompassing and are generally accomplished by the students at their own pace. The first project completed by an architecture student is capable of doing all modular office design work and placing office furniture at the appropriate locations in view of predefined human factors engineering rules. The second project completed by a computer science student interested in music is capable of providing very effective computer aided music instruction to students on an individual basis. It activates a DBMS to keep track of students musical strengths and weaknesses. It also provides for automatic testing, grading and display of individual and class performance. The possibilities of using such graphical computer assisted instruction techniques are immense.

The list of possible student projects is long and it is desirable for the students to develop the project based upon their interest and motivation. An undergraduate teacher developed a complete package for her geometry class, teaching them the principles and reasoning behind the theorems her students learn in class. An art student exhibited intense creativity in altering the design and coloring he initially put into his program by a light pen. We intend to present and discuss some of these projects at the conference. Another student provided a completely graphical computer assisted instructional environment for medical students learning human anatomy and so on.

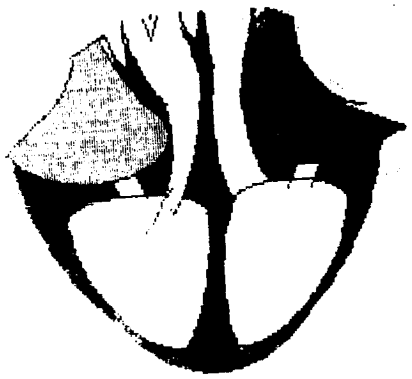


Fig. 1a. The deoxygenated blood comes from the body and fills the right atrium. Oxygenated blood from the lungs fills the left atrium.

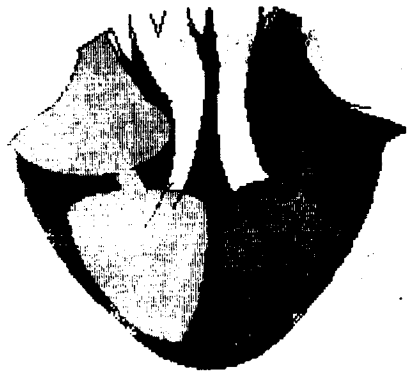


Fig. 1b. The blook is pumped into the ventricles through the open mitral and tricuspid valves.

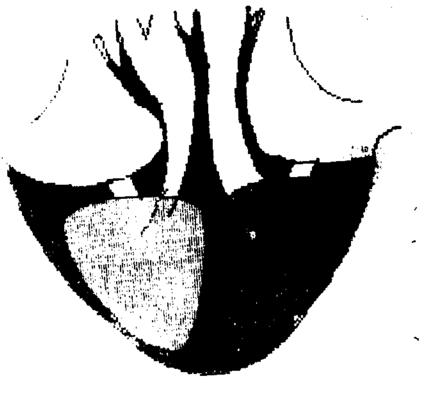


Fig. 1c. The blood fills the left and right ventricles.

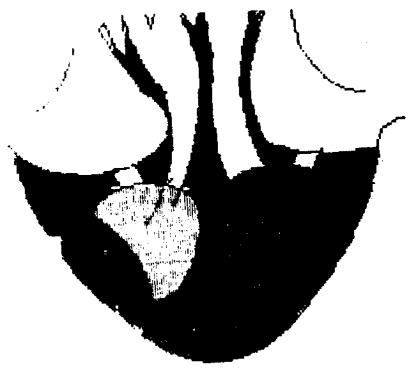


Fig. 1d. The simultaneous contraction of both ventricles results in cardiac systole.

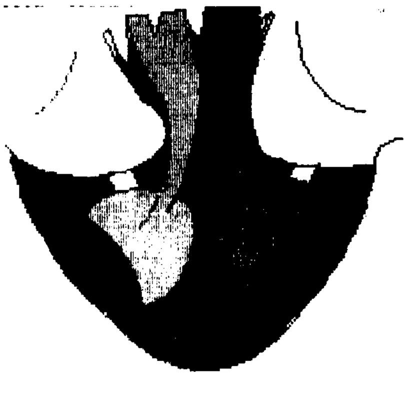


Fig. 1e. The blood is forced through the semilunar valves into the pulmonary artery and aorta.

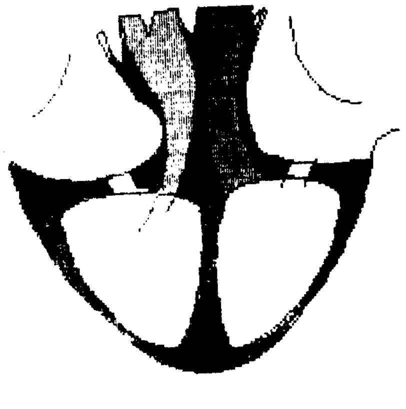


Fig. 1f. The ventricles release again and semilunar valves close. Blood flows into the atriums. The mitral and tricuspid valves open and the cycle begins again.

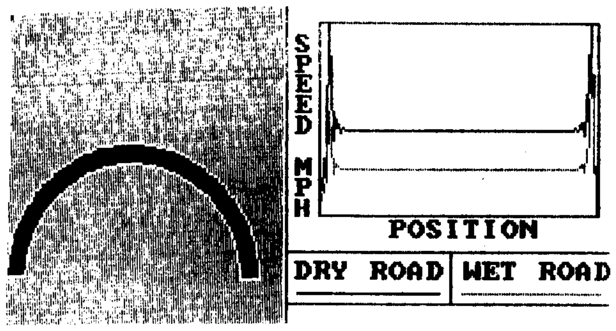


Fig. 2a. Uniform curvature and a safe speed at different road conditions.

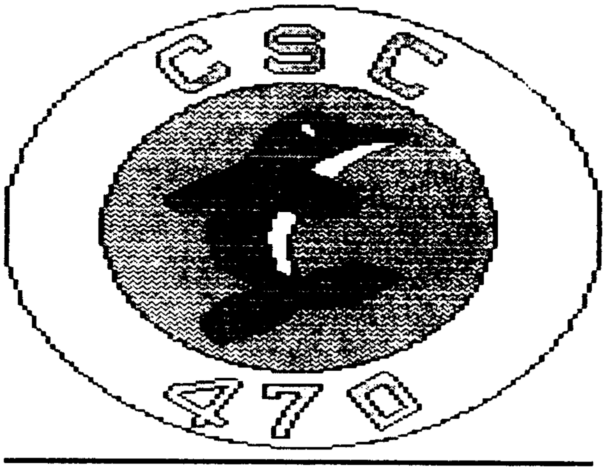


Fig. 3. A typical output of a graphics design editor.

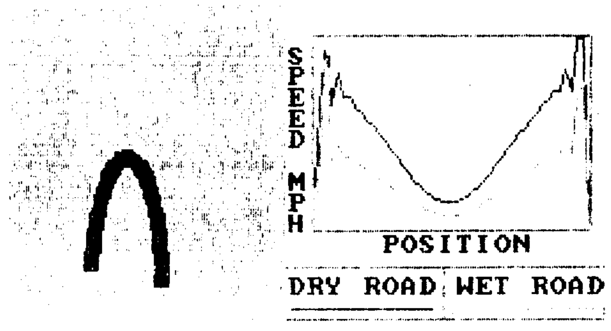


Fig. 2b. Variable curvature with the maximum at the center of the curve. The safe speed is depicted at the right.

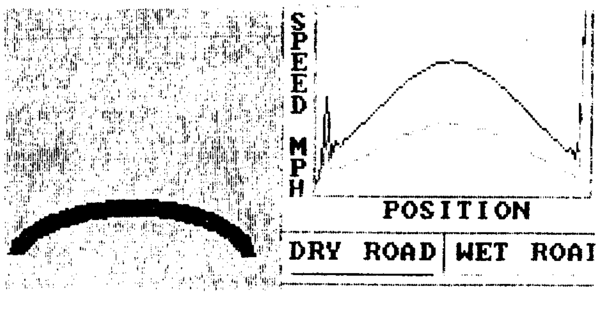


Fig. 2c. Variable curvature with minimum (or no curvature) at the center. The safe speed is depicted at the right.

VI. CONCLUSION

We have suggested and discussed topics for the one or two semesters of undergraduate graphics instructions. In the undergraduate instruction, the laboratory work is made an integral part of the instruction. A set of possible assignments are suggested in conjunction with the class discussion. We recommend that one semester of a four credit hour course as a minimum exposure to the undergraduate Computer Science students.

For the graduate instruction topics, research and assignments are recommended to a university offering mostly evening graduate instructions. Two courses not essentially in a sequence should suffice to cover topics of current interest and technical usage of computer graphics in an engineering or an image processing environment.

VII. ACKNOWLEDGEMENTS

I thank my students at the City University of New York and at the Polytechnic University of New York in Brooklyn where I have taught these courses in Computer Graphics at the Graduate and at the Undergraduate levels. The projects completed by some of my students are outstanding and I have learned the extent of commitment that these motivated students can show from my interaction with these students. It is evident from the projects they submit and the quality of reports they generate. In particular, the projects by Mr. Robert Begun (car skid), Mr. Curtis King (heart functioning), Mr. Yann Stanczewski (human anatomy), Mr. Ronald Gordon (A Music Training System) and Mr. Steven Robinson (Graphics Design Editor) need special mention. I wish them the best in their careers.

BALANCING PRACTICE AND THEORY IN TEACHING DATABASE: A PROJECT APPROACH

Barbara Beccue
Carol Chrisman

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

ABSTRACT

This paper will describe how a project is used in an undergraduate database course to illustrate a database development life cycle and give students an opportunity to apply the database concepts taught in the course. Working in pairs, students develop an application system which they implement using a micro database package. The project spans the entire development cycle from specification of requirements to implementation of the system.

INTRODUCTION

The Applied Computer Science Department at Illinois State University offers a program that attempts to balance theory with practice. It is felt that students need opportunities to apply the theory in realistic settings. Trying to provide realistic assignments presents a challenge in developing appropriate learning experiences which blend theory and practice. In programming courses this involves the use of assignments that represent typical business programs. In systems development courses, students experience the development life cycle through group projects involving an actual system being developed for real users. Developing suitable learning experiences in other courses has been more of a problem.

One course in which developing appropriate learning experiences has been a problem is the senior level database course. The main problem was the fact that providing the necessary hardware and software for practical experience would require a large amount of funding. The lack of any database management system that could be used in the course severly limited the choices of learning experiences. It is difficult for the students to develop an understanding of database technology and its impact on businesses from textbook exercises. Reading about database applications and actually experiencing the development of a database application result in different levels of understanding.

The resources available for use in the database course changed as a result of the department purchasing micro-computers and appropriate micro database packages. The database course was modified to include a semester long project during which the students develop a database application. The project spans the entire development cycle from specification of requirements to design of the database and implementation of the system. Each two person team selects their own application and the micro database package to use for the project. The type of database application system that the students develop is a customized application system that allows users, who are not trained in database, to interact with the database. In developing customized micro database applications, students write procedural code similar to writing application programs for mainframe database systems. This type of application system was desired so that the course would be relevant for either micro or mainframe databases. This paper will describe the project and how it is used to give students an opportunity to apply the theory taught in the course.

PROJECT DESCRIPTION

A typical student project might be a system for a local veterinary who runs a small clinic and boarding service.

The system would automate the record keeping for the animals treated at this clinic, maintaining a history of all visits and treatments for each animal. Reminders need to be sent to pet owners for each pet's annual checkup. The system would also maintain the record keeping for the boarding service, scheduling facilities to be used, and maintaining records about the animals being boarded. Customer bills itemizing treatment and boarding services and payments received are generated once a month.

Even though students work with different application systems, each team follows the same five part procedure for the project. The first part is selecting an application system and documenting a partial specification of this system. The second part completes the specification of the system to be developed by creating a conceptual model for the database. The third part, the design phase for the system, determines how the system will be implemented with a particular database management package. The fourth part is the actual implementation of the system. The fifth part is a summary and evaluation of the students' experience with the project and the particular database management system used. In addition to being a part of the total project, each part is treated as a separate assignment in that it is assigned, turned in, graded and returned. Problems with earlier parts must be corrected and resubmitted each time. The final project notebook contains cumulative documentation from all the parts of the project. The project accounts for 30% of a student's grade in the course.

The first part of the project has 2 steps. Step 1, selecting an application system, is often difficult for the students because they do not understand databases well enough to feel confident about selecting a suitable system. All students have prior experience with a development life cycle and group projects from a prerequisite systems development course, so they are familiar with projects of this type but not ones using databases. Rather than being overly concerned with selecting a database application, the students should consider application areas and businesses with which they are familiar. It is important that the students be able to make decisions about the system from a typical user's viewpoint. If the student can visualize using the system and understands the needs of the users, the student will be able to make knowledgeable decisions in designing the database. An application area that is somewhat complex usually works best. For example, if the system needs to keep track of 3 or 4 different objects (customers, products, sales, etc.), it will generally make a suitable class project.

The second step of this part begins the specification of the requirements for the selected application system. The documentation for this step includes an overview of the system and its users, specification of the business processing to be supported, and the types of queries to be supported. The scope and purpose of the application system, the users, and what the system will do for the users should be clear from the overview narrative. All processing needed to manage the data in the database must be identified. Any frequent processing needed for the system must be specified, whether it is updating the data or making inquiries. Students document the business processing by drawing a data flow diagram and writing a description of each process on the diagram. Students also list types of queries that the system will need to support.

Part 2 of the project creates a conceptual model of the data needed in the database. The data modelling technique taught in the course is the Entity Relationship approach. This gives the students a chance to develop an ER model for a real system. Students turn in an ER diagram and list of attributes for each entity and relationship. This completes the specification of the requirements for the team's application system.

The third part of the project is to design the database application system. The micro database packages which the students can use for the project are all relational and allow saved procedural code. This part includes two main tasks, designing a relational model and packaging the system for implementation in a way that is appropriate for the system's users. The relational model being designed should correspond directly to the ER model developed in part 2. The relational model should achieve an appropriate balance between normalized relations and retrieval efficiency. Packaging a system is organizing it into implementation units. Each unit usually becomes either a menu choice or a special command for that system and is implemented by a separate program (named block of procedural code). Programs would be provided for the standard processing described by the systems data flow diagram. In particular, any complex transaction that invloves updating several tables should be handled by a program so that it is hard for the user to create inconsistencies in the database. For ease of use, queries that would be used frequently should also be supported by programs.

The fourth part of the project is the implementation of the application system. Currently, students use one of three available packages. In this part, the needed tables are defined and sample data loaded. The procedural code decided upon in part 3 is written and tested. Each team demonstrates their working system to the class.

The fifth part requires each team to write a report which summarizes their experience with the project and evaluates the micro database package used for the implementation. Issues addressed in this report include strengths and weaknesses in the database management system. Specifically, the ease with which the system was implemented is evaluated and any limitations encountered are described.

CONCLUSION

The project which has been described has been used in this senior level database course for two years. The project has been well received and seems to have significantly improved the quality of the course. Students seem to be enthusiastic about their projects. Course evaluations have indicated that students feel the project has helped them to understand what a database is and how it can be used.

Having students develop a database application helps the students learn the theory taught in the course. One way the project helps is by creating an environment in which to teach database concepts. The different systems being developed within the class provide examples that can be used to illustrate the theory. The students' involvement with these systems make the examples more meaningful than textbook examples. The project gives students an opportunity to apply the database concepts in a real situation. This practical experience develops a depth of understanding of the use of a database management system that would not be gained without actual experience.

One goal of the course is for students to gain knowledge that can be applied in their future jobs. Many students will be employed by large companies which use mainframe commercial database management systems. Techniques used in the project can be used with either micro or mainframe database systems. For example, having students write procedural code gives them an experience similar to writing application programs for a mainframe database system. Since the project is done with micro database systems, it probably will not address some of the concerns of mainframe database systems. For example, a micro system may not allow multiple users accessing the database and may not provide security and recovery facilities.

The project is also useful to students in that it illustrates a methodology for database development. The project uses similar development phases to those taught in an earlier systems development course. Students experience this database development life cycle as they carryout the project over the course of the semester. This allows them to see how the development methodology is adapted to fit a database environment.

The use of the project has benefits for the teachers as well as for the students. The application systems provide a variety of realistic examples for class discussions. This type of project minimizes some of the problems generally associated with actual team projects. Having students provide the information that would normally come from the system users eliminates some of the problems and shortens the time needed to do the project. The small size of the teams reduces the project management difficulties, yet provides for an exchange of ideas.

The experience with this project has indicated that it is a satisfactory learning experience for the database course. Implementing the project on micro-computer systems is a relatively inexpensive way to provide hands-on experience. The project provides practical experience in applying the concepts taught in the course. This provides realistic assignments through which theory and practice can be balanced.

"Innovative Teaching Methods in the CAD/CAM Field"

by
Abhay V. Trivedi
Southern Illinois University

The Technology

Increased competition from foreign countries in every major industry has led the manufacturers and entrepreneurs in the United States in search of new and better methods for increasing the overall productivity of industry today. No other technology has had an impact on manufacturing as CAD/CAM. The term CAD, which refers to Computer Aided Design, and CAM, which refers to Computer Aided Manufacturing, together represent the integration of CAD and CAM through a common data base structure. The technology CAD/CAM involves the use of advanced computer hardware like high resolution graphics terminal, light per, digitizing pad, electrostatic plotters, along with a mini-computer or micro-computer to support the hardware tools. Also, the technology involves manufacturing tools like Numerical Control (NC), milling and drilling machines, sophisticated Machine Control Unit (MCU), as well as other sophisticated systems like Machine Vision and so on. Software for the CAD/CAM is equally advanced and expensive and capable of going beyond the graphic applications of CAD into Computer Aided Process Planning and other non-graphic applications. The technology exists. How do we attempt to make it accessible to the future users of tomorrow?

The Demand

Just as the demand for computer scientists is on the rise, so is the demand for CAD/CAM technologists. Professionals with the CAD/CAM expertise are amongst the highest paid in the nation. As the demand for better products and better designs increases, so does the demand for CAD/CAM technologists. CAD/CAM technology can be considered in a state where it has its own identity. It is this recognition which has created a tremendous interest in this future's technology being carried through the 21st century. Along with demand, CAD/CAM has brought increased competition amongst job seekers in the designing and manufacturing industry. To survive the competition, there is an important factor the job seeker has to understand, that is, to demonstrate higher skills in the technology, to demonstrate his/her "hands on" experience gained on the system and more importantly, to bring out his/her educational experience gained through attending a college or a university. Knowledge of only the demand factor can often be misleading if not accompanied with the knowledge of supply. The survival of the fittest law would still hold true eventually, which in turn gives rise to another concept--the art of acquiring knowledge, knowledge which is complete and thorough and meaningful.

Impact on Education

Any new development is the result of consistency and persistance of many brains working together. Any new idea is the result of understanding the fundamentals and building on it. It is this phase of "building the fundamentals" that a college or university education comes into play. A thorough understanding of the "fundamentals" would definitely provide better researchers for tomorrow. There is, however, a slight problem. The advancement of technology is so rapid today that unless the educational system is dynamic, it will be left far behind from reality. It is this factor of keeping pace with the technology that innovative teaching methods have surfaced more so in the last five years than ever before. The use of multi-media is not a need but a necessity, particularly in the developing areas like CAD/CAM, Robotics, and Automation in general. The current trend in industry is towards hiring people with at least some "hands on" experience on the system. A look at the CAD/CAM technology would suggest the importance of "computer" as evident by the name "Computer Aided". However, the knowledge of computers is extremely difficult to share through conventional methods like textbook reading. An integral part of CAD/CAM would be the use of a high technology laboratory, perhaps at a substantial cost, for understanding the technology in full.

The Teaching Strategy

The best way to explain the teaching strategy would be to break down the subject into its major components and look at each phase closely, so as to apply specific techniques to each individual phase. In the case of CAD/CAM, the approach would be to break the subject matter in three distinct phases:

Phase I	:	Understanding the fundamentals
Phase II	:	Application of Theory to laboratory experiments
Phase III	:	Understanding the transition from laboratory to industry

The textbook approach would be appropriate for Phase I as it deals with the explanation of the theory on which CAD/CAM is based. A simple interactive program can be written for students to get additional information on the subject not fully available in a single textbook. The interactive program would consist of a data base in which information from different sources is stored and constantly upgraded. This form of Computer Aided Instruction would be an ideal supplement to conventional classroom teaching. Another feature that can be added to such an instruction system would be a question and answer section which would be also responsible for student evaluation in this particular area.

Setting up the laboratory for implementing the CAD/CAM technology is the most time-consuming and expensive task. Typically, a stand alone CAD workstation would require:

(i) A graphics terminal

(ii) Operator input devices (cursor control devices, digitizers, Alphanumeric and other keyboard terminals)

(iii) A plotter and other output devices

(iv) Central Processing Unit

(v) Secondary storage

Also, CAM would consist of one or more numerically-controlled machines consisting of three basic components:

(i) Program of instructions

(ii) Controller Unit, also called a Machine Control Unit

(iii) Machine tool or other controlled process

Besides the hardware components of CAD and CAM, a respective amount of money has to be spent on CAD/CAM software. Phase II thus takes a student as close to an industrial environment as possible. It is here that a student gets a "hands on" experience on the equipment, understands the basic problems in equipment operation, gets a first hand look at "increasing the productivity" factor, as also the inherent differences in theory and practice. The emphasis in Phase II is laid not so much on how to replicate an industry, but to bring out the solution strategies that can be adapted later on in solving real world problems in the industry. An important philosophy that should be understood here is that, we are simulating an industrial operation and not only observing it, but trying to find an optimal solution for the problem (if any exists).

Phase II which tries to explain the transition from a laboratory environment to an industrial environment is perhaps less difficult than most people foresee. If Phases I and II are accomplished successfully, Phase III would automatically fall in place. A common approach commonly used by many colleges and universities today is the use of video tapes depicting the actual CAD/CAM operations in industry. Generally, the video tapes are inexpensive and cover a broad range of subjects beneficial to the user. Also, since the video tapes are updated constantly upon improvisations in the industry, they are a reliable source of information on current trends in industry. Usually, these video tapes are produced by a professional society and maintain a high standard code. Supplement to the use of video tapes and films is visiting ar industry implementing a CAD/CAM system. Field trips usually enhance a student's knowledge significantly as it is the first step towards his final destination. However, it is often not possible for a college or university to conduct such trips due to their isolated location from industries or sometimes also due to the time constraint.

Conclusion

The future's technology is just the beginning. The awareness to change is a good approach; however, the tools of knowledge enhancement are still not sophisticated enough for a common person to become failure free in the struggle. We are approaching more towards the common goal of increased productivity, better products, better quality, rather than looking at any isolated segment of a technology. The improvement of the overall quality of education is based on the innovative ideas and suggestions brought forth today. The change in the teaching trend should be directly proportional to the change in technology. Improvisations in the techniques have to be instant or would result in a vacuum composed of confusion and failure. Although the specific case discussed here is the CAD/CAM technology, the principles pertaining to implementing a sophisticated solution methodology remain the same and can be applied to any developing or already developed technology in general.

A PHASED APPROACH TO THE CASE STUDY IN SYSTEMS ANALYSIS AND DESIGN COURSES

by

LaVon Green

Assistant Professor

Purdue University Calumet

Information Systems and Computer Programming Department

Hammond, Indiana 46323-2094

(219) 844-0520 X498

ABSTRACT

This paper presents a technique for using the case study in CIS systems analysis and design courses. It has been applied in introductory courses, structured techniques courses, and service courses for non-CIS majors. The case study is used to simulate the phases of the system development process. The paper discusses how to prepare the students for the case study assignments, how to critique the assignments, and how to grade the final case study project notebook that is the cumulative documentation of all the assignments.

DEFINITION OF A CASE STUDY

A case study is a simulation of a real world situation. A problem is posed for the purpose of considering alternative solutions to the problem related to whatever topic is being studied. Case studies are often used for discussion purposes only. The case study technique that I am presenting is a practitional one that spans an entire course of study - a semester, trimester, or quarter.

In this application the case study is used as a simulation of the real world resolution of a business problem using a computer information system. In their book, Learning System Design, Davis, Alexander and Yelon, indicate that simulation is a better technique than the programmed approach for teaching problem solving techniques.(1) In the programmed approach, each step of a task is monitored, with the feedback of success required before the individual can proceed to the next step. On the other hand, simulation is not as effective as on-the-job training where problem-solving techniques are used in the real world and evaluated by others. For all practical purposes, on-the-job training would be impossible to arrange and control when teaching 75 students in a semester in the system development process.

THE PHASED SYSTEM DEVELOPMENT PROCESS

In the past several years, all texts that I have selected for systems analysis and design courses present a phased system development process. The reason that many textbooks present the phased approach to systems development is that most commercial organizations are recognizing the merits of the approach. The phased approach provides:

(1) a means for identifying and organizing all the tasks to be completed for the development of a computer information system into a relative sequence

(2) a framework for defining the deliverables to be produced by the project team to be reviewed at strategic points by the user and management

(3) a basis for estimating and controlling the system development process by dividing the process into smaller segments

It is this phased system development process encompassing all the systems analysis and design techniques to which I have applied the case study technique.

HOW TO APPLY THE CASE STUDY TO CIS SYSTEMS ANALYSIS AND DESIGN

As I indicated, the case study technique that I recommend is designed to simulate a real world systems development effort to solve a business problem. The student takes the situation presented in the study through all analysis and design phases of the systems development process. Most textbooks present a sample case study that can be followed throughout all the phases. However, the assignments are programmed and usually do not allow the student much freedom in selecting alternative solutions to the problem. Also most textbooks present case study situations that address one aspect of the system development process. While these case studies help the students understand a specific technique in systems development, they do not help the student understand the entire process of systems development. They do not make the student aware of the impact that decisions made during analysis phases will have on the design and eventual implementation of a computer information system. My proposed case study technique does.

The case studies I have used come from several sources:

(1) Some textbooks provide a stand-alone case study as an appendix intended for independent student development

(2) Case study manuals such as Mason Oaks by Eliason are available. However, these tend to be programmed leading to a single "right" solution

(3) Harvard Business School Case Services offers many computer-related case studies. While many of these are limited to a specific aspect of systems development, some, such as "Emery Worldwide," are excellent for adaptation to the systems analysis and design phases. "Emery Worldwide" works especially well with structured analysis and design techniques.

(4) I have written case studies based on my experiences as a consultant in the field of data processing

(5) Students select a case study from their experience using a business problem that has already been solved or one that is in the process of being solved using automation. In either situation, the techniques learned in class will be applied to the real world situation.

Because the case study technique is used to simulate the phased system development approach of the real world, the technique I use is also a phased approach. The students are assigned deliverables that correspond to the tasks for the phases presented in the text. The case study assignments in Figure 1 correspond to the seven-phase system development process covered by Senn in Analysis and Design of Information Systems.(2) Students work independently in the initial systems analysis and design course in order to practice all new techniques. Students work in teams in the advanced structured analysis and design course in order to experience the effect of structuring the work effort.

While I am discussing the initial phase of the system development methodology, the students are reading the case study or investigating their work environment for a suitable case study. Purdue University Calumet is a commuter campus with 62% of the students attending on a part-time basis. Therefore, many students are employed in the data processing area or in a business situation where suitable case study material is available. I also discuss the case study in class when prompted by students' questions and schedule private sessions for students developing their own case studies. Of course the discussion continues throughout the course. Because interviewing is the primary data collection technique used in the initial analysis phases, I set up some interviews in role-play situations to give the students practice in this difficult technique and to give them some more details relating to the business situation. Sometimes I am the interviewee and the class as a group asks the questions as the analyst. Sometimes I have the students participate in a one-on-

PRELIMINARY INVESTIGATION PHASE

- Draw an organization chart
- Write a memo to schedule an interview with a user
- Conduct the interview (one-on-one in-class role playing)
- Write a memo summarizing the interview
- Write the Feasibility Report
  - Summary of recommendations
  - Statement of problem
  - Details of problem
  - Potential benefits
    - Increased revenue
    - Decreased expenses
    - Intangible benefits
  - Feasibility: operational, technical, financial
  - Recommendations
  - Estimates and Gantt Chart for the next phase

DETERMINATION OF REQUIREMENTS PHASE

- Draw the data flow diagram for the current system
- Draw the system flowchart of the transaction processing subsystems
  - File maintenance subsystem
  - Business processing subsystem
- Draw a system flowchart of the management information subsystem
- Write the system proposal
  - Cover memo
  - Summary of recommendations
  - Overview of systems study
  - Detailed findings/requirements
  - Alternative solutions
  - Cost/benefit analysis
  - Recommendations
  - Estimates and Gantt Chart for the next phase
- Conduct the walkthrough of the proposed system flowchart

DEVELOPMENT OF PROTOTYPE SYSTEM PHASE

- Prepare a sample screen or report for user

DESIGN OF SYSTEM PHASE

- List all input transactions
- List all output screens and/or reports
- List all files: master, transaction, table, backup, history
- Design one output screen or report
- Design one input transaction form or screen
- Write the input procedure for the transaction
- Design the corresponding input transaction file
- Design one master file
- Design one table file
- Write one program specification
  - Purpose
  - Inputs
  - Outputs
  - Processes using decision tables, decision trees, Structured English, Nassi-Schneiderman Charts, or narrative
  - Hierarchy chart
- Write the control procedure for program

DEVELOPMENT OF SYSTEM PHASE

- No assignment

SYSTEM TESTING PHASE

- No assignment

IMPLEMENTATION PHASE

- Write an agenda for the post-implementation review meeting

Figure 1

one role-play situation with the interviewee given a script on how to play the role and with the individual playing the analyst asking the questions.

The students work on each phase assignment after we have studied the purpose and the techniques required to complete the tasks within the phase. The deliverables of the phase included as part of the assignment represent a sample of what would be produced in a real world situation. Each assignment is inserted in the project notebook which is the cumulative documentation of the system development process. Each assignment is submitted to me as though I were the manager of the project. I record the date of the original submission of the assignment to track the timeliness relative to assignment due dates. These due dates simulate project target dates. I critique each assignment, as would a project manager, based on the information provided and the assumptions made by the analyst and based on previous decisions made for the project. There are many "right" solutions to the case study. If an assignment needs correction, the student can resubmit the assignment as often as necessary. As in the real world which we are simulating with the case study, the systems analyst cannot proceed with subsequent phases of systems development until the current phase is approved by the users and management. The student assumes the role of the systems analyst throughout the course.

The entire project notebook is submitted at the end of the semester for grading. No grading is done until this time. It is the finished product and the record of timeliness that determines the grade for the case study. Following is the point system I use when evaluating the case studies:

Completion of all assignments	25%
Timeliness	10%
System flowchart	20%
Walkthrough presentation	5%
Quality of format	20%
Content	20%
	100%

STUDENT APPRAISAL OF THE CASE STUDY METHOD

Student response in the university-administered student evaluations have improved toward the case study project as my method changed from a semester-end assignment to the phased approach I now use. Many students have come to me saying that the major problem with the case study was that they could not spend as much time on it as they would like because of other demands on their time.

A recent study conducted by Bruce Joyce and Beverly Showers at the University of Washington also supports my approach.(3) The purpose of the study was to determine the ability of an adult student to apply a learned technique on the job. When presentation techniques alone were used, 5% of the adults could apply what they learned. When demonstration or guided practice (like a case study) were used, 10-15% could apply what they learned. This is not an overwhelming percentage of success, but it is two to three time better than with presentation alone. When on-the-job effort was combined with coaching, 90-95% could effectively apply the techniques. My students practicing the techniques with work-related case studies resemble the latter group.

My experience and research findings as noted encourage me to continue to improve the case study methods I have practiced. I plan to enhance future systems analysis and design courses with hands-on use of automated analysis and design tools. The automated approach to systems analysis and design coupled with the phased approach will put our students in step with the leaders in systems development techniques.

BIBLIOGRAPHY

1 Learning System Design: An Approach to the Improvement of Instruction, Robert H. Davis, Lawrence T. Alexander, Stephen L. Yelon. (McGraw-Hill Book Company, 1974).

2 Analysis and Design of Information Systems, James A. Senn. (McGraw-Hill Book Company, 1984).

3 "Adult Learning: More Than Information," Jim Bellanca. (Illinois Renewal Institute, Arlington Heights, IL, 1986).

AN INTEGRATED AIS-MIS COURSE

Angelo E. DiAntonio
Towson State University

Arthur B. Kahn
University of Baltimore

ABSTRACT

An experimental interdisciplinary course to interface Accounting Information Systems and Management Information Systems is described. The class included twelve graduate MIS students, an Accounting professor and a Management Information System professor.

Novel features included the design and implementation of an accounting system on a microcomputer using an electronic worksheet. An atmosphere of open discussion was established when the two professors challenged each other, and then invited the students to join in.

The overall experience was informative, instructive, and beneficial for all of the participants. Activities such as this must be encouraged and welcomed by the university community.

INTRODUCTION

During Summer 1985 an experimental interdisciplinary course was taught by a team including an Accounting professor and a Management Information Systems professor at the University of Baltimore. Intuitively, we felt the need for interaction between our disciplines. Each of us had something unique for the students and we recognized an opportunity to more intimately explore the interface between our disciplines. We also anticipated that an integrated experience would promote a synergistic interaction.

The course involved twelve students pursuing a graduate MIS program. Before enrolling, students were informed that this was an experimental interdisciplinary course and were given an overview of the course. Half of the students had undergraduate degrees or work experience in Accounting. The other half had MIS backgrounds with minimal formal background in Accounting and little interest in the pursuit of Accounting knowledge or training. They were more interested in examining applications of COBOL as well as hands-on computer experience.

COURSE ORGANIZATION

The syllabus (Fig 1) included readings from the Moscove-Simkin text, ACCOUNTING INFORMATION SYSTEMS, 1984, John Wiley. This book has a reasonable blend of Accounting Information Systems as well as Management Information Systems. It provided a solid basis for reading assignments and lectures. In our search for a text we found that the terms Accounting Information Systems and Management Information Systems do not have any generally accepted definitions. Thus, the first result from our combined effort was the personal recognition of the state of the art. The students were made aware of this.

We also used CASEBOOK IN ACCOUNTING INFORMATION SYSTEMS by Romney, Cherrington and Hansen, 1985 John Wiley & Sons, Inc. Each student was required to examine one case and use it as the basis for an oral presentation to the class. The textbook, periodicals, fellow class members, and the instructors served as resources for this activity.

A case study project required the students to write a critique of topics discussed in class and in the textbook using the style of the ACCOUNTING REVIEW, AMERICAN ACCOUNTING ASSOCIATION.

SPREADSHEET PROJECT

A spreadsheet implementation of a full accounting system was an early highlight of the course. Incidentally, the students learned spreadsheet and microcomputer skills. However, the main focus was to expose in clear yet dynamic terms the basic structure of an accounting system. The class was segmented into several teams, each assigned a specific, subsystem: Receivables, Payables, etc. There were detailed instructions with respect to constraints on the spreadsheet. Each team used a specific non-overlapping section of the spreadsheet. After each group completed their project, the templates were superimposed to form a primitive but complete accounting system.

CLASS ATMOSPHERE

The syllabus is shown in Figure 1. Class discussions and lectures closely followed selected chapters in the Moscove-Simkin textbook. Figure 2 displays the plan for the evaluation of students. It includes a final examination based on class activity, textbook, and lectures. The purposes and objectives of the evaluation were thoroughly discussed with the class during the initial class. Figure 3 lists the titles of cases that the students selected. The case discussions were parallel to the content material of the textbook and the class lectures.

Students were given ample opportunity to discuss topics under consideration without constraint. They were encouraged to give vent to their individual opinions without intimidation from the instructors or from members of the class. The discussions proved to be free, open, informative, and instructive. It was interesting to the instructors, as well as the students, to observe the manner in which topics were dissected, examined and developed to the mutual satisfaction of all the participants.

At the outset a problem arose because it was agreed that the Accounting professor would lead the lectures. The students with the MIS orientation were concerned that without a strong Accounting background they would be at a disadvantage. Fortunately, they discussed this concern with the MIS professor and the issue was resolved by assuring equal emphasis to MIS and Accounting.

It was recognized by all that Accounting terms and concepts required definition as a prerequisite to meaningful discussion. In this regard, all members of the group had responsibilities. The nonaccounting students were required to question accounting terms and concepts. In turn, the accountants were required to provide jargonless responses. The lack of accounting background in some was turned into a benefit when the nonaccountants learned to ask and the accountants learned how to answer.

DIFFERENCES IN RESPONSIBILITIES

As the course progressed, we recognized many differences within the faculty team and also among the students. The differences between the disciplines appeared to divide and create a hostile milleau. But probing consideration found a bonafide basis for the differences.

The objectives of MIS and accounting are different. The mission of MIS is to work within the organizational structure. The emphasis among accountants is external reporting. These differences cause a schism in the philosophical framework of the two disciplines. The primary MIS obligation is to the managers within the enterprise, while accountants are often responsible to stockholders, regulatory agencies, and creditors; all outsiders.

The obligation of MIS rests within the organization. There is a basic requirement to help decision-makers analyze, design and implement systems. There is a further commitment to become involved with the day-by-day activity of the organization. MIS system responsibilities are on-going and pervasive. The MIS group is concerned with the handling of individual transactions including design and flow of documents as well as the details of data entry. The selection and specification of hardware and software also fall within the province of MIS.

EXTERNAL vs INTERNAL AUDITORS

The extensive design efforts required for modern management information systems creates a dichotomy between external and internal auditors which was non-existent thirty years ago.

External auditors are responsible to such regulatory agencies as the Securities and Exchange Commission, the Federal Home Loan Bank Board and the Interstate Commerce Commission. Public accountants also are responsible to shareholders and the community at large. These groups singly and collectively depend upon the auditors to grant opinions as to the fairness of the financial statements at the end of each year. It is imperative that public accountants maintain their independence to support their unbiased stance. The auditors are bound by the "CANONS OF PROFESSIONAL ETHICS" promulgated by the American Institute of Certified Public Accountants.

Because of their exposure to civil liability and criminal penalties, the external accountants have an entirely different relationship with the company than the MIS group. Should the public accountants be found grossly negligent in the performance of their engagement, they may be subjected to penalties and damages which require extensive, expensive and exhaustive litigation. In the eighties, the malpractice insurance premiums of accounting practitioners have soared along with the more publicized medical malpractice premiums.

A high degree of credibility is essential to effective accounting practice. Any attack on credibility causes deterioration in public perception of auditors. Damages to their public image damages an accountant's ability to function. In a severe case, loss of credibility may lead to criminal prosecution and loss of license which would bar the individual from further practice of accounting. To date, there is no comparable licensure for MIS professionals.

Internal and external auditors have different roles. The external auditor is precluded from participating in the design of the system. But if the system is to be auditable, it is essential that auditors be involved with the design from inception. This dilemma may be resolved by having separate sets of auditors for the design and attest functions. Clients tend to engage accountants initially for the attest function. The accounting firms tend to use this as an entry to generate management advisory services which include system design. When a firm is performing both attest and design services for the same client, the firm must make every effort to avoid even the hint of a conflict of interest.

IMPACTS

Although we were aware of these differences before the course, it was only through discussions stimulated by the instructor team effort that we began to appreciate the divergence of our disciplines. Much of this discussion was during class and the students not only received the benefit of our conclusions but actively participated in deriving them. The nonaccounting students did gain an interest in the mechanics of Accounting. The Accounting people began to appreciate the viewpoint of nonaccountants. Both groups of students learned the necessity of gaining better communication skills. The dialogue between the two professors provided a model.

COST/BENEFIT

The cost/benefits of this course merit comment. Two professors for 12 graduate summer students just breaks even for us. We don't recommend this as a continuing practice. However, such a team effort is well justified occasionally from the viewpoints of course development, faculty development and as a springboard for research. The current paper is one of a series of research projects that was and will be generated by this effort.

Lecture 1	Introduction to Accounting Information Systems
Lecture 2	Accounting Information Systems and the Accountant
Lecture 3	Management Concepts
Lecture 4	Introduction of Computers
Lecture 5	Systems Analysis
Lecture 6	The Design Phase
Lecture 7	Computer Files and Databases for AIS
Lecture 8	Internal Controls Within AIS
Lecture 9	Controls for Computerized AIS
Lecture 10	Computer Crime
Lecture 11	Resource Acquisition
Lecture 12	Review, Findings, and Conclusions

Figure 1: SYLLABUS HIGHLIGHTS

Spreadsheet Project	15%
Project Presentation	15%
Paper	50%
Final Examination	20%
TOTAL	100%

Figure 2: EVALUATION OF STUDENTS

1	Systems Analysis and Document Flowcharting
2	Analysis of Existing System
3	General Systems Design
4	Input, Output, File, Software Evaluation
5	Detailed Systems Design-Output, Input and Files
6	Evaluation of Request for Proposal
7	Systems Configuration
8	Implementation-Task Planning

Figure 3: TITLES OF STUDENT CASE STUDIES

THE DESIGN AND USE OF A NASSI-SHNEIDERMAN PROGRAMMING TEMPLATE

Dr. Thomas W. Mason
Department of Computer and Information Systems
Florida A&M University
Tallahassee, FL 32307

Although there is evidence of adoption of Nassi-Shneiderman charting by industrial programmers, this technique is conspicuously absent in present-day programming logic textbooks. At Florida A&M University, Nassi-Shneiderman charts are taught in the Structured Design course. Their use has led to the development of a Structured Template which is a stylized version of a standard end-of-file processing method as diagrammed with N-S charts. The Structured Design course uses the Structured Template in the design of real-world data processing problems. These problems include the consideration of report titles, page headings, page summaries and report summaries. The consideration of these report enhancements led to the recognition of a program sub-structure heirarchy which is used to further reinforce the systematic approach to program design. Based on current experiences with this approach, a new course is now being prepared to integrate Pascal into the present material. Improvements in student programming ability have been observed but have not been formally measured.

INTRODUCTION

In 1966, Bohm and Jacopini's landmark paper (1) introduced the concepts of structured programming. They showed that all programs can be written with three programming constructs -- sequence, selection and iteration. Further, each of these constructs has a single entry point and a single exit point. Thus, a structured computer program is one in which the flow of control is straight-line from the first structured construct to the last.

In 1973, Nassi and Shneiderman (2) developed a method of flowcharting that graphically complemented the concepts of structured programming. Prior to their development, structured programs were displayed with traditional flowcharts. What was needed, however, was a design tool that only allowed structured programming constructs. Nassi-Shneiderman charts (see Figure 1) were the answer to this problem.

Nassi-Shneiderman charts are a sequence of rectangles. Each rectangle is of only three types -- sequence, selection and repetition. Further each rectangle has a single entry point and a single exit point. Thus, Nassi-Shneiderman charts preserve the Bohm-Jacopini requirements for structured programming without allowing the entry of non-structured elements.

The acceptance of Nassi-Shneiderman charts for information system development is growing but the evidence is largely anecdotal. Yoder and Schrag (3) reported on the experiences of programmers at the IBM Systems Product Division as of 1978. They cited advantages of Nassi-Shneiderman charts over HIPO charts, flowcharts, psuedocode and prose in the areas of program design, programming, peer review and program documentation. There has been little subsequent documentation of industrial usage of N-S charts in the literature.

If we consider textbooks to be representative of the practices used in programming education, then Nassi-Shneiderman charts have not been widely adopted in academic circles. Saret (4) mentions Nassi-Shneiderman charts but they are not used to chart algorithms. Motil (5) uses N-S charts but calls them "flow blocks." Schneyer (6) currently has the only programming logic text on the market using Nassi-Shneiderman charts. He specifically adopted them because he felt they were "the best visual aids to the construction of structured algorithms yet devised at the module level" (7).

DEVELOPMENT OF THE STRUCTURED TEMPLATE

In the past, our programming logic course (entitled Structured Design) used flowcharts and psuedocode, but we have recently re-oriented this course to use Nassi-Shneiderman charts as the preferred design tool. Further, it was noticed that the bulk of the problems used in that course were batch processing problems using end-of-file processing logic. In particular, the end-of-file processing logic that we adopted treats the end-of-file record as a special form of sentinel record.

Thus, in N-S chart form, the basic processing logic is:

```
Initialize all relevant variables
Read the fields in the first record
WHILE (not the end-of-file record)
    Process the data in those fields
    Read the fields in the next record
Calculate and write all summary values
```

From this basic logic, a template was designed to be used for all end-of-file processing problems. The purpose of the template was to focus the student's attention on those programming aspects which changed from problem to problem, while reinforcing the constancy of those program control aspects which are unchanging from one problem to the next. That template is shown in Figure 2. The student's Nassi-Shneiderman diagrams are written in the sections labeled INITIALIZATION, PROCESS RECORD and SUMMARY.

The Structured Template reads like a standard Nassi-Shneiderman diagram although stylistic changes have been made. The first section is the INITIALIZATION section of the program. This is followed by reading the first record and testing for the end-of-file record. The end-of-file record test was made explicit because of the lack of uniformity of handling eof-tests among programming languages or vendor machines. It was felt that this explicit test could be translated into any programming language.

The PROCESS-RECORD section is repeatedly executed as long as the variable "done-processing" has a value of "no." This, of course, is the "work-horse" section of the program design. The importance of this section and its repetitive nature are emphasized by the dark background that is used. This is a slight stylistic departure from standard Nassi-Shneiderman charting practices, but it seems to work.

The SUMMARY section is entered once the PROCESS-RECORD section is exited (when "done-processing" has a value of "yes"). In this section the student draws the Nassi-Shneiderman diagram for the summary actions.

USE OF THE STRUCTURED TEMPLATE

In using the Structured Template, the stress is constantly on program design. Students are introduced to a systematic method of determining the design elements to be drawn in the Template. The Structured Template does reduce program design to a fill-in-the-blanks approach but that merely frees the student to concentrate on the important aspects of the program. The keys to success in using the Structured Template are to understand the problem, understand the method of solution and to systematically design the computer program.

USE OF THE STRUCTURED TEMPLATE IN THE STRUCTURED DESIGN COURSE

In the Structured Design course, the Structured Template is used first for "warm-up" problem activities. These are basic processing activities but they are formulated outside of the typical data processing problem statements. We use a file whose values are playing cards and problems are posed which lead the student to consider the activities of:

a. selecting cards of different types
b. counting
c. accumulating
d. averaging
e. using flags to control processing
f. finding the smallest or largest value in a file
g. recognizing a sequence of cards (all cards in the sequence have the same suit)

Next, students are introduced to problems that are more representative of the "real-world." These problems use the same processing activities that were stressed in the "warm-up" problems, however, the problem assignments are made more complex by including requirements for:

report title
page heading (with page number)
page summary
report summary

Each of these report enhancements is examined and is shown to have a generic program structure when represented on the Template. Further, it is shown that page heading structures always precede record-processing structures. Record-processing always precedes page summary structures. And, finally, in the summary section, the page summary logic for the last page will always precede the report summary logic structure.

Thus, there is a heirarchy of program sub-structures that is inviolate. This heirarchy becomes an important teaching tool. Program design now begins with the step of identifying the component structures required by a given problem. Each component is then considered in turn as dictated by the heirarchy. Each component structure has a specific form which is entered into the Structured Template with the changes dictated by the particular problem.

Additional course topics include control break logic, arrays and simple data structures.

EXPERIENCES TO DATE WITH THE USE OF THE STRUCTURED TEMPLATE

The Structured Template has only been used one year. Further, of the two course instructors, only one used the Template exclusively. We have not formally tested its effectiveness. Student acceptance of Nassi-Shneiderman charting in general and the Structured Template in particular has been favorable and encouraging.

Student behavior seems to indicate that students would significantly benefit if a programming language were introduced into the course as program logic is being studied. Schneyer referred to this dilemma when he wrote: "Although my approach is language-independent...(it) is my firm belief that any course in programming should be accompanied by actual programming activity" (8). The faculty of the department have come to the same conclusion and starting in the Fall semester 1986, the structured programming logic course will be expanded to two semesters by integrating the equivalent of a semester of Pascal.

Even now, however, we can discern a marked improvement in the programming abilities of some students in COBOL, FORTRAN and Data Structures. Certainly we have observed a considerable increase in the rigor of the Structured Design course due to the use of the Structured Template. We expect the inclusion of Pascal to be the last needed ingredient to produce a superior sequence of introductory structured programming courses.

REFERENCES

(1) Bohm, C. and I. Jacopini, "Flow Diagrams, Turing Machines, and Languages with only Two Formation Rules," _Communications of the ACM_, vol. 9, no. 5, (May 1966), pp 366-371

(2) Nassi, I. and B. Shneiderman, "Flowchart Techniques for Structured Programming," _ACM SIGPLAN Notices_, vol. 8, no. 8, (August 1973), pp 12-26

(3) Yoder, C. M. and M. L. Schrag, "Nassi-Shneiderman Charts: An Alternative to Flowcharts for Design", _Proceedings of ACM SIGSOFT/SIGMETRICS Software Quality and Assurance Workshop_, (November 1978), pp 386-393

(4) Saret, L., _Data Processing Logic_, McGraw-Hill, New York, 1984

(5) Motil, J., _Programming Principles: An Introduction_, Allyn and Bacon, Inc., Boston, 1984

(6) Schneyer, R., _Modern Structured Programming: Program Logic, Style and Testing_, Mitchell Publishing, Inc., Santa Cruz, 1984

(7) Schneyer, R., p. xii

(8) Schneyer, R., p. xii

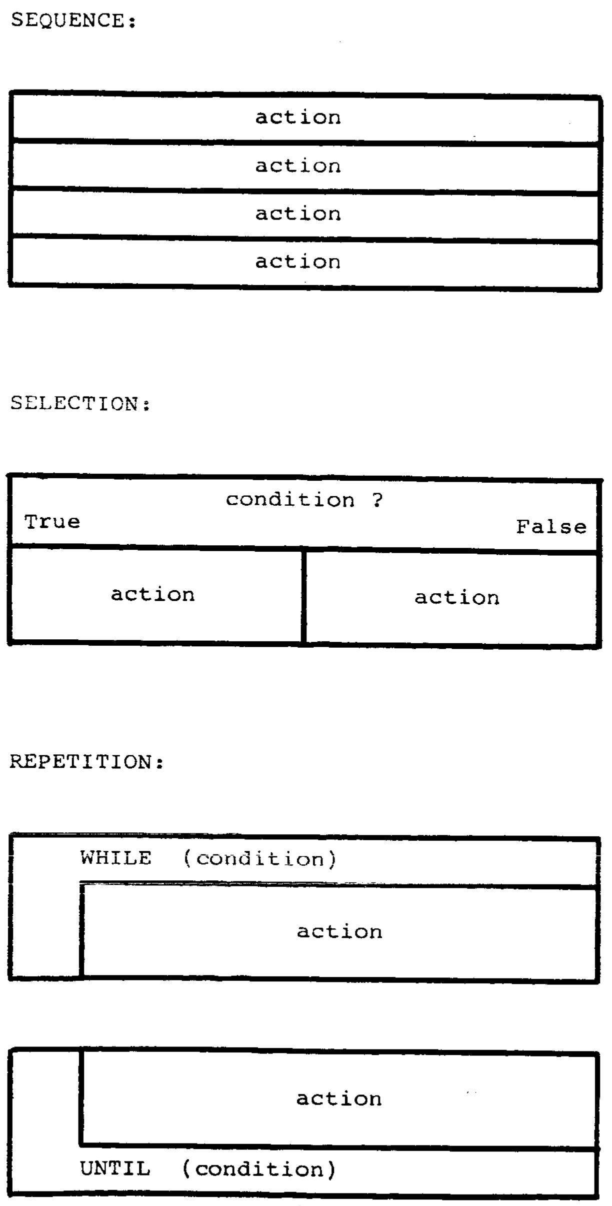


Figure 1. Design elements used in Nassi-Shneiderman charts.

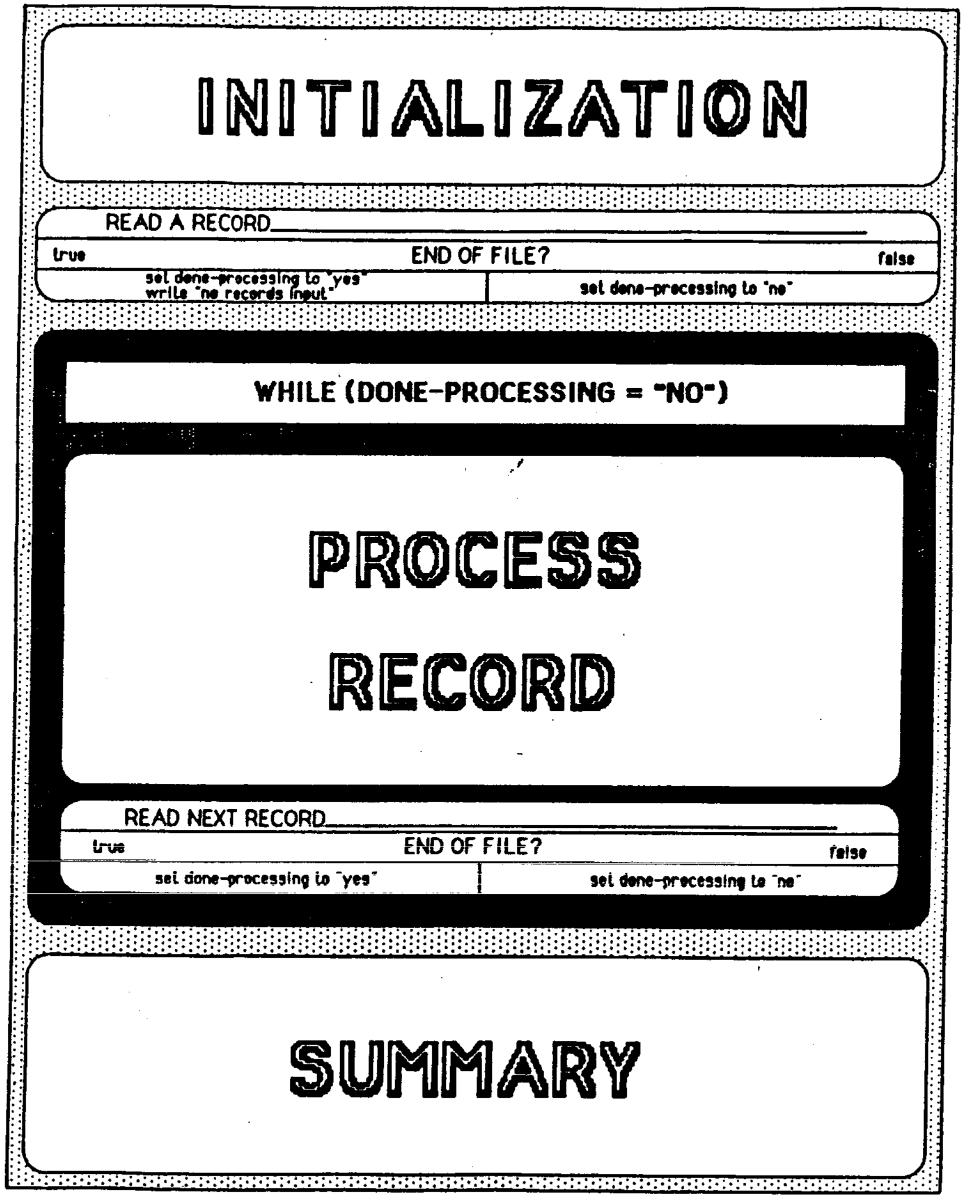

INITIALIZATION
READ A RECORD
true
END OF FILE?
false
set done-processing to "yes"
write "no records input"
set done-processing to "no"
WHILE (DONE-PROCESSING = "NO")
PROCESS RECORD
READ NEXT RECORD
true
END OF FILE?
false
set done-processing to "yes"
set done-processing to "no"
SUMMARY

Figure 2. The Structured Template

INFORMATION SYSTEMS TRAINING IN A LIVING LAB

A MODEL FOR DEVELOPING STUDENT GENERATED "LIVING" CASE STUDIES IN A SYSTEMS ANALYSIS CURRICULUM

ILENE SIEGEL DEUTSCH
SCHOOL OF COMPUTER SCIENCE AND INFORMATION SYSTEMS
PACE UNIVERSITY

This paper is intended to serve as a practical guide for the implementation of a Systems Analysis curriculum using "living" case studies, developed by the students as part of the course requirements.

INTRODUCTION

Pace University's approach to information systems integrates application and technology, "real" business professionals and students. Communication skills - both written and verbal - are an important cornerstone to this approach to the teaching process.

BACKGROUND

The New York City campus of Pace University is located in the financial and municipal district of Manhattan. A preponderance of the students are employed through Pace's extensive model cooperative education office. Information Systems is a strong,well developed, well enrolled major. The Systems Analysis course (Systems Concepts I - course number IS 206) is an upper level course for juniors and seniors. It is a required for all Information Systems majors, and is followed by another required course, Systems Design (Systems Concepts II - IS207.)

Although the New York City business community provides a rich resource of talented professionals who come into the classroom and share their expertise and experience with our students, we go beyond this. . . We see our own university administration as information systems consumers, we also explore the faculty resource with guest speaking engagements across our School of Computer Science and Information Systems, we go further by encouraging the use of "living" case studies from our students own work experience. This approach supports Pace's strong commitment to field placements, student internships, andcooperative education.

WHAT IS A "LIVING" CASE STUDY?

A living case study is a hands-on experience that a student has had in his/her own work environment. Instead of using two dimensional, professionally written case study material to illustrate systems analysis techniques, the student is asked to first write a case study about his/her actual job experience, vis-a-vis a specific problem in the organization. Here we are looking at the first step in the systems development cycle --problem identification.

The student has an opportunity to begin to develop and practice systems analysis skills as they are being learned. Using a live case study also gives the student an opportunity to call on knowledge from other disciplines while he is on his feet; he also has the opportunity to fall down in a safe, learning environment - the classroom.

IMPLEMENTING THE APPROACH

By the third week in the semester the students are grouped in self-select teams. The team members select one case to work through, as a team, during the semester. Therefore the team member whose case is chosen becomes the resident source of information. The team members review the existing system in detail. Sometimes they will make an on-site visit, invite players in the existing system to a team meeting, or design and administer a questionnaire in their effort to better understand the "living" case.

There are several presentations by each team to the class during the semester, paralleling the course work and reflecting the progressive phases in the systems development cycle. These presentations have a threefold purpose: student evaluation for grading, peer and instructor feedback, and development of public speaking skills. This process culminates in a major presentation by each team at the end of the semester - a simulation of an actual presentation to management.

Managing a Systems Analysis course as this model suggests helps better prepare the student for the dynamics of actual corporate life, provides for animated class discussions, and helps make teaching fun.

ADVANTAGES TO THE APPROACH

- Integrates academic knowledge with experiential process
- Helps students see application of systems analysis techniques within their own realm of life/work experience
- Adds excitement to a classroom
- Protects faculty from having to read the same systems analysis case study forever. . .

NECESSARY COMPONENTS TO THE APPROACH

- Students with work experience - in any size environment
- An experimental attitude toward the teaching of Systems Analysis
- A belief that interactive learning is more meaningful than static learning.

CONCLUSION

This approach - the use of student generated "live" case studies - to teaching Systems Analysis works well when a majority of the students have been or are currently employed. Using "living" case material adds depth and interesting complexities of dimension to both the learning and the teaching process. The richness of the students experience in the classroom in juxtaposition to their "living" cases illicits comments like these at semester's end:

> "My boss really wanted to know more about my ideas regarding our system."
>
> "We've changed things around the office since my team's analysis and recommendations."

Who could ask for anything more. . . integration of application and rhetoric.

REFERENCES

1. Eckols, Steve. How To Design and Develop Business Systems. Mike Murach & Assoc., Fresno, CA, 1983.

2. Goroff, Iza, "The Live Student Systems Development Project." DPMA, ISECON '85 Proceedings, pp. 20-23.

3. Lucas, Henry C. Jr., Toward Creative Systems Design. Columbia University Press, New York, NY, 1974.

4. Fricker, Al, Effective Information Management. University Press, Cambridge, England, 1982.

Ilene Siegel Deutsch
School of Computer Science and Information Systems
Pace University
One Pace Plaza
New York, NY 10038

(212) 488-1499

AN EXPERT SYSTEM FOR PROBLEM SOLVING AND DECISION MAKING IN A SYSTEMS ANALYSIS AND DESIGN COURSE

Marian Sackson
School of Computer Science and Information Systems
Pace University
Pleasantville, New York

Problem solving and decision making are some of the learning components of a systems analysis and design course. This paper describes an expert system developed to enhance the learning process for problem solving and decision making.

The learning of a structured system's life cycle, the steps of the systems analysis and design process, data flow diagrams, data definitions, and all the other aspects of an introductory course in systems are very important. But, one topic of any systems course should include problem solving and decision making. To fulfill this requisite, I have developed an expert system that can be used to test the students decision making skills.

Decision making is fundamental to human existence. We are continually faced with problem that need solutions. Decision making is a very complex activity. The process often poses alternative courses of action, each of which must be evaluated for its relative merits. Sometimes the decision depends upon the reactions of others; sometimes chance factors are involved; often, decision making is accompanied by a perplexing mixture of success and failure. Decision making involves lots of different kinds of knowledge and planning behavior. Decision making requires cognitive activity and is psychological in nature. Decisions are made by an individual or through a group process.

The term "expert system" or knowledge-based system" are computer systems that have evolved from research in the field of artificial intelligence. These computer programs are designed to represent and apply factual knowledge of specific areas of expertise to solve problems. Experts use public knowledge and private knowledge. The public knowledge includes published definitions, facts, and theories about specific domains of study. The private knowledge consists largely of "rules of thumb" that are referred to as heuristics. It is the heuristics that enable experts to make educated guesses when necessary, recognizing promising approaches to problems, and deal effectively with incomplete information.

An expert system is a computer program that behaves in manner similar to a human expert. It models the decisions of human experts, works in a specific task domain, gains power by having a large amount of knowledge and uses an automated inference process to draw conclusions from that knowledge. An expert system components can be broadly classified into a knowledge base, inference mechanism and a fact data base. The knowledge base is information derived from human expert's heuristics, judgements, and intuitions.

The inference mechanism applies the rules of the knowledge base to the information collected in a fact base to reach a decision. The fact base is information collected from the expert system user relevant to a specific problem, information inferred by the expert system and any other detailed information pertaining to a specific task.

Pace University has used a Business Strategy and Policy Simulation Game for a number of years in a Business Strategy and Policy course. Data has been collected over a five year period relevant to the business decisions made while playing the game. I used this data to develop an expert system based on group decision making.

The Management Department faculty from Pace University, that use the Management Game, had been individually given a set of 22 possible variables that influence the decisions made while "playing" the game. They rated the importance of the variables, and listed the factors that affect the variables. The decision variables are pricing, advertising, product research and development and so forth. Some of the factors influencing the pricing decision, for example, would be competitor prices, average cost to produce each unit, demand expectation and the like. They submitted their responses and they were classified and resubmitted, using a modified Delphi system. These results and the collected historical data were used to develop the expert system. The internal data contains financial statements such as an income statement, flow of funds and financial position statement. It also contains external data such as GNP, personal consumption expense, industry estimates, etc.

Recent active data from a previous game session were submitted to the expert system. The decisions outputted from the expert system were compared with game output. This was the scheme used to "fine tune" the expert system.

The expert system was then run "live" for a simulated time frame of twelve business quarters (three years). It became one of the groups of decision makers competing with six other student groups for market share, profits etc. in one industry. The presence of the expert system was not known to the students so as to preserve the integrity of the game sessions.

The comparison of the expert system decision making capabilities with the students showed average to poor student decision makers did not perform as well or better than the expert system. The better students learned from experience of reviewing the output from each business quarter and outperformed the expert system. An expert system in general contains knowledge but it can not dynamically "learn" new knowledge.

The use of the expert system described has a number of applications to training students in decision making.

(1) Active participation by students stimulates their interest in the subject.
(2) The instructor's experience and expertise can be illustrated.
(3) Students can input values for the decision variables directly into the expert system and compare the results with the decisions they would have made.
(4) Students can experiment with the expert system by adding new decision rules.
(5) The complete set of decision rules can be changed for any problem domain.
(6) Students learn from first hand experience, the capabilities and shortcomings of expert systems.

The time frame for accomplishing this task is about two semesters. The knowledge of expert systems is a prerequisite to completing the system during this time period. The first semester is needed to develop the system and "fine tune" it. The second semester is needed to run the system "live".

My expert system was written for a microcomputer in an expert tool called EXPERT-EASE. An illustration of a simple program is as follows:

The problem is to make a decision for an appropriate gift.

Money	Age	Gender	Gift
little	adult	*	calculator
much	adult	*	mindProber
litle	child	male	transformer
little	child	female	doll
much	child	*	computer

The structure of the rules follows a decision tree. Two rules for this simple system is as follows:

```
If there is little money and
        the age is child and
                 the gender is female
                         the gift is a doll.

If there is much money and
        the age is adult
                 the gift is a calculator.
```

Applications for expert systems in systems design could include:

(1) The Analysis phase
cost/benefits/ schedule
data base requirements
physical requirements
hardware configuration
software specifications
(2) The Design Phase
Pert chart for scheduling the development and implementation
(3) The Testing phase
Determining acceptance tests
Developing test data
(4) The Maintenance Phase
Determining a maintenance schedule
Decisions for type and cost of maintenance

All these applications listed above could be assisted by the development of an expert system that contains the experience and expertise of a system analyst with many years of business experience or the Systems Courses instructors. Students often think "systems" classes are both illusive in content and common sense. What they need to experience first hand is designing an ill defined system, be fraught with complex and difficult decisions and the need for contigency plans and adjustments in the design.

A simulation of these problems through the use of an expert system may substitute for lengthy lectures, reading case studies and isolated exercises. Students can determine their decisions, see what an expert system's decisions are, engage in a class discussion to evaluate the expert's decisions and learn from the total experience.

Implementing these various expert systems is not an easy task. In addition to the knowledge base, inference engine and fact base, one needs to develop a meaninful human-system dialog that does not become tedious or confusing. Following are a few suggestions for an interchange system.

```
I. Multiple Choice
        1. Amount of Money to Spend
                1) little
                2) much
        1*
        2. Age of Person
                1) Adult
                2) Child
        2*
        3. Gender of Person
                1) Male
                2) Female
        2*
* user responses

II. Direct
        1. Dollars
        AMOUNT OF MONEY TO SPEND: 30*
        2. Numeric
         AGE: 6*
        3. Verbal
        GENDER: female*
* user responses
```

The type and form of the questions is often motivated by the nature of the data to be asked as well as the amount of data needed from the expert system user. The system user can also query the system for an explanation of the responses or need for more information by the system. One advantage of expert systems is the ability to produce "what" the answer or decision is as well as "why" the answer or decision. The "why" response is usually a synopsis of the inference system rules "fired" as a result of the information input to the system from the user.

Expert-Ease can handle up to a maximum of 31 decision variables that can be assigned a maximum of 32 values. If one has a problem that requires a larger system, many other expert system tools exist.

In conclusion, this expert system usage approach to classroom instruction is part of the current interest in artificial intelligence applications. Students do not physically leave the classroom, but they engage in "active" participation and learning. We all appreciate the need for theory and training in information systems. This interacton with an expert system has been a learning experience and an exciting new approach to teaching a business course.

References
1. Newell, Allen and Herbert A. Simon. Human Problem Solving. Prentice-Hall, 1972.

2. Winston, Patrick H. and Karen A. Prendergast. The AI Business. The MIT Press, 1984.

3. Hayes-Roth, Frederick. Building Expert Systems. Addison-Wesley, 1983.

4. Simon, Herbert A. The New Science of Management Decision. Harper & Brothers, 1960.

5. Weiss, Sholom M. and Casimir A. Kulikowski. A Practical Guide to Designing Expert Systems. Rowman & Allanheld, 1984.

6. American Management Association, Inc. Simulation and Gaming: A Symposium. Report No. 55. American Management Association, 1961.

Marian Sackson
School of Computer Science and Information Systems
Pace University
Bedford Road
Pleasantville, New York 10570

(914) 993-3444

A Gateway Course in Systems Analysis & Design

Mary K. Ronner
Dept. of Math & CIS, Mercy College
Information Systems Dept., National Life of Vermont

This tutorial presents the guidelines for organizing a "live" systems course aimed at balancing theory and practice. It is a single term proposal for the Undergraduate senior-level student.

Areas of focus:

A. Course prerequisites/Background
B. Preliminary faculty efforts/setup
c. Project screening and refinement
D. Course Start-up
E. Project monitoring
F. Project follow-up

A project time table is included for your practical reference. Project samples and their "success" stories will be available during the presentation.

A. BACKGROUND

Mercy College is a small commuter college with its main campus located 20 miles north of New York City in Westchester Co.. With reasonable tuition and easy accessible by bus, train or car from throughout the county and NYC (satellite campus in the Bronx), it is a prime educational resource for working students of all ages.

The CIS curriculum is one of the major areas in the Math and Computer Science department. It is comprised of a heavy concentration of programming languages and technical theory and a smattering of theoretical business courses.

The course entitled Systems Analysis and Design which we will discuss here, is taken as one of several course choices in the final term before graduation. Students come to this course by informal direction from CIS faculty who perceive this course as a culmination of the business information systems track. The students' formal background must include: COBOL Programming, File organization skills (Batch & Direct Access), Management Information Systems (a business course). Informally, many come prepared to handle topics in DBMS, CICS, Fourth Generation/User friendly languages and LANS. Starting in Sophomore year, IBM sponsors many student internships for our CIS majors . This affords access to state-of-the-art areas not necessarily approached in the classroom.

The Systems Analysis and Design course, a 16 week dynamo is then treated as a capstone course where there is a great effort to bring together all DP knowledge areas. Because of the financial background of our CIS student population, there is no presumption that graduate school will start the following term. Therefore the job market is our prime target with our aim to make the transition from theory to practice as easy as possible.

B. THE "LIVE" PROJECT BEGINS - faculty search for projects: Media, forms, responses, follow-up.

Approximately 6 weeks prior to the terms' beginning (late December and July) the following advertisement will be placed in the business section of all local Sunday newspapers - a 2 week run:

THINK YOU NEED A COMPUTER ?
FREE
NO OBLIGATION
COMPUTER SYSTEM RESEARCH & DESIGN

Contact Prof. Mary Ronner
Dept. Of Computer Science
Mercy College, Dobbs Ferry
914-693-4500
by Aug. 3

A response date, contract person, address and phone number are essential to this ad. Expected response time is established at 2 weeks from the date of appearance. Over the course of 2 years, responses numbered from 4 to 10 mainly from single proprietorships and voluntary organizations. A secretary in the CIS department would collect initial information on index cards as follows:

NAME:	
LOCATION:	NYC
	WESTCHESTER CO.
	CONNECTICUT
	NEW JERSEY
PHONE:	DAY
	EVENING

C. PROJECT SCREENING AND REFINEMENT

After the 2 weeks response period was complete, the projected course instructor would collect the index cards and start the telephone interviews asking the following:

Type of business
The problem area
Chief resource person and availability
Existing computer system(s).

The faculty member would then explain the purpose of the project. It is important to emphasize now and throughout the term, that implementation is not part of the students' responsibility. Advise the client that if implementation is needed, it could be arranged as a private matter with individual students during the following summer or winter intercession month(s).

All inquiries are answered by the faculty member and once this step is complete, project evaluation begins. Evaluation or screening is based on problem depth, travel consideration and the availability of the resource person. Due to the condensed 16 week time frame, projects most likely to be considered are those with the clearest perceived problem definition and the ones where use of existing software products seem feasible.

All inquiries are then answered by an accept or reject letter (below). All project index cards are kept in a file for future reference. Note, even the inadequate projects are saved. (It have been my experience that the same inquiry may come in several terms and it saves time to have a back reference - you still call, in case of new developments, but you may be able to screen faster).

Sample acceptance letter:

Dear Potential Client:

The problem area we discussed Monday does seem feasible for my students to research. The actual Systems Analysis and Design course will start within the next 2 weeks. At which point, a student will be contacting you.

Thank you, in advance, for your contribution to this worthwhile research effort. If at any time you feel the need to contact me, do not hesitate to call. I am available on Tues. & Thurs. from 9 - 12 AM at my Dobbs Ferry office, 693-4500 x518.

Much success,

Sample rejection letter:

Dear________:

Thank you for the time we spent talking last Tuesday. Your project unfortunately does not fit my students' course requirement and so for this term at least, we will not be able to work with you.

We will reconsider your project for future terms and if additional requirements do arise, please do not hesitate to call. I will be available on Tues. & Thurs. from 9 - 12 AM at my Dobbs Ferry office, 693-4500 x518.

Thank you, again,

D. COURSE START-UP

From the onset of the course, the projects chosen by the instructor are both explained to the students verbally and in writing. The course outline distributed on the first session of class is to include:

A project time table;
Evaluation methodology;
Project abstracts(composite of information collected on index cards).

The agenda for the first course meeting concentrates on clarifying each project; a discussion of the team concept; identifying specialty areas within the class - posing questions on involvement in internship programs (IBM, etc) and other work experiences; helping the student population become comfortable with each other by starting discussions involving these specialty areas. The latter non-academic function becomes essential to the ongoing give-and-take of the course. As a natural transition, 3-4 member teams will be, for the most part, in place prior to the second meeting of class. Initial faculty and student team consultations should be arranged at the end of class 1.

By the end of the second class, all un-teamed students will be placed and appointments for each team to meet with the instructor to clarify the project details will have either been completed or are scheduled for the immediate future.

As a result of the faculty-team initial meeting, each team member will understand their individual role, the team effort and the importance of peer evaluation to their final grade - THERE WILL BE NO LAGGERS!! Once the instructor feels confident that the team understands the project, he/she will encourage a that first client meeting take place shortly - in time for discussion during the third class session.

E. PROJECT MONITORING

The ongoing emphasis during the course will be:

a) staying on track;
b) full team participation;
c) opening doors to sources of new information-library, specialty area oral reports.

Class time will be divided into 2 distinct areas, half theory lecture and half problem-solving in regards to detailed projects. All teams are to be prepared to discuss their project to some degree of sophistication by the third class. It is the goal of the instructor to support practical problems with a theoretical reference.

If a problem lingers for weeks (if you've addressed the same issue before!), make sure that the students understand that this is the last class discussion and now an office consultation is in order.As each team reports weekly progress, the instructor must note:

a) any off-track efforts;
b) any areas being ignored or overlooked;
c) the need to solve a stumbling block thru peer discussion;
d) any non-involved attitudes among the team members.

Essential to the progress of the classroom discussion is the instructors ability to parallel text discussion with the "live " systems development phase, ie: discuss questionnaire forms during the "live" detail analysis; PERT, CPM,etc. during "live " project control.

Quizzes should be administered from time to time (max 3) to test understanding of theory and to detect any weak areas in which the instructor can lend extra support. From experience, I found that it was possible to finish the theory material well in advance of project completion. I began slowing my pace and using this extra time to individualize with teams as an extra support person. A new voice (a guest speaker) is always helpful to introduce here, in support of a special area. A student with specialty area knowledge or a corporate DP person are excellent resources.

F. PROJECT FOLLOW-UP

At the course finale, an oral presentation by each team member is the order of the day. The presentation is made to the instructor and fellow classmates who should be prepared with questions based on the Design Abstract distributed prior. All team members should be alert to questions presented to a teammate. A portfolio is than presented containing all work - questionnaires, notes, transparencies and of course, the initial and final design proposal. The team should present a copy of the final proposal to the respective client and offer to make a presentation of the developed system on premises.The instructor should also contact the client with a note of thanks and an offer to meet with him to discuss any follow-up. As a final assignment, each student is required to complete individual evaluations of his/her team members. A comparative evaluation of a team member across the team is an vital source of final grade development.

THIS COURSE IS WORK! There is no doubt about that. What is achieved is 3-fold:

-- a sense of completeness at the end of a highly technical curriculum

-- a sense of the team effort which will follow most students into their career

-- exposure to speaking and answering question spontaneously.

What it does not achieve:

-- guaranteed success - sometimes disappointment to both the team members and the client when a project proves unworkable - there is no disgrace - the learning experience is still shared;

-- exactly equal learning experience for all students - a truth of many research courses;

-- system design from scratch - modified software only.

BUT THE TRUE VALUE IS WORTH THE EFFORT!

A more comfortable CIS major enters the world of business with eyes open to a valid sense of what is waiting for him/her to pursue! A track record for the CIS major area in general, with good feeling in the surrounding community of what your college can achieve. The projects are self-generating with each seeding several more. The sense of accomplishment to both student and instructor is overwhelming-------- so is the desire to rest!!!!!!!

TIME TABLE

Pre-class

6 Wks.	Newspaper requests
4 Wks.	Phone interviews
2 Wks.	Accept/reject letters

Class begins:

Wk. 1	Overview of Systems Analysis Projects discussed Teams organized Initial team-instructor meeting set
Wk. 2	Detailed Analysis with Tools Teams completed Initial team-instructor meeting completed Initial client meeting set Oral reports assigned
Wk. 3	Detailed Analysis with Tools Initial project discussions
Wk. 4	***Quiz 1*** Project review (brief) Overview of Systems Design
Wk. 5	**Preliminary Reports** Output Design File Design
Wk. 6	Project review (brief) Data Base Design Input Design
Wk. 7	Project review (brief) Process Design Acquisition of Hardware & Software
Wk. 8	**Detailed Analysis Reports** (written & oral presentations) Modular Design
Wk. 9	***Quiz 2*** **Detailed Analysis Reports** (written & oral presentations)
Wk. 10	Project review (brief) Overview of Development
Wk. 11	Project review (brief) Management of Systems Projects
Wk. 12	Project review (brief) Systems Follow-up
Wk. 13	***Quiz 3*** Project review (extensive)

Wk. 14	*****DUE**** Documentation Training Manuals Design abstracts (for pre-presentation review) Final project review
Wk. 15	Design & Implementation Presentations
Wk. 16	Design & Implementation Presentations

References:

Eckols, Steve. How To Design & Develop Business Systems, M. Murach & Assoc.,Inc, (1983).

Edwards, Perry. Systems Analysis, Design & Development with Structured Concepts, Holt, Rinehart & Winston, (1985).

Kern, Robert. Business Computers, Planning, Selecting & Implementing Your First System, Merrill, (1985).

Whitter, J.L., L.D. Bentley. Systems Analysis & Design Methods, Mosby Publishing Co., (1986).

Mary K. Ronner
National Life of Vermont
Montpelier, Vt. 05604

(802) 229-3821

DEVELOPMENT OF FIELD-BASED MANAGEMENT INFORMATION SYSTEMS SIMULATIONS

Dr. Stuart A. Varden
Associate Professor
School of Computer Science and Informations Systems
Pace University
New York City

"My college course work should give me a good conceptual foundation, but it's not going to be of much use to me in the real world. I'm going to have to learn everything all over again once I get a job. I'm really in school to get the degree."

(From a recent conversation with an MBA student on the practical value of graduate education.)

GRADUATE BUSINESS STUDIES UNDER FIRE

In recent years, graduate education in business and other fields has been characterized as overrated and out of touch with current business trends and practices. Some critics have accused graduate programs of selecting students mainly for their perceived ability to be successful students, while paying little regard to their potential as business or professional leaders. The presumption seems to be that those who make it through the program will also succeed in their chosen fields. What often is lost is the admittedly less tangible and harder to quantify objectives of developing people with independence of thought, good leadership ability, and a concern for moral responsibility. These are the people who will be willing to take risks to create new products and markets, and will move society forward.

It seems clear, then, that any opportunity to tie student learning more closely to current practice should be considered. The need to relate graduate education to current business practice is especially crucial in the information systems curriculum. Of all the issues that challenge today's complex organizations, none is more perplexing than how an organization should go about structuring and managing its information resources. It is only in recent years that the idea that information is an important organizational asset has been generally accepted. Moreover, the view of information systems as a strategic competitive weapon is only now beginning to emerge.

Because of the relative immaturity of the information systems field and the rapid rate of change it has undergone in its short history, approaches to "real world" problems generally are expressed in the form of anecdotes or "war stories". When an attempt is made to treat problems in a systematic manner, the result often is an informal check list of "does" and "don't" which has no theoretical or conceptual base.

The information systems field has a very active research community to be sure, but research findings usually are reported in professional journals or conference proceedings. They are rarely read by the information systems practitioners who would appear to be the people most likely to benefit from them. Instead, the primary consumers of research findings are other researchers. What appears to be missing, then, is a joint program of cooperative activities that will bring together information systems practitioners, university researchers, and students to critically explore the practices and issues in the field. The research effort described here is a step in this direction.

THE RESEARCH EFFORT

The research effort aims to apply the field-based simulation approach to more closely relate the student's educational experience to current practices and issues in the information systems field. This approach has not yet been used in the field.

The research effort is bringing together representatives from the information systems field, Pace University faculty, assessment design experts, and Pace University graduate students. A major task of the research is to identify a series of problem situations and alternative actions that would be used in the simulations. The recently formed Information Systems Department Advisory Board at Pace, New York Campus, composed of some thirty industry representatives, promises to provide an excellent pool from which to draw the panel of expert practitioners.

In addition to student use, the series of simulations should provide a useful and non-threatening professional development activity for practicing information systems professionals. The simulation, viewed as a self-assessment tool, could be made available to the information systems community through a number of channels including professional organizations or a university-based assessment center at Pace.

ADVANTAGES OF THE FIELD-BASED SIMULATION APPROACH

The field-based simulation approach should prove to be a very useful means of cooperatively exploring a range of important practices and issues in the management information systems field. This is because it focuses on decision-making behaviors in concrete situations. The content of each problem is based on a situation that might be encountered in actual practice and reflects the type of knowledge-base and decision making that is required to resolve it.

The simulation experience itself, from the point of view of the participant is a non-threatening situation in which the cost of making a poor decision in minimal.

A simulation has another advantage. It has the potential to become a vehicle for self-reflection. When one is allowed to see the results of one's actions, one is provoked to ask "Why did I take the course of action I did?" or "Is there some basic weakness or characteristic of my approach to decision making that I do not fully understand?" This process of critical self-examination is perhaps the most powerful and enduring method of self-improvement.

HOW A FIELD-BASED SIMULATION WORKS

The participant is presented with an problem situation or "opening scene" which requires a response. The participant assumes the role of the practitioner. During each stage of the simulation, the participant is provided a number of options from which to choose. Once chosen, the consequences of decisions that he or she has made are revealed. Thus, new information now is available to the participant.

The revealed responses then can be used as a basis for further, more informed responses to the situation as it unfolds over several stages. This dynamic quality of the field-based simulation approach permits the successful representation of a wide range of concrete situations that can occur in professional practice.

The participant is provided with some form of assessment of the way he or she has handled the problem situation. The assessment gives an indication of the quality of the choices make.

SCORING THE SIMULATIONS

Each alternative choice is assigned a numerical weight. The higher the number, the more appropriate the choice. Negative weights are assigned to choices that the expert panel has judged to be unsuitable to the situation. Relatively harmless choices receive near-zero or zero weights. Several scores are possible. They are reviewed below:

Proficiency:
Is the percent agreement with the recommended solution in selecting desirable choices and avoiding undesirable ones.

Efficiency:
Can be thought of as an indicator of how directly the participant arrived at a solution. If time is wasted or resources poorly used, the efficiency is low even if the eventual solution arrives at is the recommended one.

Overall Competence:
Is the weighted combination of proficiency and efficiency scores.

Errors of Comission:
Is a count of the number of undesirable choices that were selected.

Errors of Omission:
Is a count of the number of desirable choices that were overlooked.

In addition to the quantitative scores that can be computed for each problem solution, aggregate scores across a series of problems can be computed. Moreover, an analysis of the participant's pattern of responses can lead to more qualitative characterizations of the participants decision-making style. These include willingness to take risks, degree of attention paid to details, and the ability to organize and recognize trends from a collection of specific facts.

THE DELIVERY SYSTEM

Field-based simulations for the purpose of professional licensing or self-assessment currently are used in veterinary medicine, hospital administration, and other fields. Up to now, the simulation setting and method of administration has been in the form of a paper and pen test using latent-image printing. The use of latent-image printing allows the participant to uncover the consequences of choices that he or she has made. This is done by using a latent-image marking pen which contains a fluid that makes a response visible when applied to a response position chosen by the participant.

The paper and pen format, however, does not provide for immediate scoring and assessment of the participant's performance. Participants often must wait days or weeks before any feedback is available. This severely reduces the educational value that the simulation experience might offer.

To enhance their educational value, the simulations would be delivered via a microcomputer. Software is being developed to delivery the simulations and provide immediate assessment of the participant's handling of each problem situation compared with that judged best by the panel of experts and current literature. Recognized assessment methods already are available to provide a quantitative as well as a more judgmental measurement of the participant's performance. Using a computer also allows tracking the order in which a participant chooses options. This may be an important assessment factor.

STEPS IN THE RESEARCH

Finally, we review the research plan. The following steps make up the plan of the research effort.

Task To Be Performed

1. Confer with Pace faculty and selected expert practitioners to identify skills and problem situations which appear to be suitable for the design of the simulation.

2. Convene a panel of selected practitioners to participate in the design of the simulation.

3. Determine desired target audience and application of the simulation,including definition of the feedback features.

4. Design training materials for development of drafts by the practitioner panel.

5. Convene for a one-or-two day meeting a panel of expert practitioners from the business community to design the simulation.

6. Prepare revised versions for the simulations. Convene a panel to react to the current drafts.

7. Send materials to selected practitioners for the purpose of assigning weights to the options within the simulations.

8. Design and implement software to administer the simulations.

9. Reconvene the panel of expert practitioners to operator and critique the simulations.

10. Modify simulations as indicated by the panel of experts.

11. Introduce simulations into Pace courses where appropriate.

REFERENCES

1. McGuire, C.H. "Simulation Technique in the Teaching and Testing of Problem Solving Skills". Journal of Research in Science Teaching 1976, Vol. 13, No. 2.

2. Smith, I.L. "Use of Written Simulations in Credentialing Programs." Professional Practice of Psychology, 1983, Vol. 4, No. 1.

3. Smith, I.L. & Oldak, R. Validation and Testing of Self-Assessment Instruments. New York: Professional Examination Service, 1981.

Stuart A. Varden
School of Computer Science and Information Systems
Pace University
Bedford Road
Pleasantville, NY 10570

(914) 993-3396

Developing Error-Resistant Application Systems

A Case Study for Systems Analysis and Design Courses

Janet M. Cook
Department of Applied Computer Science
Illinois State University, Normal, Ill. 61761

ABSTRACT:

People expect computers to be intelligent, doing the right thing and showing common sense. That doesn't happen. The common sense must be provided by the developers of the application system who need to anticipate the ways that system users may introduce errors into the operation. College students tend to assume that people will follow directions, and they develop their systems accordingly. They need to experience the tremendous variety of causes of "user error" in order to participate intelligently in consideration of methods of error control.

EDUCATIONAL PURPOSE

Systems which merely handle correct data correctly are raw prototypes, not usable products. Systems don't get correct data entered correctly, especially now that access to application systems is being placed in the hands, keyboards and screens of users who are not full-time keyboard operators. As Howard Anderson of the Yankee Group says, "Every organization has its share of damn fools. Now every damn fool has access to a computer."[1]

Few undergraduate students have observed a complex system closely enough to appreciate the variety of ways that ignorant, interrupted or exhausted users can misuse it. This exercise simulates observation of a system similar to many that they know. It begins with a brainstorming session on sources of potential error, then points out what analysts and designers can do to anticipate potential problems.

The scenario describes a retail sales system which uses programmed Point-of-Sale Terminals, an application already familiar to students. It incorporates

(1) real-time processing which accesses a pre-stored file on a disk,

(2) naive and fickle customers as providers of data,

(3) batch jobs run using data accumulated on several dozen tapes,

(4) incentives to fraud on the part of both customers and clerks.

The possibilities for error are enormous. Because students have approached this type of system in the role of customers, clerks or both, they bring to the discussion a user's view of errors-i.e., "It's the system's fault!". That is the proper frame of mind in which to consider the problem of how errors can be reduced.

SCENARIO (See Figure 1)

A retail store, Miller's Miscellaneous Merchandise Mart, (MMMM), has installed Point of Sale Terminals (POSTs) as cash registers at its check-out lanes. Each register is a stand-alone microcomputer with a printer for cash register receipts and a disk drive holding a diskette listing current sale items and their prices. The register copies the receipt line by line onto a tape cartridge as well as printing a carbon.

Each item stocked by MMMM has a tag (H.1)[2] which gives the item's 6-digit identification number and its price. The check-out clerks have a standard routine:

(1) Type in the item's 6-digit identification number and press "#".

(2) Type in the price on the tag and press "$". If this is a sale item, the register will find it in the SALE file on the disk and display the sale price. If not, the register will display the price typed in.

(3) If more than one of this item is being purchased, type in the number being bought and press "*".

(4) For coupons or corrections, type in the value and press "-".

(5) When the transaction is complete, press "T". The POST will display the total owed for purchases, tax owed, and the total due.

(6) The customer will offer cash, a check, or a credit card. Type in the amount given and press "G". The POST will compute the amount of change due and display it. (H.2)

While the printer and screen are keeping up with the current customer's transaction the tape cartridge is recording every activity. When the cartridge is full, the recorder will BEEP and the POST keyboard will lock until a new tape is loaded. At midnight, the tape cartridges are taken to the store's computer center. A computer operator runs a program which asks for one cartridge at a time, subtracts each of today's sales from the copy of the store inventory stored on a hard disk, and prints a new inventory list for the store manager.

Every week the store has a Super Special Sale on something, say, light bulbs. (H.3) Each checkstand has a case of that item and the clerk makes a penny of commission on each SSSale item sold. The purpose is to encourage the clerks to work fast since by handling more customers they make more commissions.

Every weekend the computer operator runs the week's tapes through a commission program which is part of the payroll system. The amount of commission earned that week by each clerk is added to that paycheck and printed summaries of commissions are sent to the payroll and personnel offices through inter-office mail. The past week's tapes are then returned to the store for reuse.

DIRECTING THE DISCUSSION: WHAT COULD GO WRONG? (H.4)

ACCIDENTALLY:

(1) What could happen to make a customer's bill come out wrong?

(2) What could happen to make a new inventory count come out wrong?

(3) What could happen to make a commission come out wrong?

INTENTIONALLY:

(1) If the customer is dishonest, what could go wrong?

(2) If a clerk is dishonest or malicious, what could go wrong?

(3) If a computer operator is dishonest or malicious, what could happen?

FOCUSING ON THE ERRORS

On the blackboard, have someone write down the possible causes of error as the class suggests them. Write everything. When the list seems to be complete, go back to remove duplicates, then turn the list into a table showing where the effects of each error will appear:

<u>Source</u> <u>Affects</u>:	<u>Bill</u>	<u>Inv</u>	<u>Comm</u>
During Check-out:			
Press wrong number key	Y	Y	Y
Press a key too lightly or too long	Y	Y	Y
Miscount number of items purchased	Y	Y	Y
Group together unlike products	Y	Y	Y
Enter mismarked ID#	Y	Y	
Misread valid ID#	Y	Y	
Enter mismarked price	Y		
Misread valid price	Y		

(H.5 is a more exhaustive table of sources of error.)

Highlight the errors that are caused by hardware problems. Circle those caused by malice rather than accident. It will be apparent that the vast majority of the problems listed are "human errors" and are accidental.

REFLECTING ON THE ERRORS

Some of the errors are human errors during check-out, some are human errors during programming or processing, and some are hardware errors. Which occur more often, human errors or machine errors? Human errors during check-out errors or human errors in the computer center? In ordinary daily operations, which category of errors causes the most damage to the company? (H.6)

Some of the errors are accidental, others malicious. (H.7) Which occur more often, accidents or malicious errors? In ordinary daily operations, which cause more damage to the company?

Some errors can be prevented. Others can be detected and handled before much damage occurs. Can hardware errors be prevented? Can their damage be controlled? What about programming errors? Processing errors? Check-out errors? (H.8)

Who is responsible for preventing errors? Who is responsible for controlling the damage they cause? (H.9)

Should the store management bother to take a physical inventory of stock on the shelves, or should they accept the figures provided by the inventory file? (H.10)

When a keyboard error is made, who should be responsible for correcting it? Should the clerk be able to make unlimited corrections? Should all corrections be approved by a manager? Some compromise? (H.11) In one chain store, clerks can make two corrections for a single customer. After that the clerk must call a supervisor to make the change. What is the point of a policy like that?

Should the program allow a customer to decide not to buy something after seeing the total that was rung up? What should be done if the customer decides to cancel the whole transaction and not buy anything? Customers do do that when their credit limit is exceeded, ... (H.12)

When an error in processing is detected, who should be responsible for correcting it? Should the operator make the change on his/her own? Should all corrections be approved by a supervisor? Who should determine how soon the correction needs to be made? (H.13)

What can be done to cut down the possibilities for error? Can programs be designed to catch some or all data errors? Can systems be designed to detect all logic errors and deliberate loopholes? Can procedures be designed to catch or report all processing errors? (H.14)

USING THIS DISCUSSION AS A REFERENCE DURING THE REST OF THE COURSE

If this exercise is used at the beginning of the unit on the Investigation Phase of the System Development Life Cycle, the remainder of the course will benefit in two wa much more observant shoppers after the exercise is over and will discover many problems in the systems that they use as customers, but they will be looking at the problems with a speculative eye, mulling possible solutions. Second, as you present new aspects of the SDLC, you will have their newly-primed observations to draw on when you need examples of how to discover what errors may occur and how they can be prevented or managed.

Specifically, in the Investigation phase of the SDLC, during interviewing, ask the users for horror stories and anecdotes of the interruptions, distractions, and mistakes that have bugged them and their colleagues. "What was the most frustrating day you've had?" Then ask their managers the same questions. "For example, the check-out clerks at the MMMM stores can come up with more tales of the wild things that customers do than our analysis team would ever dream of, and their managers can tell tales of the complaints that customers bring back after they have checked over their register receipts at home."

During the Cost-Benefit Analysis of hardware, consider the advantages of "intelligent" I/O devices and terminals. "For example, MMMM should check into current error rates of scanners which read item tags and compare their rates to the error rates and processing speeds of clerks working without scanners."

In the Detailed Design and Implementation Phase of the SDLC, setting up the working procedures so that it is easy to be sure that the right things are being done/used in the right order. "For example, in the MMMM retail system, they could display the date that is on the sale disk whenever a clerk signs on to use a register. They could number the tape cartridges for each register and use them in sequence. They could have the programs print out a list of the cartridge numbers as they are processed, always keep filled tapes in the same place, keep filled tapes apart from blank tapes, ..."

Build in "escape" features so that users can cancel the last entry or the last whole transaction without losing everything else they have done. "In the MMMM stores, clerks say that a lot of customers misread tags on sale items. When the actual sale price is displayed, the customer decides the item costs too much. Some customers who want to charge purchases are rejected because they don't have enough credit left. Others decide at the last minute that a large appliance is too expensive, or get angry about the delay in getting a check approved and walk out."

Check data entered for consistency. "For example, if MMMM's POSTs can access a disk that holds a complete product list, the program can check to see whether the item number and price entered match the values in the list. If not, it would notify the clerk to doublecheck the data. The program shouldn't force the clerk to use the price in the list, however, because items like out-of-date film may be discounted."

In the Installation Phase of the SDLC, make the procedures easy and train the staff to use them. "In the MMMM stores, clerks are trained to use the touch system on their terminal keyboards so that they can reach numeric keys and control keys without ever looking awayfrom the tags on the items sold."

Provide quick access to references on procedures for special cases. "An MMMM store customer presented 'FREE Product' coupons for everything bought. The register balked at a purchase with only coupons for payment. How should the clerk handle it? "

In the Review phase of the SDLC, arrange for monitoring of how often escapes are used and by whom. If everyone uses escapes every day, there may be a problem with the system design. If one user needs them several times a day, that user many need more training or may be exploiting a loophole in the system. One retail chain lost $31M in 18 months because a data entry clerk raised her productivity figures by leaving telephone numbers out of the records of new credit customers, who then couldn't be reached if they missed payments.

Recognize that some errors will not be caught when they occur, that there are problems that only the user can control.

"In the MMMM Store situation, there is no way to check that the multiplier used when a customer buys several of the same item is correct. Even if the clerk must ring up each item separately, it is easy to lose count. System design cannot ensure that this figure will be correct. That remains the clerk's responsibility."

SUMMARY

We are preparing students to solve problems so that correct activities do take place **and incorrect ones do not**. Anticipating how and where errors can occur in application systems takes a user's insight. In this case study students combine their own experience as users with their growing understanding of the tasks of system analysis and design.

1. Anderson, Howard, 1985. Speech: "Resolving the Conflict: User Friendliness vs. Effective Security" at the 12th Annual Computer Security Conference of the Computer Security Institute, Rosemont, Ill., Nov. 5, 1985.

2. (H.1) is the first page of the handout distributed by the author during the presentation. The handout pages are diagrams or large-print copies of discussion questions from which transparencies can be made for classroom use.

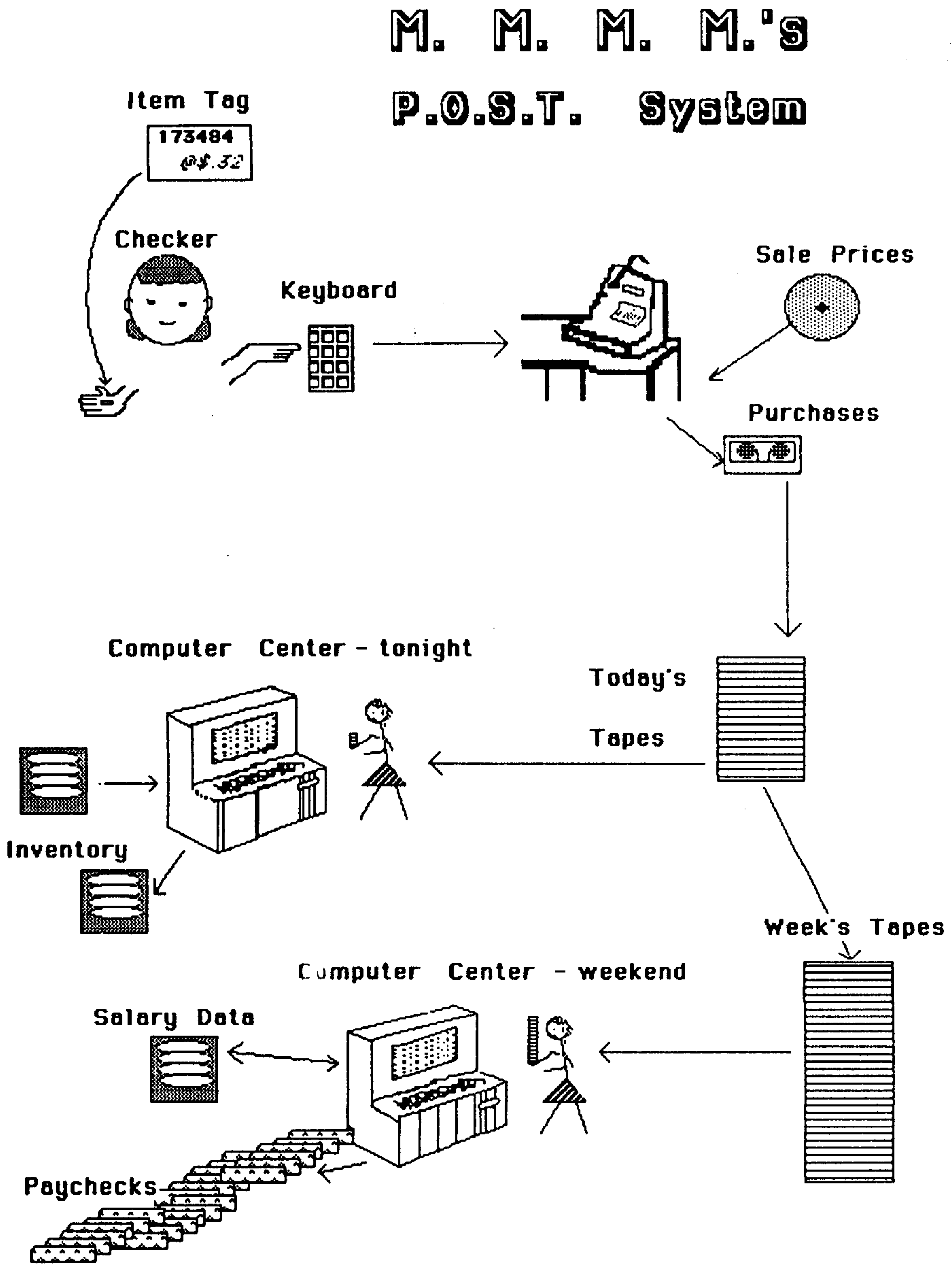

M. M. M. M.'s
P.O.S.T. System
Item Tag
173484
@$.32
Checker
Keyboard
Sale Prices
Purchases
Computer Center - tonight
Today's
Tapes
Inventory
Week's Tapes
Computer Center - weekend
Salary Data
Paychecks

Figure 1

TEACHING METHODOLOGIES FOR APPLICATIONS PROGRAM DEVELOPMENT II (CIS-3)

SANDRA E. POINDEXTER, ASSISTANT PROFESSOR OF CIS
NORTHERN MICHIGAN UNIVERSITY, MARQUETTE, MICHIGAN

The material covered in an advanced programming class can be very difficult for many students to comprehend. Any strategies which make the course more interesting, understandable, and practical are of benefit. This paper summarizes six years of experimenting with various methodologies. Both veterans and new instructors for this course should find the ideas very helpful.

DEVELOPMENT OF SYLLABUS

There is a consistency among CIS faculty, textbooks, and the DPMA Model Curriculum course CIS-3 (also listed as CIS/86-4) regarding the topics to be covered in a second programming class. The emphasis is on file maintenance, sort/merge, subroutines and expanded reporting facilities. I preface these topics with brief lectures (accompanied by handouts) on expected programming standards for assignments, Warnier-Orr, and walkthroughs. The expected standards item seems necessary as students often have differing backgrounds from the introductory COBOL course.

Point distribution for grading has evolved to be: Assignments - 50%, Walkthrough Presentation - 5%, Tests - 35%, Attendance/Participation - 10%. Students dislike the weight of tests, but I maintain that an extraordinary amount of time may be spent on assignments, much assistance may have been received in order to complete the task, and employers would not expect an employee who earned an A or B in this course to need either extra time or assistance on the job. Tests serve to separate students into their truer grade categories. It is necessary to state this fact at the start of the course and at each test point to reduce student shock and discontent when their grade average for the course suddenly drops.

CASE APPROACH

The case approach, as I use it, consists of a series of assignments related to one application. They may be physically chained together as in a validation, sort, file maintenance series or merely logically connected to the same data files. Fabricated cases are the most frequently used because the instructor has more control over the problem being solved.

Live cases are an interesting alternative. Here the class writes the programs for an actual application to be implemented at the completion of the course. Student enthusiasm was much higher with this strategy. They felt their work now had added importance and the experience was more true-to-life. For the instructor there are two drawbacks: finding a suitable case whose scope and complexity are not beyond the students' abilities, and the loss of control over problem parameters. I selected an application of a project control system developed as a System Analysis and Design class project. The scope was reasonable since it was designed with the intention of being coded by the programming class. This idea had the added benefit of a Systems Analysis and Design class which was much more interested in their assignment knowing they were passing the work along to their peers.

MAINTENANCE PROGRAMMING

I feel strongly that students must learn to modify a program written by someone else. Not only does it test knowledge, but it is the most convincing argument given to students for the value of a well structured and documented program.

One technique is to use add-on assignments giving a basic problem first. After the initial assignment is graded, select one or two representative student programs and assign an enhancement to be done on the chosen version. I change the author to "unknown" so that the student author is not pestered by classmates and because original sources are normally not available in a real business setting. Examples of this would be to write a validation program and later add a sort, or start with a sequential file maintenance problem and later revise it to be an indexed sequential file system. The techniques also allows more concepts to be programmed since the more tedious DATA DIVISION coding can be reduced.

A second exercise is to give the students an unstructured, undocumented program to rewrite into a structured form. The program works but the student must figure out what is being done by reviewing the output and code alone. Even the best students find this challenging.

PROGRAM DESIGN

While students are encouraged to prepare structured design solutions prior to coding, few actually follow the advice. This is a well known fact among all parties involved. To force their effort I include at least two detailed design assignments. In these cases I allow two weeks (of a 15 week semester) for assignment completion; one for the design which is handed-in, and a second for the program. Requiring a mock up of the expected run results with the supplied test data is an additional way of verifying the level of understanding of a problem and can clarify vague instructor specifications.

Warnier-Orr is the design tool used, but others can be chosen. Students in my classes have already worked with flowcharting, structure charts, and pseudocode. I wish to give some working knowledge of as many tools as possible. Supplemental material is necessary for Warnier-Orr as it is not used in most textbooks.

WALKTHROUGHS

Instead of presenting my program solution to the class I assign one group of two or three people to each assignment. As a team they will work on the design or program developing a quality solution and present it to the class. They are responsible for obtaining transparencies and extra copies for distribution. All members of the team are expected to be a part of the presentation and answer questions regarding their solution. The walkthrough portion of their final grade is based individually upon preparation, appearance, understanding of the material and ability to answer questions, professionalism, and eye contact. The walkthrough is given on the assignment's due date. Students are given the chance to choose team members and indicate a preference for a design or programming project.

DOCUMENTATION

Students are now at a level where the view of a programmer's job must be expanded. Writing and testing code is only one aspect of the required duties. In addition to internal documentation, each program is accompanied by one of the following: User Guide, Operator Run Instructions, or Programmer Documentation.

Of these the User Guide is the most difficult for students to write. They have become accustomed to using computer jargon and no longer relate to the end user's perspective. The instructor has to stress that the purpose of the user manual is to instruct the user how to fill out the source document, request that the job be run, submit the transactions, read the results or whatever the situation may be.

VISUAL AIDS

Textbook illustrations and overhead transparancies can be quite good, but are severely limited by their one-dimensional nature. The best aids, particularly for file organization, are video tapes. There are several excellent ones on the market and if budgets permit are worth their price.

Any device that serves to clarify difficult concepts can be adopted. I use large numbered cardboard cards to explain sorting and file maintenance, stereo albums stacked with cardboard cylinders and children's stacking toys with platters to explain disk hardware and file organization. Where lengthy material will be written on the board I prefer to give a handout duplicating the material so attention can be given to the discussion instead of note-taking. However, I leave gaps in the handout material informing the class they cannot fall asleep or they will have incomplete information!

INTERACTIVE PROGRAMMING

If resources and support are available the ideal situation would be to use the mainframe and a telecommunications system interface. This gives students the real thing, but is often not possible. A close alternative is to use micro-computers with a good COBOL compiler. Some compilers use screen handler routines more effectively than others, but most allow chaining of menus, and immediate file updating through subroutines and indexed sequential file organization. This might be tried even if few machines or compilers are available, but team work rather than individual work would be done.

A third alternative is to use standard COBOL on the mainframe and write a main COBOL program to handle student subroutines. The ACCEPT and DISPLAY verbs can also be used with free form I/O. The obvious disadvantage is in not seeing or being a part of the interactions on a screen.

SUMMARY

Perhaps these "songs and dances" should not be necessary. But the fact is that the course material is quite difficult for the average student. Any technique which helps in the learning process should be used.

CONTEMPORARY CODING

John T. Overbey, PhD, CPA
Department of Accounting and Information Systems
Western Carolina University
Cullowhee, North Carolina 28723

James J. Taylor
Principal Government & Educational Management Systems
American Management Systems
1777 N. Kent Street
Arlington, Virginia 22209

An efficient coding system is one of the most important parts of any information system. Group, block sequence, hierarchial, and random are the principal logical design formats for coding systems. If the system is to be effective it must provide for the unique identification of each data item with a minimal expenditure of storage space and processing time and amaximum liklihood that the data is classified and entered correctly. At the same timecoding schemes must provide for the increased multidimensionality being required of organizations in their reporting. Traditional coding systems in their attempt to retain meaning and incorporate maximum information have become very unweildly. It is possible using today's electronic technology, to have a brief code that will point to a record in the system that can contain almost unlimited information about the transaction.

CONTEMPORARY CODING SYSTEMS

An efficient, effective coding scheme is one of the most important parts of any information system. A simple definition of coding is "The art of assigning a meaningful code number that defines an item's permanent characteristics.[1] A more formal definition of coding is "the assignment of numbers, letters or other meaningful symbols according to a systematic plan for distinguishing the classifications to which each item belongs and for distinguishing items within a given classification from each other".[2] A coding scheme then is a pattern for classifying the data going into an information system so it can be recorded, appropriately summarized, and made available for subsequent reporting and auditing.

A coding system must meet a variety of demands, some of which run counter to each other. If the system is to be effective, it must provide for the unique identification of each data item with a minimal expenditure of time and effort in the data input, a minimal use of computer storage space and processing time, and a maximum liklihood that the data is both classified and entered correctly. At the sametime coding schemes must provide for the increased multi-dimensionality being required of organizations in their reporting. Corporations need to report to external users and to internal management by product line, profit center, and project team as well as by factory and company. Government bureaucracies must report by program and function as well as by object of expenditure, organizational structure and source of monies, all integrated into a complex fund structure. Both corporations and non-profit organizations are more and more recording and reporting actual revenues and expenditures in conjunction with an allocated budget or plan. Thus greater and greater demands are being placed on the standard approaches to coding.

Types of Coding Systems

Codes can be constructed from numbers, symbols or letters and can be classified in various ways. In a mnemonic code letters are grouped to help users recall words. In a coding system set up for a university, for example, "SB could be used for the School of Business and "SLA" could be used to refer to the School of Liberal Arts. Such codes have the advantage of being understood by users. Also they provide up to 26 alphabetic characters per position whereas numbers allow only up to 10 alternatives. Despite these advantages, most people choose coding systems that utilize numbers.

Group-Codes - In a group code each position has a certain meaning. For example a university could use a seven digit student identification number in which the first position signified the college enrolled in, the second position the class rank (freshman, sophomore, etc.), the third position signified male or female, and the last four positions could be used as a unique identifier for that particular student. (This assumes that there are no more than 9999 students in the university). This type of code is also referred to as a chained code or a polycode.

Block Codes - Accounting charts of accounts are a good example of block codes. For example all assets would be given an account number from 1000 to 1999, liabilities would be given an account number from 2000 to 2999, etc. Thus in block coding a particular sequence of numbers is reserved for a certain class of data items.

Sequence Codes - In a sequence coding system the first data item encountered is given the number "1", the second is given the number "2", and so forth. This type of coding system is very useful for document numbers where good control dictates that each document be accounted for. The code itself gives the user no meaning. This type of code is also called a serial code.

Hierarchial Codes - This type of code is, in a sense, a subset of group codes. For example, within a university, hierarchial codes could be used to identify departments or cost centers. The first position could be used to distinguish between divisions: academic , housing, athletics, etc. The second could be used for colleges within the academic division, and the third for departments within a college. Thus the second and third positions cannot stand alone as they are not unique; and have no meaning until combined with the first position. Thus the accounting department and the english department might have the same one digit number. This type of code is also referred to as a monocode.

Random Codes - In a random coding system a data item is assigned a random number which has no meaning of any kind, but uniquely identifies the data item. Thus if there are under 100,000 data items a given data item can be assigned any number between 1 and 99,999 that is not already assigned. As stated this code gives no meaning to the user, but this system provides a unique identifier for each data item with the smallest number of digits of any other system except possibly the sequence code.

Traditional Coding Systems

The designer of a coding scheme has the above types of systems from which to select. For an accounting information system, the department or cost center identifier must appear on every transaction flowing through the system. In manual systems or electronic systems designed along the line of their manual forerunners, it was usually considered necessary to have a coding system with which the user could identify key components of the transaction and the department or cost center involved just by looking at the code itself. A University accounting system will be used to illustrate this approach and a standard group code will be used.

The example will be based on a fund accounting system for a typical state higher education system in which disbursements are made centrally and it is desired to meet the basic financial reporting requirements for higher education and also group disbursements by cost center to provide information for administrative control. In such a system, expenditures will be grouped by fund, function, cost center and object of expenditure. The term fund is used here to indicate a self-balancing set of accounts. Colleges and universities typically utilize several funds including a current unrestricted fund and a current restricted fund, a loan fund, an endowment fund, and one or more plant funds. Function denotes the purpose for an expenditure such as instruction, research or financial aide. To get cost center information it will be necessary to identify each institution in the state system, each school or administrative area within the institution, and each department within the school or administrative area. A partial listing of the required tables is given in Exhibit I.

The tables can be used to construct a group code to classify expenditures for a variety of reporting purposes. If the elements of the code will be identified in the order of fund, function, institution, college and department, then an unrestricted current fund expenditure for instruction in the accounting department within the School of Business of Western State University would be coded 1-1-01-02-03. For the sake of illustrating the point, this coding system has been simplified and has several underlying assumptions. It assumes no more than nine funds and nine functions, no more than 99 institutions, no more than 99 schools or departments within an institution and no more than 99 departments within a school. The system also makes some less obvious assumptions such as that the organizational structure within each institution and the department structure within each school is similar. It also assumes that the organizational structure within each area is hierarchial with no department crossing school reporting lines.

Modified Approaches

While such an approach to a coding scheme has its limitations, it is still commonly used in both user designed systems and in proprietory systems. Two new concepts have been developed which provide additional flexibility to the designer of a coding scheme. They are the use of attributes, or the key result field concept and the use of mapping in data entry with teleprocessing monitors. A key field in computer terms is a unique identifier for a record. In the example being used, (03) in the last position of code above refers to the Accounting Department within the School of Business (02) within Western State University (01), thus the numbers 01-02-03 can be a unique identifier to a record which contains additional information--the result--such as the actual name of the department, supervisor's name, mailing address of the department, etc.

The use of the key-result field carried to its logical extreme is that of the pseudo-code. In this case for example the accounting distribution is first defined by including full debit and credit information in the result field. Thus an arbitrary key points to a result field which specifies the debit and credit distribution. Thus the code "123" could be a key field in a record in which the result fields contain the following information: fund, function, institution, college, department, expenditure object code (debit) and credit offset, and much more such as name of department head, fiscal year, source of funding, etc., if desired.

The developing of data entry methods using teleprocessing monitors offers an additional method for achieving flexibility in the development of an account structure. Fields on teleprocessing screens are mapped to fields in the file. Thus, for example, the use of one field would indicate a revenue code, another an expenditure object code, and another a balance sheet account as opposed to utilizing the initial digits to distinguish between these types of accounts. Thus 007 in one field would indicate a particular kind of expenditure, while 007 in another field would indicate a particular kind of revenue, etc.

CONCLUSION

An ideal coding scheme has inherent meaning, maximum information and has individual codes that are very brief (no more than eight or ten characters). Brevity is important because short codes are easier, faster, and hence cheaper, to input. If a code has inherent meaning the user can look at the code and obtain information about the transaction directly from the code itself without having to go to some explanatory table. Maximum information is important because in todays complex world, organizations must classify and report transactions in many different ways.

Traditional coding systems in their attempt to retain meaning and incorporate maximum information have often become very unwieldy. Codes requiring the use of thirty, fifty, or even seventy positions were not uncommon. If the user is willing to go with little no inherent meaning in the code, it is possible using todays electronic technology to have a very brief code that will point to a record in the system that can contain almost unlimited information about the transaction. Some systems today have space to store as much as 250 different pieces of information about a transaction.

FIGURE I

FUND		FUNCTION	
Code	Meaning	Code	Meaning
1	Current Unrestricted	1	Instruction
2	Current Restricted	2	Research
3	Loan	3	Student Finan. Aid
4	Plant	4	Instit. Supp.

INSTITUTION		SCHOOL OR ADMINISTRATIVE AREA	
Code	Meaning	Code	Meaning
01	Western State Univ.	01	Arts and Sci.
02	Mountain State Univ.	02	Business
03	Eastern State Univ.	03	Allied Health
04	Southern State Univ.	04	Student Serv.

DEPARTMENTS IN THE SCHOOL OF BUSINESS

Code	Meaning
01	Information Systems
02	Marketing
03	Accounting
04	Finance

REFERENCES

[1]"Managing MRO by the Numbers", Purchasing, November 25, 1981, p.106.

[2]National Association of Accountants, "Classification and Coding Techniques to Facilitate Accounting Operations," Reserved Report 34 (NY: National Association of Accountants 1959), p.3.

MICROCOMPUTERS IN THE MIS CURRICULUM

Mary Sumner and John Schrage
Department of Management Information Systems
Campus Box 1106
Southern Illinois University at Edwardsville
Edwardsville, Illinois 62026

The purpose of this survey is to determine the coverage of microcomputer concepts and applications within courses in the MIS curriculum and to learn what types of instructional materials are being used to support microcomputer instruction. The findings indicate that microcomputer concepts and applications are covered in introductory data processing, MIS, and microcomputer courses. Many of the MIS educators surveyed emphasized the teaching of basic microcomputer skills in MIS courses so that students could use microcomputer software as a tool in courses in other business disciplines.

BACKGROUND

At universities around the country, students are learning how to use microcomputers in business courses. At Harvard, students use spreadsheet software to perform analyses required in Harvard Business School case studies. Using Lotus 1-2-3 and data diskettes with financial and accounting data, students solve cases using the computer as a tool. At Stanford, one basic computer course is used to introduce students to computer-based modeling and decision-making. After that, students use computers which are available in micro labs to solve problems in many courses. Microcomputer uses are also integrated into finance and marketing courses at Wharton School of Business (LaPlante, 1986).

The move toward microcomputers in higher education is widespread. Of the 125 business schools surveyed in the Second Annual UCLA Survey of Business School Computer Usage, 119 had microcomputers for faculty and student use. Microcomputer software most commonly used in instruction included Wordstar for word processing, Lotus 1-2-3 for spreadsheet analysis, dBase II and III for data base applications, and BASIC for programming. For courses in computers and information systems, management science, statistics, and production and operations management, over 75 percent of the respondents indicated computer uses. In graduate level courses in finance, accounting, and marketing, computer uses were prevalent. Ninety-one percent of the undergraduate programs and 75 percent of the graduate programs surveyed required a course in computers and information systems.

In his study of the undergraduate MIS course offered in A.A.C.S.B. schools of business, McLeod surveyed a national sample of 117 faculty to determine the content and instructional methods in this course. There was a broad consensus on what topics should be covered, with the greatest emphasis being placed on systems analysis and systems theory. In 27 (43.5%) of the schools he studied, students were introduced to pre-written software such as word processing or spreadsheet packages, but there was limited hands-on exposure to this software. Even though data base management system concepts were covered by three-fourths of the faculty surveyed, only one-fourth of them reported using a data base package like dBASE II (McLeod, 1985).

Over one-third of the schools McLeod surveyed were using computer-based case problems and required their students to use computers, special data bases, and appropriate software to solve business case problems. The use of statistical packages such as SAS and SPSS was cited by 37 percent of the respondents. It was interesting to note that the use of non computer-based problems was the most commonly used experiential activity in the MIS course, with two-thirds of the respondents using management cases. In his discussion, McLeod points out that if students were trained to use software for word processing, data base, and graphics applications in the MIS course, they could use these tools for case analysis in subsequent course work throughout the business curriculum.

In many schools today, MIS faculty members are playing a service role by integrating microcomputer concepts and applications into introductory courses in data processing concepts and MIS. In this way, students learn how to use software as a problem-solving tool. In addition, microcomputers are being used to teach concepts such as data base design and systems analysis within courses in the MIS curriculum. Using a microcomputer data base to teach relational data base concepts, for example, reinforces basic concepts and provides valuable hands-on experience (Smith, 1985).

Another approach to providing microcomputer instruction is to integrate microcomputer applications into the content area courses in business. Exercises or cases can provide students with opportunities to use software for data base manipulation and analysis in such areas as financial planning and marketing research (Burke, 1985),

OBJECTIVES

The major objective of this study is to determine the coverage of microcomputer concepts and applications within courses in the MIS curriculum. The study looks at the topics covered and software used in the introductory data processing course, the MIS course, and a microcomputer applications course. Instructional materials used to support microcomputer instruction are surveyed. Another objective of the study is to determine whether MIS academics feel that microcomputer instruction should be taught in separate courses, included as a part of courses in the MIS curriculum, or integrated into courses such as finance, marketing, and accounting.

METHODS USED

The population surveyed for this study included all MIS faculty members listed in the 1983 MISRC/McGraw-Hill Directory of Management Information Systems Faculty who indicated that their areas of specialization were IS3, Systems and Information Concepts in Organizations, or CIS10, Decision Support Systems. Faculty members with these areas were selected because they were the ones likely to teach courses in introductory data processing concepts and information systems concepts for managers.

A brief questionnaire was developed and mailed to 107 MIS faculty members. They were asked to estimate the percentage of coverage of microcomputer concepts in MIS courses, the microcomputer topics covered in these courses, and microcomputer software used to teach these concepts and applications.

FINDINGS

The findings are based on the responses of 25 of the 107 MIS educators who received the survey. The respondents represented 18 states and Canada and 25 different university-based MIS programs.

The major focus of the questionnaire was on three courses in the MIS curriculum: the introductory data processing course, the MIS course, and a microcomputer course. All of the educators surveyed identified a specific introductory course and an MIS course. When asked to describe a specific microcomputer course, nine of the respondents listed a specific course title having to do with micros (e.g. Microcomputers in Management etc.) and five others listed courses with titles such as Computer-based Business Systems which were microcomputer-oriented. One respondent noted that microcomputer concepts were taught in non-credit workshops outside the curriculum and another said that microcomputer topics were integrated throughout the curriculum.

The ranges of course coverage devoted to microcomputer topics were widespread. In the introductory data processing course, the approximate percentage of coverage of micro-computer concepts ranged from 0 percent to 80 percent, with an average of 23 percent of the course spent teaching microcomputer topics. The average percentage of coverage of microcomputer topics in the MIS course was 13 percent, with a range of 0 percent to 100 percent.

The respondents were asked to indicate in which course various microcomputer topics were covered. The percentage of respondents indicating that each of the following topics were covered in the introductory data processing course, the MIS course, and a separate microcomputer course is described in Table 1:

	Percentage of Respondents		
Topics:	Intro. to DP	MIS Course	Micro Course
Microcomputer hardware	72%	32%	36%
Microcomputer storage devices	68%	28%	36%
Microcomputer input output devices	64%	40%	32%
Microcomputer operating systems	56%	72%	40%
Utility packages	2%	1%	2%
Spreadsheet applications	56%	40%	40%
Data base applications	56%	36%	36%
Word processing applications	60%	32%	36%
Project management applications	12%	28%	2%
Graphics applications	32%	2%	32%
Accounting applications	29%	2%	2%
BASIC programming	64%	0%	24%
Microcomputer local networks	24%	2%	24%
Decision support systems using micros	28%	32%	24%
Analysis and design of micro systems	12%	36%	2%
Micro-mainframe communications	40%	32%	24%

Table 1: Microcomputer Topics

Most of the respondents indicated that basic microcomputer hardware topics were covered in the introductory data processing course. Word processing applications were covered primarily in the introductory course, as was the case with BASIC programming on microcomputers. About one-half of the respondents indicated that data base and spreadsheet concepts were covered in the introductory course, while an additional one-third of the respondents reported that these applications were covered in the MIS course and in a separate microcomputer course. Project management applications were covered primarily in the MIS course, although only one-third of the respondents indicated that decision support systems using micros was a topic covered in the MIS course.

Other microcomputer topics which were covered by over one-fourth of the respondents in both the introductory data processing course and a microcomputer course were local area networks and micro-mainframe communications. In contrast, one-third of the respondents noted including the analysis and design of microcomputer-based systems in the MIS course, while very few respondents indicated that this topic was included in either the introductory class or the microcomputer class.

The findings indicate that microcomputer hardware concepts are covered both in the introductory course and in the microcomputer course, but that microcomputer applications are covered in every course. It is interesting to note that many of these tools are introduced right at the beginning in the basic data processing course, with supplementary coverage in the areas of data base, spreadsheet analysis, and project management being provided in the MIS course. It seems that this type of coverage is replacing some of the traditional concepts in these courses. Hardware and software topics, which may have been presented as concepts in the past, can now be visualized by students using microcomputers.

SOFTWARE USED

When asked to describe the software packages being used in the introductory data processing, MIS, and microcomputer courses, the overwhelming majority used dBASE II and III for data base applications and Lotus 1-2-3 for spreadsheet and graphics applications. Although generic software for data base and spreadsheet applications was used by several respondents in the introductory course, production software such as dBASE III, RBASE 5000, Multiplan, and Lotus 1-2-3 was used in both the MIS course and microcomputer course. Word processing software was a different story, with a wide range of packages reported. Although Wordstar was the most frequently used package in all three courses, at least five other packages (e.g. Displaywrite III, PCWrite, Writing Assistant, Word Perfect, Multimate) were reported by at least one respondent. There seems to be no consensus on word processing software, although the use of dBASE II and III and Lotus 1-2-3 reflects the widespread acceptance of these packages in industry.

When asked what software packages they would prefer using if there were no constraints on their choice, several of the respondents who were not currently able to use dBASE III or Lotus 1-2-3 indicated that they would prefer these packages over the ones they were currently using. This finding indicates that these schools may be following software standards being established in the business community. At least one respondent out of the 25 indicated an interest in using integrated software such as Symphony and Framework. An interest in using fourth generation tools such as FOCUS was indicated by two of the respondents but was not as yet widespread.

INSTRUCTIONAL MATERIALS

One of the issues confronting MIS educators who are teaching microcomputer concepts and applications is how to find instructional materials that support students learning hands-on skills. If such materials are not available in textbooks, tutorials, and software documentation, instructors are faced with developing these materials themselves. When asked to indicate the instructional materials used to support instruction in each of the application areas, the respondents indicated that textbook and instructor-developed materials were the most commonly used, as indicated in Table 2:

	Instructional Materials Used In:			
	Data Base	Spread sheet	Word Processing	Graphics
On-line tutorials	32%	56%	40%	16%
Textbook materials	64%	72%	68%	44%
Instructor-developed	52%	64%	48%	36%
Print tutorials	2%	2%	12%	1%
Software documentation	48%	48%	36%	28%

Table 2: Percentage of Respondents

The highest percentage of respondents (62%) indicated using textbook materials to support microcomputer instruction for all of the software used. Half of the respondents indicated using instructor-developed materials, and 40 percent used software documentation supplied by the vendor. One-third of the respondents indicated that on-line tutorials were used, and less than five percent of the respondents used print tutorials for microcomputer instruction.

These findings indicate the importance of textbook materials for microcomputer instruction. Because of the lack of these materials, or the lack of materials supporting specific software packages used, many instructors are developing their own tutorials and supplementary materials. Good software documentation and the availability of good textbook materials may determine the software choices of MIS educators in the future.

INSTRUCTIONAL APPROACH

One of the major issues in today's MIS curriculum is where to include microcomputer content. When asked whether they felt microcomputer content should be integrated into MIS courses, included in non-MIS courses, or taught in separate courses or workshops, over half of the respondents felt that microcomputer content could be integrated into the introductory course, the MIS course, or into non-MIS courses, as indicated by Table 3:

Method Used:	Percentage Responding:
Integrated into Intro to DP	68%
Integrated into MIS course	56%
Integrated into non-MIS course	52%
Taught in a separate course	2%
Taught outside of course work	6%

Table 3: Method of Teaching Microcomputer Content

Most of the respondents who felt that microcomputer concepts should be integrated into the Intro. to DP course and the MIS course in the information systems curriculum felt that the basics could be covered in these courses so that students could apply these tools to problems in content area courses. MIS instructors felt they were performing a service role to other departments within their business schools by teaching microcomputer applications. These skills, they felt, could be applied in the content areas. In addition, one respondent noted, microcomputer software such as data base management systems could be used to demonstrate complex concepts which previously may have been less clear because of the reliance on mainframe-based tools.

Several respondents felt that microcomputer applications should be taught within non-MIS courses to avoid a specialized approach and to tie these tools to the application context within which they would be used. By integrating skills training into non-MIS courses, they felt, non-MIS faculty members would have an opportunity to benefit from microcomputer exposure.

Those favoring the teaching of microcomputer concepts in a separate course suggested that this would eliminate the duplication of effort involved in teaching microcomputer applications in many MIS and non-MIS courses. At Harvard, for example, MBA's are required to take workshops outside of regular classes to become familiar with microcomputer tools. These tools are used primarily for problem solving and modeling in required case study analysis in the business curriculum.

EQUIPMENT USED

Of the 25 MIS educators surveyed, 22 noted that IBM or IBM compatible software was used to support microcomputer instruction, and three indicated that Apple microcomputers were being used. Much of the software selected for business school computing ran in an IBM-compatible environment. The standards that industry is setting for hardware and software may be having an impact on the choices business schools are making.

CONCLUSIONS

In general, the microcomputer has had a major impact on MIS instruction. Microcomputer concepts and applications are covered in introductory data processing, MIS, and microcomputer courses. Many of the MIS educators surveyed stressed the idea of teaching basic skills in MIS courses so that students could apply the tools they learned in courses in finance, marketing, and accounting.

The instructional materials MIS faculty members rely upon most to teach microcomputer concepts in MIS courses are textbook materials and instructor-developed materials. It would seem that textbook publishers with good microcomputer materials integrated into both introductory data processing and MIS texts may provide teachers with the materials they need. Because of software preferences and local school needs, however, instructor-developed materials may continue to be developed.

The topics covered in this study provide opportunities for further research in how microcomputer concepts can be successfully offered. Today, there is a general infusion of microcomputer topics throughout the curriculum, which may cause some redundancy. Eventually, as students master microcomputer skills in basic courses, more advanced courses in the MIS program can move away from basic skills training approaches and concentrate on microcomputer application development using data base software and programming languages.

REFERENCES

Burke, Thomas, "Developing Micro-Based Courses," Proceedings of the Fourth Annual Information Systems Education Conference, Houston, 1985, pp. 26-28.

Frand, Jason L. and McClean, Ephraim R., "Summary of the Second Annual UCLA Survey of Business School Computer Usage," Communications of the ACM, Vol. 29, No.3, January 1986, pp. 12-18.

LaPlante, Alice, "Micros Earn their MBA's," InfoWorld February 3, 1986, pp. 24-27.

McCleod, Ray, Jr., "The Undergraduate MIS Course in A.A.C.S.B. Schools," Journal of Management Information Systems, V. II, No. 2, Fall 1985, pp. 73-85.

Russell, David L., "Using a Microcomputer to Teach Relational Database Theory and Practice," Proceedings of the Fourth Annual Information Systems Education Conference, Houston, 1985, pp. 102-103.

Smith, Wayne D., The Role of Microcomputers in the CIS Curriculum, Proceedings of the Fourth Annual Information Systems Education Conference, Houston, 1985, pp. 24-25.

STRUCTURED PROGRAMMING IN BASIC ON THE IBM PC: A GUIDE TO TEXT SELECTION AND AUTHORING

Eli Boyd Cohen
Elizabeth Cohen Boyd

Department of Management Information Science
California State University, Sacramento
6000 J Street, Sacramento, CA 95819-2694
(916) 427-7000
TRACK: How to Teach CIS Courses

ABSTRACT

Many colleges select IBM PC BASIC for their introductory language to teach business students about programming. Currently there are many texts available for adoption. These authors developed a grid of desired features to rate the available texts. No reviewed text satisfied all our our adoption criteria.

This paper presents these adoption criteria with the hope that they may prove helpful for other adopters. Also, we hope that authors of IBM PC Business BASIC texts will consider these criteria when writing or revising their text.

PROBLEM WITH MOST TEXTS

Teachers of college **Introduction to Management Information Science** and **Business BASIC** courses have a problem finding an adequate text. Currently, there are dozens of potential texts to review. This problem grows with the publication of new BASIC texts each year.

Within this growing segment of the BASIC market, no single IBM PC Business BASIC text dominates. We believe that the lack of market leadership is due to the text meeting the common needs of instructors of Business BASIC.

The most common complaints about the existing texts fall into one of these categories.

(1) The texts are either dull, or unduly spirited, lacking in business focus.

(2) The texts lack full explanations.

(3) Programs and program segments in the texts have many bugs.

(4) The text fails to teach programming style.

(5) The text fails to teach structured methodology.

(6) The limitations of the student are not properly addressed.

(a) The student who has no or little prior knowledge of structured techniques. They may even have learned bad programming habits from prior (high school) courses.

(b) The student who is a business major and yet has a limited knowledge of business. This means that while a business focus may be in order, the text must limit its business applications to those familiar to all.

(c) The student who has limited motivation to learn this subject. They are not declared computer science or MIS majors. Indeed, many have learned to fear computers.

The remainder of the paper provides details on these points. It provides specifications for a potential text adopter and for prospective authors on what should be in a Business BASIC text.

SPECIFICATIONS FOR THE IDEAL IBM PC BUSINESS BASIC TEXT

Three areas characterize what we view as important in an IBM PC Business BASIC text. These areas are topical coverage, pedagogy, and organization.

TOPICAL COVERAGE (WHAT IS COVERED)

The text must be comprehensive in its coverage of BASIC, structured programming, systems of programs, and the specifics of the IBM PC BASIC.

A. Comprehensive Coverage of BASIC

Some texts omit fundamental programming concepts (such as the file handling) to justify simplicity. We would like to see BASIC carried as far as random file access, and include auxiliary topics, such as business graphics.

The course for which the text is being adopted may not and need not teach all the concepts included in the text. The student benefits none the less from reading the BASIC has features beyond what the current course teaches. Further, some students may learn these features on their own, or be inspired to take an advanced course.

B. Teach Structured Programming

Many BASIC texts fail to teach the student how to program. Students need to learn not only the syntax of a language, but also the semantics of programming (good style).

In structured techniques we include the following:

(1) structured program development, including the use of design tools,
(2) structured testing, including top down testing, and how to develop one's own test data,

(3) debugging, including common mistakes and how to correct them,

(4) data dictionary creation and maintenance,

(5) Documentation. Students must learn how to document (the techniques of documentation, what to document, in line and external documentation). More specifically, they must learn WHY to document their programs.

3. Systems of Programs

Systems of programs (which use common files) points out the needs for standards and the need for systems' documentation.

4. IBM PC Specifics

The student will be working with the IBM PC. Consequently, they need a discussion of IBM PC specific features. Features requiring discussion include the disk operating system, and the physical layout of PC, including the keyboard.

Since the text should limit its coverage to the BASIC of the IBM PC, it should include coverage of the special features of that BASIC and of the IBM PC Computer. For example, the PC allows on-screen editing. This concept, difficult to teach but easy to use, needs exploration in the text.

Teach Business

In teaching the use of personal computers in business, the text can expose the student to how business uses personal computer (and not just how business uses BASIC). That is, the text should describe various scenarios in which personal computers are currently being used in the business environment.

GOOD PEDAGOGY (HOW IT IS COVERED)

Just as important as good topical coverage is good pedagogy. Foremost in good pedagogy is clarity of explanations. Techniques which enhance clarity include the use of many examples, provide illustrations which explain, and teach features step by step.

Since the text is to teach structured techniques in addition to syntax, the examples used must all show good style.

Problem Solving Orientation

A problem solving orientation helps the student to identify with the problem and helps maintain interest. Here the author moves from topic to topic based on the need to solve a problem. Since this text should stress business problems, the applications should relate to the business world.

The approach then would require the author to give reasons for structured style and reasons for not using statements such as GO TO.

Deal with the Students' Fear of Computers

The desired text should acknowledge and deal with the fear of many students about computers. Some ways in which this can be accomplished include the following:

(1) A gentle introduction, to put the student at ease.

Current tests in other fields have introduced the idea of including stories as part of the text.

(2) Perhaps include characters in these stories with which the student can identify.

(3) Let the student see the characters make mistakes, so they do not set unrealistic expectations for their own programming.

Teaching on a microcomputer, such as the IBM PC, provides a special learning opportunity.

Personal computers are more than just interactive. Interactive mainframes and minicomputers provide interactive computing. We have observed that using personal computers, students are willing to take greater ownership of their computer's actions than with timesharing.

ORGANIZATION

The third set of specifications for reviewers and authors writing BASIC texts is in the area of organization.

Of course, the topics must progress logically, from subject to subject, level to level. The text should, at its start, give a sense of what is BASIC, the IBM PC, structured techniques, and the programming environment of IBM PC BASIC. It must accomplish all this without intimidating the fearful student. Then it must, if it is to be complete, go into much greater detail on these and other topics.

One technique we advise text reviewers to look for is the use of **advanced organizers.** Advanced organizers are those means by which the student gets an idea on which is being taught before the student reads about the topic. Heading titles that describe the topic being taught are helpful. For example, the topic heading SAVE **puts a copy of your program on disk** is an example of such an advanced organizer.

Quick reference guides also help the student use the text both for instruction and for reference.

In reviewing texts, pay careful attention to exercises. Good exercises are at varied levels of difficulty, and tap a variety of types of learning. For example, some exercises might ask the student what would happen if a particular program were run. Others would ask them to write or modify a program.

The author of a BASIC text need not follow the traditional path. For example, we teach READ...DATA only after the student has a firm grasp of the concept of files. This avoids the inherent confusion many students have between programs and data. Consequently, we keep an open mind when an author's organization of topic differs from the conventional.

SUMMARY

Current tests are insufficient for teaching structured programming in BASIC on the IBM PC. These authors have enumerated features which they believe should be included in such a text. The features fall into the categories of topical coverage, pedagogy, and organization.

We hope that future authors of BASIC texts will follow these specifications.

USING MICROCOMPUTERS TO DOWNLOAD CICS PROGRAMMING LOGIC

Andrew M. Suhy

Associate Professor
Department of Computer Information Systems
Ferris State College
Big Rapids, Michigan 49307

ABSTRACT

On-line programming is rapidly changing the way data processing needs are being met by increasing numbers of businesses. Yet the hardware and software support needed to teach mainframe on-line programming skills are often beyond the financial reach of many colleges. This paper describes an approach that permits the teaching of mainframe on-line programming at colleges that lack extensive resources.

INTRODUCTION

One of the most sought after areas of computer expertise by large corporations is training in on-line programming. While batch programmers will still be in demand to maintain existing programs, an increasing percentage of applications is going on-line. This demand is often manifested in tele-processing software known as CICS. CICS allows traditional programming languages such as COBOL, PL/1, and BAL to be run on-line rather than in batch mode. With the sharp sharp increase in preference for interactive terminals over punched cards, it appears that the trend toward on-line programming will continue to grow for the forseeable future.

UNIQUE PROBLEMS

Campus recruiters stress repeatedly the importance of on-line programming experience, notably CICS and COBOL, yet few universities currently offer this valuable training. A major reason is the onerous expense involved in the purchase of a powerful main-frame and the software support needed to run CICS. In addition, because of the nature of CICS a well-trained support staff is often needed in the academic computer center. This is because CICS requires a special programming logic that involves the manipulation of carefully controlled loops that can easily become infinite loops. When this occurs, the entire system often crashes. This is sharply different from running batch programs where almost all student programming errors are handled easily without having a major impact on the rest of the operating system. With most student programming mishaps even the most extensive errors will usually generate only a series of error messages while leaving the rest of the system intact.

FINANCIAL AND TECHNICAL BARRIERS

Because of the increased hardware and support staff costs, many small and medium sized colleges are unable to afford a full implementation of CICS and consequently have not taught CICS or other popular mainframe on-line programming concepts, and have instead reverted to batch programming techniques or limited themselves to simplified microcomputer packages and applications. Even colleges that can afford CICS often hesitate to offer many sections of CICS because of a reluctance to tie up valuable mainframe resources or a fear of having the computer systems fail because of a runaway loop or an extrapartition access.

POSSIBLE SOLUTIONS

Despite these barriers, it is possible to teach on-line programming concepts as required by CICS or other on-line support software with only limited computer resources. Ideally, a college should have access to a mainframe with CICS, but it is also possible to teach on-line programming using the resources of even the poorest computer department. While any concept can be taught abstractly, hands-on experience requires some type of computer although not necessarily a mainframe. Since most computer departments have access to at least a few microcomputers this is not a problem. Furthermore, under the approach I am advocating BASIC can be used as the learning conduit. This has many advantages.

ADVANTAGES

First, since any microcomputer can be used, this program of study is within the reach of virtually all computer departments. It is not even necessary to own an IBM PC to be able to develop programming concepts that can be used on an IBM product on an IBM mainframe. This means that departments that have invested heavily in Apple, Commodore, Radio Shack or DEC can all use this approach with similar efficacy. Indeed, in some ways the Apple IIe is better suited for demonstrating IBM concepts than the IBM PC.

Second, since BASIC is used,there are absolutely no additional software costs since BASIC is included in the overall purchase price of most microcomputers. The version of BASIC used is essentially irrelevant. This factor alone could save a computer department many thousands of dollars in software expenditures.

Third, BASIC is particularly well-suited for on-line programming. Unlike many other programming languages, BASIC consists almost entirely of executable instructions. Few, if any, instructions are needed for data definition--there are no Environment Divisions, File Control Sections, types, or sophisticated data structures. Because BASIC has one of the loosest syntax requirements of the popular programming languages, it may not be the best language to teach structured batch programming concepts. Yet, it it precisely this flexibility and lack of structure that facilitates the teaching of CICS on-line programming concepts. In a CICS program there are multiple entry and exit points as well as forced GO TO's that are generated automatically by the system. Using standard pseudoconversational programming techniques which allow a single program to be used by many terminals, a CICS program is terminated every time a screen is sent to a terminal. Execution of a program can be initiated or terminated at many points through the use of control key options or program function keys that are selected by the user. This was incorporated into CICS to allow for the somewhat unpredictable nature of on-line man-machine interaction.

Much of the predictability found in batch programming applications is absent in on-line programming. Records can be entered in any order, some are edited some are not, options can be changed in midstream, the user must always be allowed to see the beginning menu and to alter or terminate processing at virtually any point. Every invocation of a HANDLE AID or HANDLE ABEND automatically generates GO TO statements in the application program--this generation is not under the control of the programmer. This, in turn, wreaks havoc on the traditional concepts of structured batch programming. Consequently, it is difficult to conceptualize, let alone write a program using CICS logic in a heavily structured language such as Pascal.

Because of its sheer simplicity, BASIC has a syntax that even students with minimal exposure to programming will find to be almost transparent. Instead of struggling with the syntax of CICS, students are able to concentrate more fully on pseudo-conversational programming techniques and screen design.

Through the use of LOCATE commands on the IBM PC or the VTAB, HTAB, and POKE commands on other microcomputers, it is possible to establish full screen menus with multiple line choices similar to those used by CICS. By using random files in BASIC, it is possible to create systems for data capture, inquiry, updating, and browsing. In CICS this would require many separate steps.

Students frequently have difficulty grasping the connection between the way multiple seemingly identical screens are presented to the user and the program logic that sends and receives these screens. By having the screen layout expressed and defined in BASIC within the program itself rather than having the screen defined one way in a MAPSET and then brought into an application program in a substantially altered form, it is much easier for students to grasp the screen/program interface.

A major conceptual stumbling block is separating input data from output data when the display screens are identical. In CICS this distinction is crucial, yet is often buried inside of a macro or EXEC CICS instruction. In BASIC, however, it is not possible to input or output data without explicit and different instructions. Thus, while the screen may appear to be identical whether input data or output data is on the screen, it is clear from the program logic that a PRINT statement produced the output and an INPUT statement produced the input.

This feature aids greatly in clarifying the concepts of SEND MAP/ SEND MAP DATAONLY SEND MAP MAPONLY and the RECEIVE MAP family of commands. Thus, the two major conceptual difficulties facing students learning CICS, namely, deeply nested loops with multiple entry and exit points and the sending and receiving of full screen maps can be explained in great depth without using a mainframe or having access to CICS.

SUMMARY

Downloading CICS logic to microcomputers has numerous advantages. First, hardware investment is minimal and software costs are virtually non-existent. Second, once students understand the techniques of interactive programming and screen design, it is then fairly easy to learn the syntax of CICS or another on-line software support package. Third, changes can be made very rapidly using this micro model. It is easier and much quicker to recompile a BASIC program than it is to compile a mapset, then update the CICS mapset tables, then compile the application program, then update the CICS program tables, and then finally execute one test. With BASIC all five steps are combined into one and can be executed very rapidly thus affording almost immediate feedback which is especially critical in designing interactive systems. Fourth, because mistakes made on microcomputers will affect only that micro, the danger of bringing down an entire computer system is eliminated. Very expensive mainframe time is also freed up because testing is performed at the micro level.

Using this approach, it is possible to teach valuable and highly marketable skills needed for full screen on-line programming on powerful mainframes even if an institution does not have access to a mainframe. Thus, a very powerful skill is within the reach of virtually any computer department at minimal cost.

REFERENCES

1. Adams, David R. and Leigh, William E. Programming Business Systems with BASIC. Cincinnati: Southwestern Publishing Co. 1984.

2. Customer Information Control Systems/ Virtual Storage (CICS/VS) Version 1 Release 5 Application Programmers Reference Manual (Command Level). White Plains: IBM Data Processing Division. 1981.

3. Customer Information Control Systems/ Virtual Storage (CICS/VS) Version 1 Release 5 General Information. White Plains: IBM Data Processing Division. 1979.

4. Jatich, Alida. CICS Command Level Programming. New York: John Wiley and Sons Inc. 1985.

5. Kacmar, Charles J. On-Line Systems Design and Implementation Using COBOL and Command Level CICS. Reston: Reston Publ. Co. 1984.

6. Lim, Pacifico Amarga. CICS/VS Command Level with ANS COBOL Examples. Van Nostrand Reinhold Co. 1982.

7. Lowe, Doug. CICS for the COBOL Programmer. Fresno: Mike Murach and Associates. 1984.

THE PROGRAM DEVELOPMENT LIFE CYCLE
What should the first course emphasize?

Lloyd R. Weaver, CDP
Assistant Professor
Department of Computer Technology
Purdue University

ABSTRACT

To equip today's undergraduate student with the fundamental skills necessary for business data processing, a foundation in structured programming, modularization and debugging is required. Many students, teachers and authors are still not using correct structured programming techniques though most programmers claim to be proponents of structure. Few students have been assigned complex enough tasks to truly require modularization into subtasks and have not learned the techniques required for this skill. In addition, the amount of debugging is usually limited in a small program assignment. These problems can be rectified by shifting the emphasis in the fundamentals courses from just coding the program to a complete program development life cycle and by assigning more complex business algorithms requiring true modularization.

INTRODUCTION

The first course in business data processing must emphasize the restricted control structures that are the building blocks of a structured program. The course must project assignments large and complex enough to require decomposition into workable modules. Many first courses concentrate on a language and the coding or implementation of a program. This practice must be redirected to downplay the implementation and emphasize the analysis and planning tools and methodologies. These tools and methods are collectively called the program development life cycle. By using this cycle, the student will complete projects faster and will develop more consistent, better quality software.

PROGRAMMING DEVELOPMENT LIFE CYCLE

Much has been printed concerning the business information system life cycle. The programmer can learn much from this engineering based methodology. As the information system has a life cycle, so should each program element of the system. Through time, program requirements change due to external stimuli (i.e. business laws, tax rates, etc). Thus, modification of existing software is necessary and is perhaps the most costly element of data processing. The overall cost of a program through its lifetime can be significantly reduced if the original program is developed to facilitate this need for modification.

When one considers the relatively short period of a program's lifetime spent in initial construction, every effort must be made to ease the process of modification. This requires planning, proper structure and good programming style.

Many programs are written without prior planning. This is possible because of the simple nature of most course assignments. When the student graduates and is faced with the reality of large business systems, the student is unable to cope with the complexity.

For the initial construction and for every major modification, the following phases must be completed.

(1) Analysis of the design specifications
(2) Design of the algorithm logic
(3) Generation of test data
(4) Design walkthrough
(5) Design implementation
(6) Program walkthrough
(7) Testing and debug
(8) Most of the phases result in documentation as a by-product. Documentation is <u>not</u> a separate phase. It should be noted that these phases are iterative. If a problem develops in one phase, the programmer must, frequently, return to a previous phase to correct the problem.

Figure 1 depicts the program development life cycle:

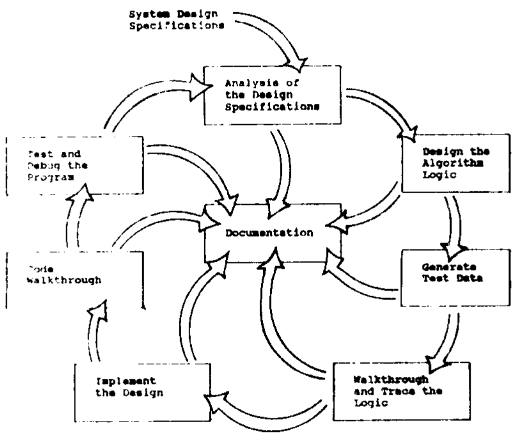

System Design Specifications
Analysis of the Design Specifications
Design the Algorithm Logic
Generate Test Data
Walkthrough and Trace the Logic
Implement the Design
Code Walkthrough
Test and Debug the Program
Documentation

Figure 1.
Program Development Life Cycle

It is necessary to look at the relative importance of each phase in this cycle. Most programming courses concentrate 90% of the instruction in the implementation phase when in reality, it is the least important phase. Consider the building of a house. The actual construction is like the coding of the program. Imagine trying to build a house without an architect's plans: walls going up and being painted before the electrical, plumbing and insulation was installed, etc. This would require tearing down the walls for the installation and the additional costs for reinstallation of new materials. Coding the program without a logic plan results in similar additional costs and in a lower quality product. The first four phases of the program development life cycle should be the emphasis for the first programming course.

ANALYZE THE DESIGN SPECIFICATIONS

All programs have three general components: Input, Processing and Output. To analyze the requirements of the algorithm, it is beneficial to first study the output or informational requirements and coordinate this study with analysis of the input data. Ensure each output has an input source. For those output elements not directly supported by input data, determine that first, the processing requirements are known and secondly, that each process is supported by all required input data. If the data is to come from storage files, become comfortable with the record structure of each file.

As the specifications are analyzed, formulate a "things-to-do" list. Without concern for hierarchical order, identify the main tasks required to complete the algorithm. Continue to decompose each main task into subtasks based on function until each module displays the following characteristics:

(1) implements a single independent function
(2) contains a single entry and exit point
(3) is able to be separately coded and tested
(4) is transferable to other programs.

When all tasks and subtasks have been identified, determine the hierarchical relationship among the tasks. Unlike the traditional flowchart, major tasks go <u>above</u> their component subtasks.

Figure 2 provides a general form for the hierarchy chart.

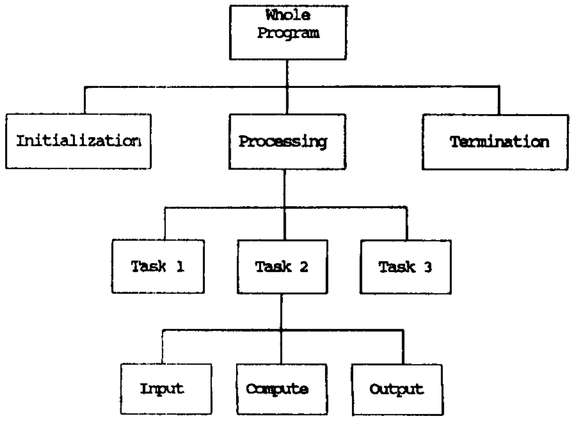


Figure 2.
Generic Hierarchy Chart

DESIGN OF THE ALGORITHM LOGIC

There has been much debate over the tools used in program logic design. Advocates of the symbolic or geometric tools such as the traditional flowchart or more recent structure charts argue with the advocates of the nongeometric tools such as pseudocode. Students exposed to structure charts and pseudocode seem to prefer the geometric tool initially, but after their design skills have improved, find the pseudocode easier to document (especially with word processors) and easier to translate into a programming language.

Regardless of the tool used, the designer must develop good programming habits with structured programming, proper use of modularization and style considerations such as meaningful variables, standardized indentation, etc.

A structured program is an algorithm consisting of one or more restricted control structures, each of which has a single entry and single exit. There are three forms of restricted control:

(1) The Sequential Control Structure has a straight line appearance. Each instruc-

tion is executed following its immediate predecessor.

(2) The Conditional Control Structure allows the program to make a binary decision based upon a condition or multiple conditions. Care must be taken to ensure the single entry, single exit requirement of the structure.

(3) The Repetitive Control Structure is used to iterate the same instructions multiple times. Most languages have two forms of this structure. One is used when the number of iterations can be predetermined and the other is used when the number of iterations is unknown. It is in this control structure that the most common violation of the single entry/exit premise of structured programming occurs.

Consider the following example: An unknown number of customer records are on a customer file. Each record consists of the account number, name and account balance. Access all records sequentially, printing the customer name and balance to a report.

Sum the balances and print the total under the balance column. An account number of 0000 marks the end of the file.

Analysis of the algorithm indicates a need to open the customer file, initialize the total, access a record, determine if the end of the file has been reached, sum the balance and print the record. When all records have been accessed, print the total and close the file.

Using a flowchart geometric tool as shown in figure 3, it is easy to violate structured programming:

Notice the repetitive control structure formed by the if/goto combination has two entry points. This is a violation of structured programming.

Using the same flowchart geometric tool, the algorithm can be designed without the violation. This is shown by the same example in figure 4. Note the single entry and exit on the repetitive control structure formed by the if/goto combination.

A better geometric tool is the structure chart which recognizes the repetitive control structure as a REPEAT WHILE. The Repeat While and its corresponding End Repeat clause helps to ensure the single entry and exit design of the control structure. Figure 5 shows the example using a structure chart.

The use of pseudocode as a nongeometric design tool has the advantage of being more easily modified on a word processor and takes significantly less space. Some first course students seem to need the geometric

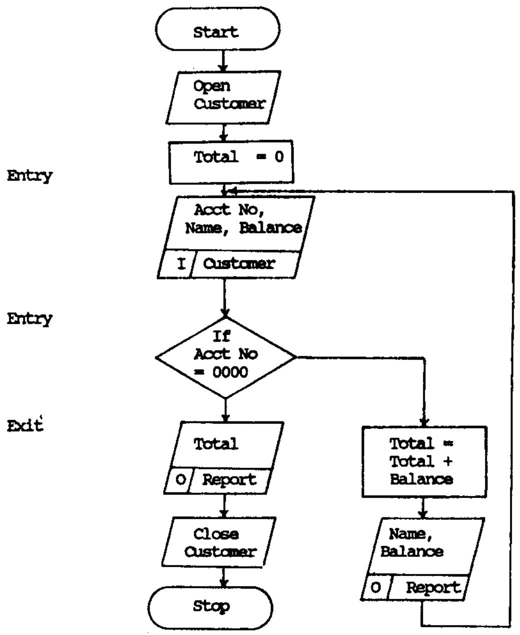


Figure 3.
Flowchart Showing Violation of Structured Programming

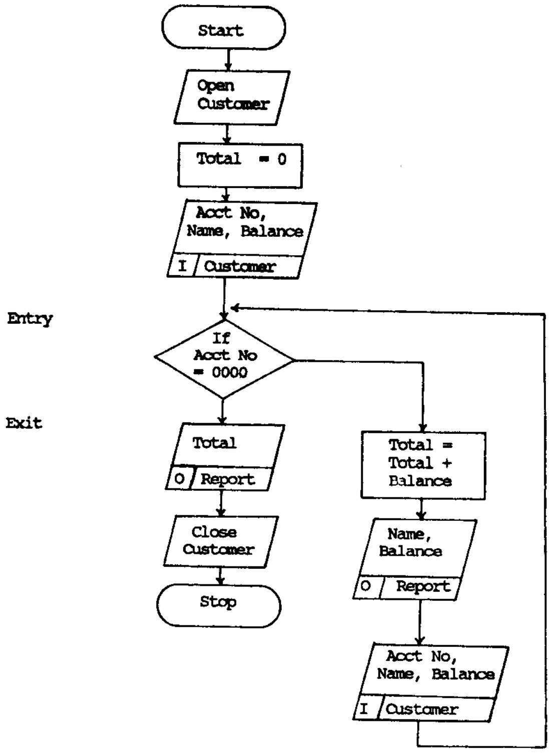


Figure 4.
Example Showing Structured Programming

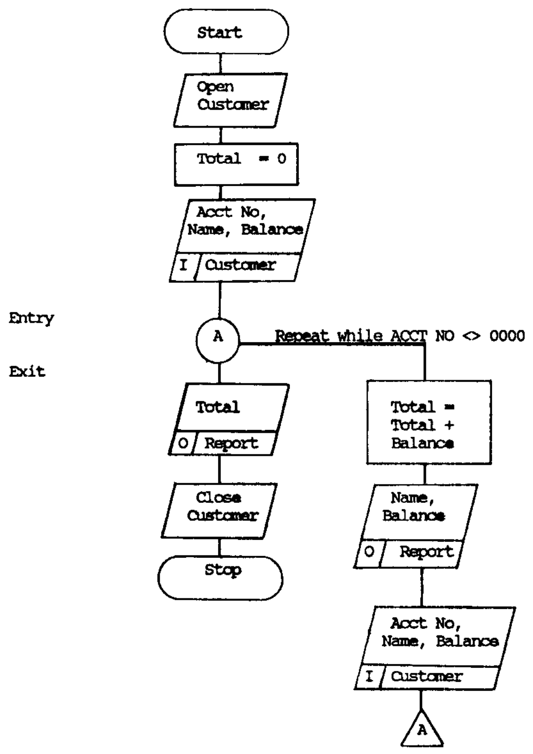


Figure 5.
The Structure Chart Helps to Avoid Structured Programming Violations.

symbology to formulate the logic while others prefer the nongeometric pseudocode. Much research will be necessary before a definitive statement can be made as to the suitability of a particular design tool for the first course. Pseudocode for the example is shown as figure 6.

```
Begin CUSTOMER REPORT module

  Open CUSTOMER FILE for input
  Define CUSTOMER RECORD as CUST ACCT NO  (A,4)
                            CUST NAME     (A,30)
                            BALANCE       (R,4.2)
  Open CUSTOMER REPORT for output
  Set TOTAL to 0
  Access CUSTOMER RECORD from CUSTOMER FILE

  Repeat while CUST ACCT NO <> 0000
    Set TOTAL to TOTAL + BALANCE
    Print CUST NAME, BALANCE to CUSTOMER REPORT
    Access CUSTOMER RECORD from CUSTOMER FILE
  End repeat

  Print TOTAL to CUSTOMER REPORT
  Close CUSTOMER FILE, CUSTOMER REPORT

End CUSTOMER REPORT module
```

Figure 6.
Pseudocode Design Tool

Design tool selection could be left to personal choice, but the algorithm must be constructed using only the three control structures, each restricted by single entry and exit.

As an exercise in formal system's documentation, it is also appropriate to include with each module design, a statement of purpose or description of the module, what data is required by the module, what information is produced by the module and references to additional modules being processed.

TEST DATA GENERATION

To completely test an algorithm, it may be necessary to generate several sets of test data. The end result, however, must be to thoroughly test each branch of each conditional control structure and test each repetitive control through at least two iterations. There are frequently minimum and/or maximum test values associated with any condition. These values may be called the bookends. It is important to test the bookends as well as a median value, since the processing requirements frequently differ in each case. Most first courses in data processing assume edited data which require little error detection. It is appropriate, in the latter assignments, to introduce the concepts necessary to detect and manage data entry errors. The emphasis is to introduce the concepts of error detection and eliminate the false security of "perfect" data. In general, make the test data as close to real production data as the academic climate will allow.

Two levels of test data generation are appropriate for the first course. The first level is the test plan. The test plan contains no actual data values, but is a prose description of what the test set should contain. The second level is formal test data in the form in which the data will be input into the program. Instructors often provide the test data in this second form without requiring students to plan any requirements for testing their algorithms. A more thorough understanding of the algorithm is necessary to plan its testing, thus the addition of at least the first level test plan is recommended. To thoroughly walkthrough the algorithm is facilitated by actual data, so the second level test set is additionally recommended.

THE LOGIC WALKTHROUGH

Just as ensuring that the blueprint will build a quality house, it is important to prove that the logic design will produce a quality program. The logic walkthrough is the most overlooked, yet most important phase in the program development life cycle. Complex programs requiring modularization often contain logic errors, particularly in the interfaces between modules. If these errors are detected **before** the implementation begins, the program will

(1) take far less time to implement,
(2) be of better quality since patchwork fixes often destroy the original flow of the logic,
(3) require reduced utilization of scarce and costly resources (CPU, terminal, drives, etc.) and
(4) be less costly to initially implement and, because of the increase in quality, be less costly to modify over its lifetime.

The simplest method of logic walkthrough is the trace technique. This technique requires the student to put a box on paper for each data dictionary element (variable) in the program and chronologically record the contents of the data dictionary element each time the value changes. Using the test data set(s) designed in the testing section, the student processes his design by hand in the same manner that the computer would process the corresponding program. As each statement is processed, the dictionary elements affected would be recorded as new values in their respective boxes. As output statements are processed, the student prints the last value in any referenced box to a printer spacing chart, simulating the printed reports with actual test data. Later, when the design has been implemented, the test data can be used with the program and the output compared to that on the printer spacing chart to test for logical accuracy.

Even though a trace as described above may take up to four hours to complete, the total time spent on the project from analysis to completed product is significantly less than it would be with the implement and repair cycle employed by some students. Figure 7 illustrates the mean time spent in each activity given two equal groups of students working on the same project. Group 1 employes the program development life cycle and group 2 goes straight to the implementation phase after reading the specification. Note that, even though there are more phases in the program development life cycle, the total time spent from analysis to project completion is reduced.

Group 1: Program Development Life Cycle	Mean Hrs	Group 2: Implement and Repair Cycle	Mean Hrs
Analysis:	2.6	Analysis:	2.4
Logic Development:	13.8		
Test Data Generation:	.5		
Logic Walkthrough:	2.5		
Implementation:	11.8	Implementation:	16.8
Test and Debug:	.5	Test and Debug:	20.1
Total:	31.7	Total:	39.3
Total number of runs to produce a correct program:	4	Total number of runs to produce a correct program:	17

Figure 7.
Comparison of A Program Developed Under the Program Development Life Cycle Versus the Implement and Repair Cycle

Add to this total time productivity, the improved quality of the planned programs, and a clear case for a proper program development life cycle becomes evident.

This life cycle approach is necessary to solve the complex problems which are the norm in business data processing. It is important that educators provide their students with the a productive methodology and the fundamental tools to modularize this complexity and produce high quality structured programs.

STUDENT PROFILES AND SUGGESTIONS FOR IMPROVEMENT IN THE INTRODUCTORY COMPUTER CENTER

Geoffry S. Howard, V. Michael Lahey, Catherine M. Murphy, and Glenn N. Thomas, Kent State University

ABSTRACT

This study surveyed a large number of students in a typical introduction to computers course to obtain a profile of the students along demographic, psychological, computer ability, and course expectation dimensions. Analysis of the results shows that there are compelling reasons for offering three separate versions of the course: one for high computer anxiety clients, one for those with high computer literacy and substantial computer experience, and one for the majority of students who fall between these two extremes. Descriptive data about introductory computer students is provided in a form that will be useful to educators in course content and teaching methods.

INTRODUCTION

Special attention should be devoted to the "Introduction to Computers" course. This critically important course is taught in nearly every institute of higher education and involves a very large number of students and faculty. It has a major influence on students' lifelong attitudes about the usefulness of computers and their impact on organizations and society. Many students make decisions about pursuing a computer career based largely on experiences in this course. At a more practical level, the quality of the computer skills and computer knowledge imparted, will affect students' later success due to the increasing use of computers in all areas of study.

It is troubling to discover that substantial numbers of educators experience difficulties designing and teaching this crucially important course. In a panel discussion on information systems curricula at the Sixth Annual International Conference on Information Systems (4), curriculum experts such as Nunamaker and Davis reported that they receive far more inquiries about how to design and teach the introductory course than about any other course in the computer and information systems curriculum. Reports of faculty and student dissatisfaction are widespread.

Educators can improve the introductory computer course by studying the students. Course content can be matched to complement the levels of computer skills and knowledge of entering students. Special sections can be designed for subsets of the student population with unique problems or abilities. A differential approach can be adopted for those who are taking the course as an elective versus as a requirement.

This study of a large sample of students in a course entitled "Introduction to Computers and Data Processing" provides inputs that can guide educators in correctly making course administration and design decisions. The following student profiles and interpretation are used to generate specific suggestions for course improvement.

1. Demographic Profile--Who is taking the course?
2. Psychological Profile--What kinds of students are in the course?
3. Rationale Profile--Why are students taking the course?
4. Knowledge Profile--What computer skills and knowledge do students already possess when they come to the course?
5. Expectations Profile--What computer skill and knowledge outcomes do students expect to result from their participation in the course?

METHODOLOGY

A consolidated questionnaire consisting of established psychometric instruments and demographic questions was administered to two large sections of the introductory course. Table 1 summarizes the demographic, psychological, and other variables measured by the questionnaire and cites, where appropriate, the sources of the instruments.

Students were not told in advance that the questionnaire would be given. It was explained that their responses would be used to guide redesign of the introductory computer course at Kent State and possibly at other universities. The questionnaire was not administered by the faculty member teaching the course. Students were assured that their responses would not effect their grade in the course, and that the instructor would not see individual responses until after the end of the term. A total of 239 usable responses were obtained.

ANALYSIS AND RESULTS

The demographic profile of the students appears in Table 2. These demographic data contain few surprises and are provided to allow other researchers to use, replicate, or extend the results of this study. It is important to note that the course services students from a diversity of colleges and majors. This complicates the instructor's problem of trying to generate examples and assignments with broad relevance to the class. The most surprising finding is that 17% of the students own a computer or a terminal. This implies that substantial numbers of students are coming to the course with considerable hands-on experience.

Table 3 summarizes the psychological profile of the students surveyed. Computer anxiety has been found to be a clearly identifiable and measurable construct (8,11,5,6), defined as "Fear of the impending interaction with a computer that is disproportionate to the threat presented by the computer." Math anxiety is similar in nature to computer anxiety. Locus of control is a psychological construct that classifies individuals along a spectrum ranging from "External" to "Internal" (10). Externals see the forces that control their lives, such as luck and fate, as outside themselves, whereas internals believe that they can project their will on people and things and can thus control events. Cognitive style ranges from "Analytical" to "Heuristic," and has been widely discussed in the literature on human- computer interaction (12,2,7). Trait anxiety simply refers to a person's tendency to be anxious. This characteristic has been found to remain fairly stable throughout a person's lifetime (9).

A computer anxiety score of 20 or above has been considered in previous studies (8,5) to represent a seriously high level of computer anxiety. Of the students in the present sample, 18.9% are in this high anxiety category. Howard, Murphy, and Thomas (6) found, in a pre-post field study, that not all such high anxiety students will experience a significant drop in anxiety as a result of taking an introduction to computers course. Indeed, some students experience an increase in anxiety from the course, indicating that contact with computers may aggravate an already serious fear instead of alleviating it. The authors cited, argue that a separate section of the course should be taught for such students in order to help them overcome their negative feelings and to learn to use computers effectively.

A better understanding of the origins and implications of computer anxiety can be developed by identifying variables likely to be related to computer anxiety, measuring those variables, and testing for significant correlations with computer anxiety. This was done for the psychological and most of the demographic variables shown in Table 1. The results of the analysis appear in Table 4. These results indicate that highly computer anxious students expect that computers will impact society unfavorably. Math anxiety and trait anxiety go hand in hand with computer anxiety (as expected), external locus of control types are slightly more computer anxious than internals, and age and years of work experience both vary inversely with computer anxiety. Surprisingly, there is no cognitive style effect. Females are a bit more computer anxious than males at the beginning of the introductory course.

These results provide psychological and demographic profiles of the computer anxious students in the class. They are: female, younger, external locus of control types, have personalities prone to anxiety, and carry lower G.P.A.'s. These effects, though interesting, are with one exception, fairly weak. The exception is that the 17% of students who own a computer or terminal were found to be very significantly less computer anxious than their counterparts. Whether they are not computer anxious because they own computers or they bought computers because they had low computer anxiety cannot be determined from these data. It is clear, though, that these students approach the introductory course with very different skills and attitudes than their peers.

Our third objective was to obtain a "Rationale" profile of the class. Students were asked whether they were taking the course because it was required, was an elective, or whether they really didn't know why they were in the course. Results are in Table 5. As expected, the vast majority (83%) of students are in the course because it is required. Analysis of variance showed that those taking the course as a requirement were significantly more computer anxious (F=10.53, p=.000) than those who were in the course for the other two reasons. This argues for a separate section of the course for those not taking it as a requirement. Further support for this suggestion follows:

A fourth goal was to compile a "Knowledge" profile of the students in a typical introductory computer class in order to determine whether high schools and a generally more computer literate environment are producing students with computer knowledge and skills that obsolete college level introductory computer courses. Computer knowledge was measured using an objective test containing 15 questions of varying difficulty about hardware, software, and systems. Students were also asked to self-assess their levels of computer skill (hand-on keyboarding and programming skills) as "Zero," "Low," "Moderate," "High," or "Very High." Additionally, questions were included to assess the students' experience levels in specific areas such as BASIC and word processing. The objective knowledge test yielded a mean score of 4.8, with a possible range of 0 to 15. While this indicates that most students enter the course with the expected low levels of computer knowledge, there are a few students with considerable computer background who are likely to be very bored with the course. Their self-assessed levels of computer knowledge coincide very closely with the results of the objective test.

Self-assessments of computer skill were even more negative, as only 5.9% of the respondents rated their skills as "High" or "Very High." Women assessed their computer skills at the beginning of the course significantly lower than did men (F=4, p=0.031). This difference between the sexes disappears in the expected levels of computer skills at the end of the course. Women seem to feel that they enter at a disadvantage to the men, but, that by the end of the course, their abilities will equal their male counterparts. There was no significant difference between the sexes on the precourse knowledge self-assessment or on the scores from the objective knowledge test.

Especially significant among the computer experience results in Table 6 is the finding that 41% of the students have at least some experience in BASIC, and that 14% have more than a year of such experience. Designing introductory courses to assume no previous programming experience may demotivate a significant portion of the students.

Not surprisingly, computer anxiety correlated significantly inversely with objectively measured computer knowledge (-.73, p=.000), computer experience (-.41, p=.000), self-assessed computer knowledge (-.52, p=.000), and computer skills (-.50, p=.000). This means that computer knowledge and skill appear to be more strongly related to computer anxiety than the psychological correlates discussed earlier.

The "Expectations" profile of Table 7 is built from responses to questions about students' expected grade in the course, how well they will like the course, and the amount of computer skill and knowledge they expect to gain from the course. Most students enter the course with positive attitudes, expecting to like the course and do well in it. They have reasonable expectations about the computer skill and knowledge gains they will achieve from the course. Only a minority expect they will become experts. It is important for educators to capitalize on these initially positive attitudes by avoiding design and execution errors in the course that could destroy these positive attitudes. Interestingly, computer anxiety correlated -.51 (p=.000) with the grade expectations, so students with high anxiety expect to get low grades in the course. Also, those who expect to dislike the course have higher computer anxiety (.34, p=.000).

STUDENT PROFILE--SUMMARIZED

Demographically, the course primarily attracts sophomores from a diversity of majors. The age, sex, work experience, and G.P.A. of the computer students show them to be a representative cross section of undergraduates. About 17% of the students taking the course at Kent State own a computer or terminal. Psychologically, almost 19% of the students are highly computer anxious. These same students are also negative about the impacts of computers on society, have high math and trait anxiety, and are external locus of control types. Women tend to be more computer anxious than men. Students with lowest computer anxiety have terminals, are taking the course as an elective, and have high computer knowledge, skills, and experience. The vast majority of students, though are taking the course because it is required.

About 40% of the students have had BASIC experience, but the mean general computer knowledge and skill scores are quite low. This implies the computer literacy thrust at the high school level may be leaving many gaps that need to be filled by universities.

Student's expectations about the course are reasonable, and they are coming into the course with generally positive attitudes.

RECOMMENDATIONS

1. Students should be pretested for computer knowledge, skill, experience, and computer anxiety.
   a. The 20% with highest anxiety and lowest computer ability, especially those for whom the course is required, should be placed in a special version of the course that offers a slower pace on the machine and more personal assistance. An adaptation of the cognitive skills modification math anxiety treatment (3) could be used in this special section.
   b. The 20% with lowest anxiety, highest computer ability, and who own terminals/computers should be placed in a fast track version of the course.

2. The results indicate that if one wishes to avoid the trouble of giving a pretest, that the close approximation to the three-course-versions model above would be to place students taking the course because it is required into the normal version of the course, and those taking it as an elective in a fast track version.

3. Course assignments, application examples, and text books should be chosen with the awareness that many major areas of study are represented in the client group. A test containing only business-oriented material will be inappropriate for a nursing student.

REFERENCES

1. Barkin, S. An investigation into some factors affecting information system utilization. Unpublished Ph.D. dissertation, University of Minnesota, 1974.

2. Dickson, G. W., Senn, J. A., & Chervaney, N. L. Research in management information systems: The Minnesota experiments. <u>Management Science</u>, <u>23</u>, May, 1977.

3. Fennema, E., & Sherman, J. A. Fennema-Sherman mathematics attitude scales, instruments. Designed to measure attitudes toward the learning of mathematics by females and males. <u>Journal for Research in Education</u>, <u>7</u>, 1976, 324-326.

4. ICIS, Sixth International Conference on Information Systems: Panel session on information systems curriculum. Nunnamaker, Davis, Couger et al. Indianapolis, Indiana: December, 1985.

5. Howard, G. S. Computer anxiety and management use of microcomputers. Ann Arbor: UMI Research Press, 1986. (In press.)

6. Howard, G. S., Murphy, C. M., & Thomas, G. N. Computer anxiety considerations for design of introductory computer courses. Under review for publication in <u>Educational Research Quarterly</u>.

7. Huber, G. P. Cognitive style as a basis for MIS and DSS designs: Much ado about nothing? <u>Management Science</u>, <u>29</u>, May, 1983.

8. Raub, A. C. Correlates of computer anxiety in college students. Unpublished Ph.D. dissertation, University of Pennsylvania, 1981.

9. Spielberger, C. D., Gorsuch, R. L., & Lushene, R. E. <u>Strait-trait anxiety inventory manual</u>. Palo Alto: Consulting Psychologists Press, 1970.

10. Valecha, G. K., & Ostrom, T. M. An abbreviated measure of internal-external locus of control. <u>Journal of Personality Assessment</u>, <u>38</u> (4), 1974, 369-376.

11. Weinberg, S. B., & English, J. T. Correlates of cyberphobia. Unpublished paper, St. Joseph's University, Philadelphia, Pennsylvania, 1983.

12. Zmud, R. W. Individual differences and MIS success: A review of the empirical literature. <u>Management Science</u>, <u>25</u>, October, 1979.

Variable Type	Variable	Citation
Demographic	Age	
	Sex	
	College	
	Grade Point Average	
	Year in school	
	Years of full time work experience	
	Own a computer or terminal?	
Psychological	Computer anxiety	Raub (1981)
	Attitude toward impact of computers on society	Raub (1981)
	Math anxiety	Fennema and Sherman (1976)
	Locus of control	Valecha and Ostrom (1974)
	Cognitive style	Barkin (1974)
	Trait anxiety	Speilberger, et al., (1970)
Rationale	Reason for taking the course	
Knowledge	Computer knowledge (objective measurement)	
	Computer experience	
	Computer skill (self-assessed)	
	Computer knowledge (self-assessed)	
Expectations	Expected grade in the course	
	Degree to which student expects to like the course	
	Level of computer skill expected at end of course (self-assessed)	
	Level of computer knowledge expected at end of course (self-assessed)	

Table 1. Variables assessed by the questionnaire.

Variable	Result	
Age	x=20.4 o=2.8	
Sex	52% Male, 48% Female	
College	Arts and Sciences	18%
	Business	41%
	Education	2%
	Fine and Professional Arts	37%
	Physical Education/Recreation	1%
	Nursing	1%
G.P.A.	x=2.7 o=.61	
Year in school	Freshman	25%
	Sophomore	51%
	Junior	15%
	Senior	7%
	Other (Audit, etc.)	2%
Years of full time work experience	Zero	30%
	Less than 1 year	20%
	1-3 years	34%
	4-6 years	11%
	More than 6 years	5%
Own a computer or a computer terminal?	Yes	17%
	No	83%

Table 2. Demographic profile of the "Introduction to Computers" students. (N=239)

Variable	Result
Computer anxiety	x=13.4 o=7.2 18.8% > 20 Possible range: 0 (low anxiety) - 40 (high anxiety)
Attitude toward the impact of computers on society	x=22.9 o=4.8 Possible range: 0 (negative attitude) - 32 (positive attitude)
Math anxiety	x=21.5 o=10.8 Possible range: 0 (low anxiety) - 48 (high anxiety)
Locus of control	x=24.5 o=4.9 Possible range: 11 (internal) - 44 (external)
Cognitive style	x=11.1 o=13.8 Possible range: 0 (heuristic) - 17 (analytical)
Trait anxiety	x=20.4 o=7.4 Possible range: 0 (low anxiety) - 60 (high anxiety)

Table 3. Psychological profile of the "Introduction to Computers" students. (N=239)

Variable Pair	Result
CMPANX-SOCIMP	-.35 P=.000*
CMPANX-MTHANX	.29 P=.000*
CMPANX-LOCCTL	.12 P=.028* (Externals more anxious)
CMPANX-COGSTL	.02 p=.300
CMPANX-TRTANX	.35 p=.000*
CMPANX-AGE	-.16 p=.005*
CMPANX-GPA	-.13 p=.026*
CMPANX-YRSWRK	-.10 p=.055
CMPANX-SEX	F=5.08 p=.025 (Females more anxious)
CMPANX-OWN	F=10.02 p=.002* (Students not owning a computer or terminal much more anxious.)

Legend:		
	CMPANX	- Computer anxiety
	SOCIMP	- Attitude toward impact of computers on society
	MTHANX	- Math anxiety
	LOCCTL	- Locus of control
	COGSTL	- Cognitive style
	TRTANX	- Trait anxiety
	AGE	- Age
	GPA	- Grade Point Average
	YRSWRK	- Years of full time work experience
	OWN	- Own a computer or terminal? Yes/No

Notes: 1. Correlation coefficients are Pearsonian
2. Significant ($p \leq .05$) correlates are indicated by an asterisk.

Table 4. Psychological and demographic correlates of computer anxiety in the "Introduction to Computers" students. (N=239)

Category of Computer Experience	Percentage of Respondents Indicating Each Range of Length of Experience			
	Zero	<1 Year	1-3 Years	>3 Years
Word Processing	63%	27%	9%	3%
Video Games	34%	32%	23%	12%
Use of Graphics Packages on Microcomputers	76%	15%	7%	1%
Operated Equipment - No Programming	39%	39%	18%	5%
Participated in Non-Technical Design of Information Systems	89%	6%	3%	2%
Participated in Technical Design of Information Systems	93%	4%	2%	1%
Programmed in BASIC	59%	27%	12%	2%
Programmed in Fortran, COBOL, etc.	85%	10%	3%	2%
Earned Money Using Their Computer Skills	93%	2%	4%	1%

Table 6. Computer experience profile of the "Introduction to Computers" students. (N=239)

Expected Grade	%	Degree to Which Students Expect to Like the Course	%
A	41%	Like it a lot	28%
B	49%	Like it somewhat	54%
C	9.6%	No opinion either way	11.3%
D	0.4%	Dislike it somewhat	4.6%
F	0.0%	Dislike it a lot	2.1%

	Computer Skill Assessment Before Course (%)	Computer Skill Expected After Course (%)
Zero	38.1%	0.4%
Low	32.2%	5.4%
Moderate	23.8%	63.6%
High	4.6%	25.5%
Very High	1.3%	5.0%

	Computer Knowledge Assessment Before Course (%)	Computer Knowledge Expected After Course (%)
Zero	27.2%	0.4%
Low	48.1%	3.8%
Moderate	20.5%	58.2%
High	3.3%	33.5%
Very High	0.8%	4.2%

Table 7. Expectations profile of the "Introduction to Computers" students. (N=239)

Reason for Taking Course	Percentage	Mean Computer Anxiety Score
Required course.	83%	14.28
Elective course.	15%	9.19
Don't know why I'm taking the course.	1%	4.67

Table 5. Rationale profile for the "Introduction to Computers" students. (N=239)

RETURN TO INDUSTRY PROGRAMS FOR FACULTY

By: Carol S. Chapman, MBA
AFFILIATIONS: Haywood Technical College, Clyde, NC
Business Computer Programming Instructor

Data Processing Management Association Member, Western Carolina Chapter
National Business Education Association Member

BS, MBA Degrees from Western Carolina University

Haywood Technical College offers a Return To Industry Program for faculty members to allow them exposure to business environments and ideals in order to increase instructor effectiveness in preparing students for a career. This paper is a statement of the benefits this program has to offer.

INTRODUCTION

A college Return To Industry Program involves the return of faculty members into the Business work environment for a period of time, or avails funds for attendance of conferences and/or workshops related to Business or non-academic situations. State funding provides the necessary monies to operate the program.

BENEFITS

Participants in the Return To Industry Program usually choose to participate for one of several reasons. One is to become familiarized once again with the so called "Real World" atmosphere of the field in which the participant is teaching. The patterns of work development, decision-logic structure, and "Real World" politics are among the topics most often discussed. A second reason to participate concerns new instructors who have had little or no business experience. They need to become familiar with concepts dealing with personnel interaction, daily work structure, timetables, and management of duties, stress and goals. Professional Development Seminars devoted to non-academic environments are often a good source for this type of information.

The fields of Data Processing and Management Information Systems are dynamic to the point that academic life does not always provide the hands-on experience needed to prepare students for careers. Return To Industry provides a benefit to faculty in allowing interaction with daily business operations, which helps both the faculty member and the business. The information exchange between faculty and business is positive toward promoting a better understanding of the changing technology and needs in the industry. Since one goal of education is to prepare students for the work force, it is essential to keep faculty updated on business expectations and demands for and upon employees.

A third reason to participate involves Maslow's hierarchy of needs. Self-esteem needs and the need to have approval from others often force people to seek the additional training or learning which they perceive will bring this approval. In order to achieve or maintain a place of importance in the academic field, a faculty member must often prove that he or she is capable of surviving in the true business environment. A thorough knowledge of the current literature in the chosen field is not quite enough. By returning to industry for a short period of time the short-term goal of success is much easier to achieve than a long-term goal of success as a permanent employee in the industry.

EVALUATIONS

Participants are asked to evaluate their Return To Industry experiences. This evaluation includes questions regarding the nature of the business exposure, the duration involved, the extent to which the user understood the purpose for the program, and the users observations regarding their adventure. These observations include suggested optimum time for return experiences, perceived benefits prior to and subsequent to the experience, skills learned, and how these all might be incorporated into the teaching methods and/or information presented to other faculty and students.

SUMMARY

The Return To Industry Program has merit and most participants agree that it is valuable to increasing instructor information and understanding regarding the business environment. This information is therefore available to be passed along to other instructors and to students in preparing them for what they will face in the "Real World" of employment in a career field.

THE BUSINESS FACULTY EXCHANGE:
WHEN MINIMUM EFFORT = MAXIMUM BENEFIT

L. L. Purcell M.B.A., CDP
Systems and Programming Supervisor
Georgia Farm Bureau

Adjunct Instructor
Macon Junior College

The purpose of the business faculty exchange is to help students prepare for their careers thereby, producing qualified employees to fill business needs. The three level communication process creating the exchange requires minimal efforts from participants. The idea level is exchanging thoughts, through continuing dialogue, on current practices and course/curriculum content. The action level represents activities that can be implemented to achieve better understanding of subject material for students. The personnel level is a deeper commitment to the exchange by using individuals to bridge gaps. All methods recommended represent minimal effort and result in both short and long-term concrete benefits.

THE BUSINESS FACULTY EXCHANGE: WHEN MINIMUM EFFORT = MAXIMUM BENEFIT

Why bother - minimal effort or not, what is the value a business-faculty exchange?

The importance depends on your point of view. As a business person I find it frustrating to have openings for systems personnel and no one qualified to fill them. Each June dozens of resumes from eager computer science graduates pile upon my desk. Students appear to have been either coding BASIC on P/Cs or programming compilers in FORTRAN, and in neither case do they know about anything but flowcharting. Concepts and skills, such as accounting, COBOL, VSAM, and JCL are greeted with blank looks by the former and scorn by the latter.

As an educator it bothers me to see students graduate not having been appropriately trained to find positions. The main purpose of education is to prepare students for their career. If students cannot get jobs - we have not done ours.

How - how can we easily and effectively help the students and consequently help ourselves?

The thesis is minimal effort can result in maximum benefit. The following is a practical process through layers of communication, which has been tried, to support that thesis. The realization that both business and faculty are on the same team working towards a common goal should be the basis for all communication. Working with this supposition will help to preclude the non-productive "finger-pointing" sessions of who hasn't done what.

The initial contact can be made by either party, through a DPMA chapter or on an individual basis.

On the individual basis - a phone and an invitation will get things started. The educator can phone a DP manager of a local shop with inquiries such as:

None of our graduates have been hired by your firm, do our graduates lack some particular training?

You have hired some of our graduates, have they performed up to your expectations?

The DP manager can phone the educator with comments regarding their needs and is there a possibility of the school helping. No matter who makes the first move, the call should result in definite arrangements to meet for further discussion.

If there is a DPMA chapter available, either party can extend an invitation to the other to visit for the purpose of opening up communications.

Another option for business oriented chapters is to invite a faculty member to discuss the curriculum offered at their institution. A note of caution - there was a situation where a local professor was speaking at a meeting. He was expressing his delight with the school's scientifically oriented classes, when a few members of the business audience verbally attacked him for ignoring their needs. A loud discussion ensued to the benefit of no one, particularly the students who are still not getting hired. This should not deter anyone from trying this option, just do so in a controlled environment.

It is not important how contact is established, as long as it is positive, it will serve to lead into the first level of communication.

Ideas - the first level.

Exchanging thoughts and feelings on the question "What can students be taught to prepare them for jobs?" The basis for discussion should be current technologies, current practices in local shops, and schools' curriculum. The meeting needs only to last a couple of hours, thereby requiring almost no effort from either party.

The gains that will be derived for the school is knowledge about what the students need to learn. The business has the opportunity to help form the education of future employees. The more appropriate training students receive in college means the less you must give them on the job.

Even if contact stops here, there should have been enough information exchanged to help college highlight areas needing modifications, if any. It should have also given the schools a direction to work in, for these areas. The intention, however, is not to let it stop there, but let it be the beginning of continuing dialogue. Future sessions can occur on an as needed basis to review texts, and develop course and curriculum content.

The benefits are not immediate when ideas are exchanged, instead they are long-term and great when students graduate and are gladly hired.

Actions - a more involved level.

The action level means doing something to implement the ideas. It is where business asks "What can we do to support your efforts to provide us with employees?" Colleges have a right to expect this offer the same way DP managers expect users to get involved when systems are being developed for them. Systems can be developed without user input, but usually do not suit the users' needs as well as they could. Applying the same theory to student development - without business's help, schools can still prepare students but not as well as they could with it.

Some illustrations:

Be a guest lecturer. If there is a topic that the faculty cannot adequately cover, such as operating systems and JCL have a programmer come in to do that lecture.

Host student tours. Invite a class to tour your facility; have operations personnel explain their equipment and job functions. In viewing the programming area, have analysts explain the program development process used.

The business community can be a vast resource, and in the examples above the time required is only a few hours. This underlines the fact again, that minimal effort can result in maximum benefits.

There are services that the college can offer to the business community, so that this level of communication is not one-sided. Perhaps the college can offer a class a group of employees need, waiving normal entrance requirements for those who would not traditionally qualify. Or if a class is filled, employees can be registered regardless. Here too, the effort on the part of the college is minimal and yet can be of benefit to a business.

Personnel - a deep commitment and involvement toward the goal.

Exchanging personnel does mean more effort, but if many people are working together, the effort for each will be less. The most obvious way to exchange is to have a business person, who is also qualified to teach, do so. Personally, it is rewarding, using my experience to help students understand real world computing. It takes about six hours a week, occasionally more, to prepare and teach one course. If you have more than one person in your community who can help this way, have them alternate quarters, so that one person does not have all the responsibility.

If there is no one in the area who has the qualifying degrees to teach, perhaps special arrangements can be made because of their technical expertise. This would take time on the part of the school to work out details of the arrangements, but it should be worth it. Another option, if the above cannot be worked out, have the business person act as guide on projects or resource person that students may contact. It must be noted, that for classes scheduled during normal working hours a company must be willing to let the employee off.

Internships and cooperative programs are other types of exchanges. These programs are designed to give students experience working in the field. In terms of efforts there is little administratively the company must do, but it does take company time to train someone in the shop specifics. Keep in mind, this intern is only with your firm for three months. The school might have more administratively to do because they must value the credits and help the student work the hours in.

The benefits of such programs are great. The business gets a few programs written by a "no cost" employee. The free trial period basis allows the company an opportunity to try out an employee. Many firms, when impressed with an intern's work, will hire them when they graduate because the firm knows what it is getting. To the student, internship means experience for their resume, and a chance to get their foot in the door.

In conclusion, we are speaking of effort on your part to make a contribution to the education system. The effort is minimal though, and can be demonstrated in many ways.

Most important is the great reward derived from the work. Minimum effort results in maximum benefit, try it, it is worth your time.

Strategies for Supporting Diversity in End User Computing

Rosann Webb Collins

Microcomputer Trainer/Consultant
Academic Computer Center
University of North Carolina at Greensboro

ABSTRACT

Many organizations mandate the purchase and use of one microcomputer hardware system and a limited set of microcomputer software; this standardization is preferable for many reasons, including ease of microcomputer support service. Many universities have historical and continuing reasons for supporting diversity in end user computing. End user support for microcomputers need not ignore the use of existing systems nor the purchase and use of new systems which match individual needs rather than the organization standard. In this paper several strategies for, critical factors in, and methods of supporting and training faculty and staff users of diverse hardware and software systems will be presented.

The university, like many other organizations, has historical and continuing reasons for supporting diversity in end user computing. While every effort should be made to encourage selection of microcomputer systems which are standard for the organization, end user support should not ignore the use of existing systems nor the purchase and use of new systems which match individual needs rather than the organization standard. In this paper several strategies for, critical factors in, and methods of supporting and training faculty and staff users of diverse hardware and software systems will be presented.

Why diversity?

Supporting faculty and staff end users who have the same microcomputer hardware and software is certainly preferable from the point of view of the support service. Conformity of this type makes training easier, because the end users learn on exactly the same configuration they will be using at their desks. Standardization makes the individual consultation effort easier, because staff build up a repertoire of solutions to problems and become very knowledgeable on the capabilities and limitations of that system. Advising end users on system purchases becomes simple when options are severely limited, and greater discounts are available when hardware and software are purchased in quantity. Why, then, would an end user support service tolerate or even encourage diversity?

At the University of North Carolina at Greensboro, one reason for diversity is historical. Many faculty and staff at the university, unlike those at many other universities, began using microcomputers in the mid 1970's, before the microcomputer marketplace became dominated by a few brand names/operating systems. As a result, academic and administrative units use everything from older systems like Commodore Pets, S-100 bus home brew systems, Apple II Pluses and CP/M-based microcomputer systems. Even within type of hardware or operating system, there is great variety in types of software used for the same application. For example, one School uses a DEC Rainbow in the dean's office, Epson CP/M systems in departmental offices, and Apple II Pluses, IIe's and IIc's in a faculty/student laboratory. Since no microcomputer products were on state contract until 1980, and since the central administration exerted no control over purchase of the equipment and provided no support for its use, there was no incentive for individuals to select compatible equipment. Each department or individual bought what was available, what staff were familiar with, or what a vendor sold them.

Beyond this historical reason for diversity of equipment, there are continuing reasons why one brand of hardware and one set of software cannot be mandated. Within even a medium sized university there are disciplines which require that a computer generate various foreign language characters on screen and on the printer, be connected to laboratory devices for data input, generate formulae and models on the screen and printer, support CAD applications, manipulate high density graphics such as satellite photographs, as well as perform the standard applications such as word processing and spreadsheets. No one hardware and software system can meet all these needs. While this diversity of needs is not typical of all business organizations, there are many which mirror this range of applications.

Another reason for choosing a non-standard system is the background of the end user. Some faculty have much experience in using one particular system for a specific application. For example, on the Greensboro campus, several Psychology and Physics professors use Commodore systems to interface directly with laboratory equipment for data collection. While this application could be done with an MS-DOS system, these faculty already know how to make these interfaces and program the systems. While faculty members might choose to move to a more powerful system and therefore force themselves to learn a completely new system, suggesting that they abandon their experience and knowledge for the sake of standardization would be absurd.

Not only are the microcomputer systems used at the university varied, the users of information systems are also quite diverse. Like most business organizations, end users range from novices to power users to the individuals who actually design microcomputer hardware and software. Many users, especially new faculty, come from more sophisticated computing environments, and are making a transition from being power users of mainframe computers to learning to work within the constraints of microcomputer systems. Many are quite expert in one computing environment but close to a novice on microcomputers.

STRATEGIES FOR SUPPORT

The problems created for a end user support service by diversity of systems and users are many. But five strategies for support have been developed and employed in such an effort, with great success.

1. Training for Common Uses and Functions

The cornerstone strategy for supporting diverse end user computing systems is a training effort which emphasizes common uses and functions in applications. Rather than trying to offer a training program on all types of hardware and several packages of each type, the training should center on teaching the application itself, merely in the context of a certain piece of hardware and software.

For example, a seminar entitled "Word Processing in the Office" was designed for secretaries and other clerical workers in the various administrative units on campus. The curriculum for this seminar focuses on the basic functions of word processing programs and on how to organize and maintain word processing files. This information is presented using a specific program, but participants are warned that the programs and hardware they will use will change radically during their career. With this orientation, the participants know they will need to transfer general principles for using this application to specific systems, both now and in the future.

Since a desireable side effect of the training with a specific tool on a specific hardware system is to encourage conformity where appropriate, the tools and systems for training should be those that are recommended for purchase. Not only do the participants realize the benefits of using those systems, but it provides a way for them to test what unique needs those systems may or may not meet.

2. Consultation for Unique Needs and Problems

The first strategy would be unworkable without the second: consultation for unique needs and operational detail.

While the end users are expected to attempt to make their unique system work for their unique needs, they must not be abandoned to a world of manuals and technical support hot lines. A consultant with the appropriate background can answer many specific questions from their experience, over the phone. Others can be solved by serving as a filter between the end user and technical support resources. The consultant can find it in the manual or coax it from the hot line or glean it from their personal support group (friends, colleagues, users groups), much easier and quicker than the average end user. After initial training sessions, end users can learn on a need-to-know basis. Because the motivation of the learner is at its highest, he or she is more likely to actually learn in this situation than in any other.

3. Diagnosis

Most end users are not capable of diagnosing a problem with their system. They are acting as both user and operator of what is really a complete and complex computer system which they do not (nor do they need to) fully understand, and therefore they are unable to step back and isolate the source of a problem. Support personnel must be available to diagnose problems which are not well defined, and after definition, channel the problem to the appropriate resource for solution. When problems range from pilot error, hardware malfunctions in any of several units, software bugs, incorrect data, to inadequate or unreliable power supplies, it is essential that a first pass at diagnosing be done by end user support staff who appreciate this range of possibilities.

4. <u>Building Independence from Dependence</u>
Novice users in particular demand much "hand holding" while they begin to plumb the depths of a microcomputer application. Training and consultation provide that initial, intense support. But this support must also strive to nurture independence in users. The fact that the support staff is very busy is fortunate, since having to wait for answers encourages the end users to try to work on answers for themselves. Users should be told to "do what you can", and then seek assistance. Insist that users keep of log of problems, and then ask for help; the process of writing down a problem can help the users develop their own skills in defining problems. Encourage informal support groups of users with similar systems and/or applications. Putting a small group of secretaries with the same system together for a bag lunch once a month lets them meet other people they can call for help besides the official support group. Always reward users for independent accomplishments with praise.

5. <u>Supporting Blue Skies</u>
An end user support group should be able to move users from a blue sky idea of what they want to do into a specific description of a computer system that is a tool for accomplishing that goal. Many times this is a process of identifying limitations and possibilities of what is available; sometimes nothing is available and a new tool must be developed. Advertise that the staff will listen to even vague ideas: help the users clarify their ideas and find their way through the maze of tools and possibilities. Then realistically assess what can be done as a guide to further thought and work. This process should not be limited by a precondition of adherence to a standard microcomputer system. Facilitating the creative process is essential in university end user support; it is exciting and challenging in any environment.

CRITICAL FACTORS

There are two main critical factors in implementing these strategies: the curriculum of the training and the characteristics of the end user support staff.

The training curriculum should be a model of the total end user support system. The structure of the seminar sessions should include:

a) information on the common uses and functions of the application;

b) guided, hands-on experience with a specific implementation of the application;

c) activities which require the users to solve problems with the application, with the trainer available for assistance; and

d) dialogue between participants and the trainer on what is possible with this and other applications.

The sessions should prepare them for independent work on problems, with assistance and support when necessary.

The second critical factor centers on the characteristics of the support staff. They must have both the microcomputer experience and the interpersonal skills to keep the strategies in mind while dealing with individual´s needs and styles. They must be able to analyze the user in relation to experience, to communicate in terms the users can understand given that experience, and to provide the appropriate assistance: this high wire act must be combined with technical competence on microcomputer hardware and software. If either the interpersonal skills or the technical expertise are missing, the support will be lacking.

At the university, as in many other types of organizations, a diversity of microcomputer-based information processing tools must be supported, just as the library must provide various formats, sources, and levels of information. Supporting diversity is not impossible if the support service is designed with that characteristic in mind and if staff with appropriate skills can be employed to implement the support strategies.

SPUTNIK STRATEGY FOR FACULTY DEVELOPMENT IN MIS

by

Dr. N. Nagarajan
Assistant Professor in Information Systems
Fordham University
103 Thebaud
Bronx, NY 10458

ABSTRACT

Our duty to our country demands of us an effort to provide the means of a thorough education. There is perhaps no nation whose interest would be more deeply affected, by a substitution of the superficial for solid learning ... Not one of our colleges is a place for thorough teaching; and not one of the better class of them does half of what it might do, by bringing the minds of its instructors to act directly and vigorously on the minds of its pupils ...

This is a quotation from the Yale College Faculty Report on the Course of Instruction in Yale College produced in 1828! One can see how the several reports of current days also reflect the same sorry state of affairs in the academic arena. The educational systems are at the brink of an unprecedented identity crisis. The institutions of a society, especially the ivied and ivory towers of academic institutions, are not adapting fast enough to the impact and consequences of technological changes. Advances in the technology of education help mass higher education to become a reality; but, education in technological advances is essential and vital if higher education is to deal with reality in an effective manner.

In his address to the Israel Institute of Technology in 1966, Eric Ashby discussed in detail on the need to appreciate such a crisis in the coming years and provide adequate measures to stem the adverse consequences thereof.

"At present such a machine would be many times more expensive than a teacher ... But, before the end of this century, such machines will be in the market. Just what part they play in the educational revolution will depend on the skill which goes into the programming ..." (1)

In the rapid strides of computer technology, the faculty are not adequately equipped to train their students in many cases. Their background and specialization are out of time and tunes with the requirements for proper understanding of the advances in the technology and their vast impact on society and the students. It is therefore imperative that the question of faculty development should take utmost precedence in our campuses, if we were to do justice to the primary mission of the institutions of higher learning.

There is an increasing awareness and emphasis on the use of mainframe and micro-computers in the MBA classes for their courses in most disciplines. The freshmen of future college entry-level classes will be already well trained in the principles and practice of computers and data processing, and so their expectations will naturally be still higher than may be found in present times. This trend stresses the far cry for overcoming the major obstacle of qualified faculty to teach Management Information Systems courses with confidence to instil enthusiasm into the receptive minds of their wards.

'Computer literacy' is essentially relational (!) and a moving target. In the early sixties, the term connoted a basic ability to appreciate the computer capabilities and impact on society, and to prepare programs in a computer language with confidence. In the next decade, the rush was to the Computer Sciences major in colleges, and computer-related career-orientation in the curriculum development. In the present decade, computers have become household assets, considered to enhance productivity in operations and expand human capabilities. Sophisticated tools and techniques have been developed to help a nontechnical user and even young kids and family members to derive the benefits and pleasures by using affordable yet powerful computers. Fourth Generation languages, Expert Systems, Graphics, Databases, and complex communication network technology have become the order of the day. Dr. Kenneth King of Cornell University argues that "United States is in a worldwide technology race and leadership in computing is critical to win that race."(2)

By 1995, as indicated already, the new student population will be amply 'computer literate', as the elements of computer science and applications would have been dealt with in detail in elementary and secondary schools. Subjects which form the course content now in the graduate curricula will be moved down to the undergraduate levels. Thus, there will be, in my opinion, a total metamorphosis in the needs of the new generation of college students, wanting for more challenge and scope to learn and utilize the computer resources to tackle complex systems.

Again, to quote Dr. King, "curricular changes tend to move at the rate of a pig through a python!" (2). This is common knowledge to everyone, as the campuses cannot afford to change with swift alacrity to the shifts in the environment for obvious and somewhat valid, pragmatic reasons like the uncertainty of the technology trends, paucity of funds to provide all the essential hardware etc. as necessitated by the trends, and, above all, the lack of sufficient faculty power to meet the critical demands. The result is parallel to teaching Dalton's Atomic theory in this nuclear age.

In order to overcome this paramount problem, it is necessary that all the segments of society seriously look into the various ways and means to solve it in a satisfactory manner. There should be a Sputnik approach to face the challenge of unprecedented dimensions.

First, the campus management should provide adequate support for training the faculty in keeping themselves aware of the state-of-the-art, as it were, in the varied developments of the electronic technology and its applications not just in scientific areas but in the other fields as business, education and home as well. The tenured faculty in traditional fields feel threatened by the lure of the computer-oriented majors, and felt somewhat relieved when the recent trend shows a slight drop in the freshmen rushing to computer science majors! It will be an effective way to retrain such faculty for fruitful utilization of their experience and expertise in other fields, thus deriving utmost benefit with Pareto-like benefit all around.

Second, the businesses and other professional organizations should seriously consider providing strong support to higher education institutions in the form of funds and equipment. Such a strategy is now under way in the United States through the munificence of leading corporations as IBM with cooperative ventures with leading universities. Again, it may sometimes prove to be the familiar pattern where the rich get richer due to their clout and aura and other poorer institutions left lagging behind, like Oliver asking for more.

American Assembly of Collegiate Schools of Business (AACSB) have recognized this lacuna of trained faculty in MIS fields at the same time when information systems area is experiencing explosive importance in business schools. It has attempted to ameliorate the abject situation to certain extent by running annual summer institutes in Basic and Advanced Information Systems Faculty, as a response to the emerging needs of business and to the faculty shortage among schools. These programs are an important step forward in meeting the urgent needs of business schools, and consequently of business and industry. But, the registration to each institute is limited to mere 50 persons, indicating that much more needs to be accomplished in this direction in order to meet the staggering lack.

Third, the governments and their specific agencies should provide adequate support for the retraining of talents of faculty in the requisite fields of changing technology. The clarion call of the politicians to reach the stars with technological superiority should correspondingly be translated into tangible, targeted programs for training the faculty who are expected to train their students in turn.

Otherwise, the ill-informed faculty would be grudgingly leading the innocent minds of their students into dark alleys with dead end, thus defeating the purpose of the educational mission. Such a tragedy will give further support to the satiric comment of the present-day situation:

"If you can't get any other job, teach. If you can't teach any other subject, teach EDP. If you can't teach EDP, teach EXPERT SYSTEMS!"

BIBLIOGRAPHY

1. Ashby, Eric. Adapting Universities to a Technological Society, San Francisco, CA: Jossey-Bass (1974).

2. King, Kenneth M. "Evolution of the Concept of Computer Literacy," EDUCOM BULLETIN, Fall,'85 Vol. 20 No. 3.

BRIDGING THE ACADEMIC/BUSINESS GAP: A THREE PRONGED APPROACH

Eli Boyd Cohen

Department of Management Information Science
California State University, Sacramento
6000 J Street, Sacramento, CA 95819-2694
(916) 427-7000

TRACK: Business-Faculty Exchange

ABSTRACT

The author serves as Educational Trustee for a local Data Processing Management Association chapter. As such, the author developed a three pronged approach to bridging the gap between academia and business. The approach's three activities include the following: 1) a Jobs Faire, in which Data Processing Management Association members talk about their jobs with students, 2) a conference session, in which top management address a data processing audience on the subject of what colleges can do to help business, 3) a symposium of the region's top employers and the colleges' information systems departments.

RATIONALE

Education is one of the main purposes of the Data Processing Management Association. An important part of this education is improving the communications between business and academia in the data processing community. A local Data Processing Management Association chapter began "bridging this gap" when the author assumed the office of Educational Trustee.

Local colleges must do more than just train students for entry level positions. The colleges should serve as sites for retraining general management on information system and for upgrading the training of the current data processing personnel. In the author's view, Data Processing Management Association must work with the colleges in meeting these goals.

THREE PRONGED APPROACH

For these reasons, the local Data Processing Management Association chapter conducted three sessions to bridge this gap.

1. Data Processing and Students

To help current students better understand the jobs for which they are training, the chapter conducted a Jobs Faire. Member of Data Processing Management Association who hold different positions in the industry formed a panel which visited the local colleges. Each member described his/her job and related personal experiences about their career.

The jobs represented in this panel included analyst, beginning programmer, programmer team leader, data processing manager, and technical specialist.

One of the benefits for the students of this presentation is to overcome misconceptions. Some students were avoiding taking programming classes because they believed that analysts need not know programming. Others had intended to make data processing management their first job.

In addition, the presentations may indirectly affect curriculum through influencing the students' choice of electives. Popular electives are taught more often then unpopular ones, and so serve more students.

The panel was video taped, and this tape given to the school's library. It is hoped that by this means many more students will learn about DPMA and the local job market.

This program is highly recommended. It is easy to arrange with little cost and little risk, while it provides higher benefit.

2. Data Processing and General Management (with Academia)

The local chapter of DPMA conducted an Information Processing Symposium. The Symposium contained dozens of training sessions. This symposium made possible a second program to bridge the Academic/Business gap. One of the sessions was devoted to this topic.

Several general managers were invited to address the session. The managers were asked to respond to questions which they received with their invitation. These questions included the following:

"What problems with computerization does management see?"

"Can the data processing department help?"

"Can local colleges help?"

The answers given by the managers (from their perspective) were surprising. All speakers focused on problems that their company had with communications with their data processing department.

More troubling was that these top managers did not perceive colleges as a resource for help with this problem, or even for retraining their personnel. (The author views this response as a source of evidence that the colleges are not doing a credible job in marking their services to business.)

This project cost little, had low risk, but also had limited benefit.

It was low cost since the symposium was already set up. Only the management of this session was required.

It was low risk because, as a worst case, it would have low attendance.

Unfortunately, it must also be considered low benefit, as not everyone who was interested could attend.

3. General Management, Data Processing, and Academia

The third prong of the three prong attack to bridge the gap involves inviting the top data processing employers and the colleges in the area to take part in a symposium. The purpose of this symposium is to promote communication between these two groups and improve the data processing curriculum.

All participants will be required to pay a modest fee. This fee will be used to defray the costs of putting on the conference. Requiring a fee for participation will also demonstrate the commitment to this event.

The employers will be asked to send a delegation including representatives of top general management and of data processing management.

The universities will be asked to send a delegation from their department of MIS or its equivalent. The colleges invited will include four-year colleges, two-year colleges, and trade schools.

The approach taken in inviting participants will be that the symposium is in their interests as well as a civil obligation. For the employers, it is in their interest to tell the schools what their needs are for trained personnel. For the schools, curriculum review requires input from the community.

In addition, it provides the groups with the opportunity to exchange views and marketing. The schools in particular will be interested in showing what their programs have to offer.

In addition to providing the schools with feedback on their **present curriculum** in relation to needs, the symposium also will be a start for **internships** programs. Many college programs either require or encourage students to serve as interns as part of their requirements for graduation. Colleges are also recognizing the benefits of having their faculty serve as interns to business.

The symposium is planned as follows:

The formal portion will be composed of formal presentations and questioning from the floor. Attendees will be placed in panels, with each being given the task to address selected questions. These questions will be given to each participant in advance of the conference. Background materials also will be sent in advance.

Questioning will follow each panel discussion.

The symposium also will have an informal, yet equally important time. During the planned socializing, a great deal will be accomplished, primarily on the inter-personal level.

This activity has high cost, high risk, and high benefit.

The high cost includes a financial cost (for lunch and meeting room). More importantly, it will take a great deal of volunteer time, a most precious commodity.

It will also incur high risk, largely because so many things must go right for the activity to succeed.

However, the benefit from this activity is extremely high: for the curriculum, college professors, top management, and data processing departments

SUMMARY

Bridging the gap between business and academia is an important task. One Data Processing Management Association chapter found three separate activities helped, each in its own way, to bring the parties together. One activity had DPMA members describe their job experiences with students. Another had top management share their view of Data Processing and colleges with DPMA members. A third is a formal symposium between faculty and employers.

PREPARING CIS STUDENTS TO BECOME EFFECTIVE SYSTEMS ANALYSTS

Patricia C. Eiland
Assistant Professor
Auburn University at Montgomery
Ava S. Honan
Assistant Professor
Auburn University at Montgomery

ABSTRACT

Today's business environment has been impacted at every level by an onslaught of new advances in computer technology and the availability of affordable computer hardware and software designed to assist businesses in both strategic planning and in day to day operations. Most organizations must rely on either "in-house" or contracted technical support to assist in the determination of which computer-related tools will maximize efforts to attain objectives and respond to daily operational needs. In this situation someone must bridge the gap between the non-technical management seeking the solution and the technical support staff that understands the computer's capabilities and limitations. The person in this interface role is often a systems analyst. The purpose of this paper is to explore the skills that systems analysts must possess to function effectively and how to best prepare students for a career in this role.

INTRODUCTION

Management today finds itself in the sometimes precarious position of determining which of the many computer-related tools, if any, would be most effective in improving the decision making processes or the operational environment of its organization. Persons in this role, often lacking the technical knowledge to select an adequate, much less optimal, solution, often call upon a systems or programmer analyst for technical assistance.

The primary role of the systems analyst becomes one of providing a bridge between the need of management for information support and the selection of tools that will provide that support. It is the responsibility of the systems analyst to determine how the various application areas in the organization function, what strategic and operational decisions are made and what information is required in making them. Any constraints and mandates must be uncovered and understood. The analyst must determine which automated tools, if any, would provide optimal support under existing, unremoveable constraints.(1)

The ability to function in this role requires special skills in both the technical and behavioral aspects of the situation. Tact and other communication skills, leadership qualities, technical expertise and a sensitivity toward other people and the impact of technology on people are essential skills to a successful systems analyst. Providing opportunities to develop these skills should be one of the primary goals of a Computer Information Systems (CIS) curriculum.

One must acknowledge up front that an analyst is not a "super person" able to solve everyone's problems and provide the best technical support solutions in a business environment. He (or she) is simply a "normal" individual with certain specialized skills and inborn traits geared toward analytical work and communications doing his (or her) best to provide a service by interfacing today's evolving computer technology into the business environment so that the non-technical community can function more effectively.

THE CHARACTERISTICS OF AN EFFECTIVE SYSTEMS ANALYST

A systems analyst must be able to communicate between the technical (data processing) staff and non-technical end users. This demands a solid grasp on the operations and information requirements of the application areas being considered. This must be balanced by a sound knowledge of the available information system support technology including available systems and applications software. The analyst assumes a translator role providing technical information to users in "layman terms" and an explanation of end user requirements and constraints to the technical support staff.

Even though an analyst may not be an expert in the particular application area, he (she) must possess the skills required to ascertain the processes utilized in the application and the required information support through observation and inquiry.

Successful analysts must be aware of available technical tools, both hardware and software, and how these tools function in an application environment in terms of both the capabilities and limitations. Exposure to a diverse spectrum of available hardware and software from those in the micro computer environment to mainframes should be incorporated into our Computer Information System (CIS) curriculum. We can not expect to make our students experts on this broad spectrum of technology. However, exposure will provide them with a broader base of knowledge for determining possible solutions to information processing problems and a basis for more indepth investigation. (1)

An analyst must have a firm grasp of effective techniques for determining user needs based on operational requirements and decisions made in the day-to-day operations and strategic planning functions of the organization. He (she) must be able to mesh these informational requirements into functional component procedural flows. He or she then must be able to select the methods for most effectively implementing the resultant logical information retrieval system under the existing organizational constraints. Our curriculum must provide both basic analytical techniques and methodology for developing simple applications as well as "full blown" management information systems in order to give students a better grasp of the transition between identifying user needs and translating these needs into an operational information retrieval system.

An analyst must be a self-starter, motivated by the job at hand and able to proceed even when the objectives are poorly defined or appear to conflict with one another. This requires leadership abilities, administrative skills and the ability to plan ahead and maintain control of a project. In the role of a communicator and catalyst, the analyst is the focal point that determines the outcome of the project. Even though many factors can make or break a management information system (MIS), the lack of adequate communication between the technical staff and the user will certainly cause the MIS to fail to meet its objectives. (1)

Our students must learn to complete large assignments with minimal direction. This continues to be a difficult hump to overcome with most students. Many of them have great difficulty (even with detailed directions) in breaking down large or complex assignments into workable modules. Many lack the discipline to follow through on a timely basis and continue to believe that even the largest project can be accomplished a few days before the due date! This philosophy spells disaster for an analyst on the job. We owe it to our students to instill falacy of this attitude in them before they enter the job market.

An analyst must be analytical and be able to approach a problem using logic. He (she) must be able to take complicated or confusing situations and break them down into well organized but simple components which can be understood. An open mind is necessary to ascertain reasonable solutions; the ability to listen and the ability to ask the right questions in order to hear not only what a user is saying but also what he meant to say is essential.

This leads to another role that the CIS faculty must be willing to assume. This is the role of advisor. Many students are lead into the CIS major by the favorable prospects of job availability and large salaries. We continuously see students struggle in their course work because they lack the innate aptitude for logic and complex problem solving. These students need to understand their options and to be able to move in another direction without being made to feel inept. No matter how complete our curriculum is, some people simply do not have the aptitude for analytical functions.

APPROACHING AN MIS DEVELOPMENT PROBLEM

One primary consideration is the technique that the analyst uses in approaching an MIS development problem. In order to meet user needs, an MIS should be based on the decisions and resulting actions used in executing the job functions within the organization. In this structure the existence of any information in the system should be tied directly to the decision and actions taken by end users in accomplishing their job functions. By tactful inquiry and repeated research into each business function, the analyst must insure that needs are met in a timely fashion. (2)

The analyst can never view a design task from a strictly technical viewpoint. This can only result in a system that, though technically sound, does not provide needed information in timely or useable form. Hardware and/or software will not necessarily function according to user specification and even may not have been needed. The negative attitudes generated by such situations remain in the organization long afterward and thwart future efforts to provide information support. (1)

Students should be taught to assimilate data requirements into a logical database which depicts all hierarchical path and timing constraints required to provide adequate information processing. Input and output with similar content, hierarchy and timing should be grouped together. The primary concern in this assimilation process is to insure that all data is present that is needed by the end users in operations and decision making processes and that necessary paths exist for retrieval and update.

Along with the above development steps, techniquies must be taught for editing data and for providing efficient retrieval processing. Students should be made to realize that the most cost effective and effective implementation of an information systems can be manual, automated or a combination of the two as dictated by the circumstances and constraints at hand. They should learn that the physical means of implementation should be determined after logical design is complete and could include existing software packages or require the design and implementation of new software in order to meet the system needs.

We should provide the student sound structured design techniques using a variety of tools such as data flow diagrams, flow charts, Warnier-Orr diagrams, Nassi-Schneiderman charts and decision tables. We should apply these techniques in realistic models that actually exist in the business environment.

Complete design models including needs analysis, logical design, cost analysis, physical design, system implementation and system testing should be available for students to study. Class discussions centered around such models will remove the "text book" view of the "real world." Outside speakers involved in systems design also increase the exposure to realistic problem solving situations.

CHOOSING THE IMPLEMENTATION TOOLS

The role of an analyst at this point has changed significantly over the past few decades. With the availability today of micro computers, comprehensive database and query software packages for most computers as well as many other technological breakthroughs, an analyst must have much broader technical background in order to provide the most cost effective and efficient solution to the need for management information and operational information support. Though often bound by internal constraints such as existing equipment, available funds, time, user resistance and security requirements, the analyst must be able to suggest solutions within given constraints as well as to show management ways that an information system could be more effective and/or more cost efficient if certain restraints were modified or removed and other options considered. In this role the analyst becomes an advisor who must show management the strong points and shortcomings of each solution with tact. When final constraints are established and it is evident that they are not likely to change, the analyst must be able to select the optimal solution under those conditions and be able to make everyone involved understand what can be expected. Again, Communication skills and sensitivity to the impact of the proposed technology on the involved individuals are crucial. (1)

Through hands-on experience and technical reading our students should be made aware of available software packages, their applications and some of their limitations. One of the largest markets for analysts today is in the area of customer support of existing application software, particularly in the micro environment. A graduate with a familiarity with a variety of products and the ability to take new products and quickly develop a working knowledge of them will be in great demand.

COPING WITH USER RESISTANCE

In every organization there are individuals who refuse to cooperate in any effort from an outsider to attempt to understand his job functions or to improve the working environment or the techniques used in the job. The systems analyst must learn to take on varied roles in coping with this roadblock. If the problem is isolated, the analyst may try to work with these individuals or to circumvent them in order to gain the necessary information without alienating them. This is at times quite a feat and at times impossible. Experience, tact, patience and persistence are the best tools to work with in this situation.

Other times the analyst may have to be a "confronter" and lay the situation out to the appropriate individuals explaining why the current approach to the function is not the most efficient or effective or why the current approach simply will not get the job done. An analyst must learn to use tact, patience and extreme sensitivity in handling the situation, defending his position with facts. management must be made to understand that an impass exists and must be handled. The management decision made then becomes a constraint within which the best solution must be determined. (1)

OTHER PROBLEM - EXISTING FUNCTIONS

So far we have discussed only the situation of developing new management information tools without considering maintenance of existing support functions. Many data processing shops become buried under a near impossible load of response to immediate crises, constantly "patching" and "putting out today's fires." How can an analyst take time to analyze an organization and develop new or improved support when all of the available time is spent reacting to the current situation? Solving this dilemma requires management and communication skills used to sort out critical functions while providing time for development.

CONCLUSION

The role of the analyst is to be the fulcrum in the process of providing management information support in the business environment. Such an individual must, by nature, be analytical and logical while at the same time possess skills necessary to work with others and have the ability to organize and lead. Though we cannot directly provide an individual with all the skills necessary to become an effective analyst, we can help students identify their own natural aptitudes and provide a broad base of background skills which enhance innate abilities.

REFERENCES

(1) Eiland, Patricia and Honan, Ava, "The Role of the Systems Analyst in Developing Effective Management Information Systems and Tools," paper presentation Eastern Region Meeting of the Association of Human Resources Management and Organizational Behavior in Clearwater, Florida, May 1986.

(2) Eiland, Patricia, "Developing Effective Management Information Systems End Users," paper presentation at ISECON '85, Houston, Texas, October, 1985.

THE ELECTRONIC SPREADSHEET AND THE MARKOV CHAIN FORECASTING MODEL

Seyed-Mahmoud Aghazadeh, Ph.D.
Assistant Professor of Management
Information Systems
Department of Business Administration
State University of New York
College at Fredonia
Fredonia, N. Y. 14063

ABSTRACT

The aim of this paper is to show how to prepare forecasting by the Markov processes using an electronic spreadsheet.

Business decision makers use electronic spreadsheets more than any other type of software. The most popular electronic spreadsheet programs are Lotus 1-2-3, Super Calc, Visi Calc, Multiplan, and Target Planner Calc. These and many other similar programs are ideal for solving problems that can be represented by a matrix of interrelated values such as accounting worksheets. The programs are used to enter values in the cells of the matrix to perform calculations with the values, and to display the results.

The aim of this paper is to show how to prepare forecasting by the Markov processes using an electronic spreadsheet.

The Markov processes represent one of the best known and most useful classes of stochastic processes. The model can forecast a firm's sales at a specific future time if the total market sales for that period can be reasonably estimated.

STEP-BY-STEP PROCEDURE

The following is a step-by-step procedure for preparing forecasting with the Markov processes, using the Lotus 1-2-3 spreadsheet package. The specific example developed deals with brand switching.

Consider three brands, X, Y, Z, that all satisfy the same need and thus substitute for each other. The product is a convenience item that is bought frequently. The buyers will be faced from time to time with a buying decision that may result in a change from one brand to another.

Step 1: Consider Table 1 (transition matrix) which is similar to a spreadsheet with row numbers 1, 2, 3, and the column designation A, B, C, etc. The numbers in the second row (.90, .05, .05) reflect the research result that brand X retains .90 of its users, loses .05 of its users to brand Y, and loses .05 of its users to brand Z. The numers in the third row (.15, .80, .05) indicate that brand Y retains .80 of its users, loses .15 of its users to brand X, and loses .05 of its users to brand Z. The numbers in the fourth row show that brand Z retains .70 of its users, loses .10 of its users to brand X, and loses .20 of its users to brand Y. The same line of reasoning can be applied with respect to the columns of the transition matrix. For example, the numbers in column B (.90, .15, .10) indicate that brand X retains .90 of its users, gains .15 of brand Y's users and gains .10 of brand Z's users during a given time period.

The producers of the X, Y, and Z brands are concerned not with these (transition) probabilities but with their relative shares of the market for the purpose of establishing their advertising policies. More specifically, each producer wishes to determine the rate or probability that any particular customer will purchase its brand.

Step 2: Let's assume that further marketing research data indicates that brand X currently captures .50 of the market; brand Y, .40; and brand Z, .10. The interpretation of these numbers is that any particular randomly chosen user of the product X, Y, Z will have the (state) probabilities of $P_X(0) = .50$, $P_Y(0) = .40$, and $P_Z(0) = .10$ respectively in the initial period n = 0 (or let's say September 1). These values are shown in cells B5, C5, and D5 in Table 1.

Table 1

Probability of loss →

Probability of gain ↓

	A	B	C	D
1	Probability of retention	X	Y	Z
2	X	.90	.05	.05
3	Y	.15	.80	.05
4	Z	.10	.20	.70
5	Initial market shares	.50	.40	.10
6		.45	.025	.025
7		.06	.320	.020
8		.01	.020	.070
9	1st period market shares	.52	.365	.115
10		.468	.026	.026
11		.05475	.292	.01825
12		.0115	.023	.0805
13	2nd period market shares	.53425	.341	.12475
14				

Step 3: By utilizing the information about the initial conditions and transition pattern, management can forecast a state probability (the market share) at any stage of the evolutionary process. In order to estimate X's market share or the probability that brand X will be chosen (assuming that this choice will be made by retaining brand X or by switching from either brand Y or Z to X), the joint probabilities can be used. The joint probabilities for these results can be determined by multiplying the respective transition probability in column B of the transition matrix (.90, .15, .10) by the corresponding state probability or the current market shares in cells B5, C5, and D5 respectively. Summing these, the new state probability or X's market share can be obtained. For example, B6: +B2*\$B\$5 or .45 = (.90) (.50); B7: +B3*\$C\$5 or .06 = (.15) (.40); and B8: +B4*\$D\$5 or .01 = (.10) (.10).

X's market share for the first period (October 1) can be computed by using the command @SUM(B6..B8), which results in $P_x(1)$=.52 in cell B9.

Step 4: Determining Y's and Z's market shares for the first period (October 1): using the replicate command in the spreadsheet package, one can obtain the market share for Y and Z which would be $P_y(1)$ = .365, $P_z(1)$ = .115 in cells C9 and D9 respectively.

Notice that the market shares (state probabilities) for the first period have changed, indicating more customers will be buying brand X and Y than before and that brand Y's share of the market will drop.

Step 5: Multiple period estimates: using the market shares for period 1 in the same manner as before, the applicable state probabilities (market shares) for period 2 (November 1) can be forecast. For example,

B10: +B2*\$B\$9 or .468 = (.90) (.52);

B11: +B3*\$C\$9 or .05475 = (.15) (.365);

B12: +B4*\$D\$9 or .0115 = (.10) (.115).

X's market share for the second period (November 1) can be computed by using the command @SUM(B10..B12), which results in $P_x(2)$= .53425, and finally, using the replicate command, the market shares for Y and Z would be $P_y(2)$ = .341

and $P_z(2) = .12475$ respectively.

Continuing in this manner, the other benefit of the Markov process, namely its ability to determine points of equilibrium, may be obtained. By comparing the market shares during different periods, a decision maker can tell that the successive changes are smaller. This suggests that the market shares may be converging toward a set of constants. At that point the process reaches a steady state and will remain unchanged until external actions change the transition probabilities.

REFERENCES

(1) Ellis, D. "Forecasts can be Prepared with a Spreadsheet Package." Journal of Business Forecasting. Spring 1985, pp. 16-17.

(2) Forgionne, G. A. Quantitative Decision Making, Belmont, California: Wadsworth, 1986.

(3) Hillier, F. S., and G. J. Lieberman. Introduction to Operations Research, 3rd edition, San Francisco: Holden-Day, 1980.

(4) Holt, J. A. Cases and Applications in Lotus 1-2-3, Homewood, Illinois: Irwin, 1986.

(5) Kemeny, J. G., and J. L. Snell. Finite Markov Chains, New York: Springer-Verlag, 1976.

(6) Kress, G. Practical Techniques of Business Forecasting, Westport, Connecticut: Quorum Books, 1985.

(7) Martin, J. J. Bayesian Decision Problems and Markov Chains, Reprint 1967 edition, Melbourne, Florida: Robert E. Kreiger, 1975.

(8) Wagner, H. M. Principles of Operations Research with Applications to Managerial Decisions, 2nd edition, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

THE TEACHING OF LOTUS 1-2-3: A TUTORIAL FOR NON-COMPUTER MAJORS

William J. Dorin
Information Systems and Computer Programming
Purdue University Calumet

ABSTRACT

A number of the software packages, supplied with the computer literacy textbooks do not allow the student to use the full power of the software package. Since LOTUS 1-2-3 is the most popular spreadsheet, we determined that Management students at Purdue University Calumet would find this package most useful. The Introduction to Computer Programming for Management course covers many aspects of computer literacy, including the power and applications of LOTUS. This paper presents one assignment, broken into two parts, that utilizes a simple payroll system to illustrate the capabilities of LOTUS 1-2-3.

INTRODUCTION

"In a rapidly changing society it will be critical that students...(have) abilities to communicate in all forms..." (1) As we continue to grow in our information society one form of communications will be through the computer. It is because literacy refers to communication that we are able to apply a definition of computer literacy as the ability to use the computer to communicate information. However, literacy is more than awareness that computers exist. Literacy must include the ability to use the computer to solve problems. That is, an understanding of the "fundamental principles of computer processing." (2)

At Purdue University Calumet we offer computer literacy for management majors as a two-course sequence (CIS 204 and CIS 205). In the first course we cover the basics of word processing using Word, an electronic spreadsheet using LOTUS 1-2-3, computer terminology, and programming using Microsoft BASIC.

The second semester course continues with computer usage and understanding by incorporating the concepts of Management Information systems. The emphasis is on business systems analysis and design with the implementation of a case-study project using dBASE III.

The purpose of this paper is to describe the technique used to teach LOTUS 1-2-3, and to present part of a tutorial I wrote to teach the concepts of the spreadsheet to the students. There are many problems present in teaching any software package to a group of students. Although too numerous to expand on here, one problem is giving the student the "hands on" experience so vital in the learning process. The tutorial seems to be the best solution to this problem. This tutorial is not a Computer Assisted Instruction package, but a step by step set of written instructions to acquaint the student with spreadsheets and the various functions of LOTUS. Each spreadsheet, called a "Lab", steps the student through using the different commands, formulas, and functions. One of the assignments used in the tutorial is included in this paper.

The first task of the tutorial is to acquaint the student with the keyboard and the characteristics of the spreadsheet. The student learns the uses of the arrows, Home, End, PgUp, and PgDn keys of the numeric keypad. The cell, cell pointer, and command area of the spreadsheet are defined, and the student becomes familiar with the spreadsheet's row and column size by moving the cell pointer to various locations. The student becomes acquainted with the use of the menu commands by stepping through tasks like changing the size of a cell and saving or retrieving a file.

The labs vary in length of instruction and difficulty. In general, each student spends approximately two hours per lab. With a total of six labs, including the introductory demonstration, the student receives approximately 12 hours of instruction.

Lab 3 from the tutorial is illustrated in the following section. It is a two-part lab and is presented in its entirety. The purpose of the lab is to construct a small payroll spreadsheet. It introduces the student to the @IF function of LOTUS, as well as a sample of its Database aspects by including instructions for sorting data.

EXAMPLE ASSIGNMENT

LAB THREE A.

The purpose of this lab is to introduce you to the @IF function and the SORT command. This lab is a simple PAYROLL sheet.

1. At cell C1 enter
   Your Name SMALL BUSINESS COMPANY

2. At the following cells, enter the labels and data:

```
A3 ^NAME    B3 ^ID-NUM  C3 "RATE
D3 "HOURS   E3 "REG PAY F3 "OVER 40
G3 "GROSS PAY

A5  Dave    B5 ^1032  C5 5.35  D5  40
A6  Cindy   B6 ^2031  C6 6.35  D6  37.5
A7  Chet    B7 ^3821  C7 7.35  D7  42.5
A8  Sally   B8 ^4287  C8 9.50  D8  50
A9  Tom     B9 ^4891  C9 5.75  D9  40
A10 Bob    B10 ^6981 C10 3.90 D10 49.5
```

The up-arrow (^) signifies the label is to be centered in the cell and the double quote (") signifies the label is to be right justified.

3. Now you must enter some formulas and copy them to other cells.
   A. At cell E5 enter
      @IF(D5>40,C5*40,C5*D5)
   B. Type "/" to activate menu.
   C. Type C to COPY.
   E. At ENTER RANGE TO COPY FROM: hit <RET>.
   F. At ENTER RANGE TO COPY TO: type a period (.).
   G. Using down arrow key highlight column until row 10 (E5..E10).
   H. Hit <RET>.

4. At cell F5 enter
   @IF(D5>40,((D5-40)*1.5*D5),0)
   hit<RET>.
   Copy down to row 10 as above.

5. At cell G5 enter formula (E5+F5) hit <RET>. Copy down to row 10 as above.

6. Make these figures represent money. Do that by
   A. Typing "/" to activate menu.
   B. Type R for RANGE.
   C. Type F for FORMAT.
   D. Type a comma (,).
   E. Enter in C5..G10 for the range and hit <RET>.

7. Save and print spreadsheet. Remember what you named your spreadsheet because you will use it in LAB3B.

NOTE: The @IF function tests values called conditions and based on whether the condition test is true or false, it will execute one of two options or arguments defined in the function.

A condition is the comparing two cell values to each other or a cell value to a constant. The options or arguments would be executing one of two formulas or displaying a specific value.

LAB THREE B.

The purpose of this lab is gain experience in retrieving existing spreadsheets to continue working with data and using the SORT command.

1. Load in spreadsheet from Lab 3A.
   A. Type "/" to activate menu.
   B. Type F for FILE.
   C. Type R for RETRIEVE.
   D. Enter the name of your lab 3A and hit <RET>.

2. A. Enter your Name in cell A11.
   B. Enter the last four digits of your social security number in cell B11 (Id-Num).
   C. Give yourself a rate of pay of 11.50 and hour in cell C11.
   D. Enter in hours of 40 hours worked in cell D11.

3. A. Goto cell E11 and copy the formula from E10 to E11
   B. Goto cell F10 and copy the formula from F10 to F11
   C. Goto cell G10 and copy the formula from G10 to G11

4. A. Make sure the cell pointer is in A5 by using the GOTO command (F5 key).

5. To SORT the names and to assure that all data is sorted with the names, complete the following steps:
   A. Type "/" to activate menu.
   B. Type D for DATA.
   C. Type S for SORT.
   E. Type D for DATA-RANGE.
   F. At the DATA RANGE prompt, specify records to be sorted by typing a period (.) and using the arrow keys to highlight all the data so it will be sorted along with names. The range should be A5..G10. Hit <RET>.
   G. Type P for PRIMARY KEY and move cell pointer to cell A5 and hit <RET>. This tells LOTUS what data should be sorted.
   H. At the ENTER SORT ORDER (A OR D) prompt, type an A for ascending order. This means the data will be sorted alphabetically or numerically from low to high order.
   I. Enter G for GO and the sort will take place.

NOTE:- It is important that the data stays with each individual when the sort takes place. That is why steps E and F are important.

6. Save and print sorted version of this spreadsheet.

SUMMARY

The purpose of this paper was to present an example of how LOTUS 1-2-3 is taught to Management majors at Purdue University Calumet. By defining computer literacy and stating that the Management major needs these same skills, I discussed the Computer Information Systems department's two-semester course sequence for Management students.

This paper looked at an example assignment used to teach some of the concepts of LOTUS. The "lab", as it was called, steps the student through a simple payroll system. The @IF function was explained and then used by the student to determine if overtime pay was earned.

The second part of the lab demonstrated to the student that retrieval of the same worksheet was possible for updating data. It also had the student add his/her name to the data and then sort all the data in ascending order by name.

By giving the student step-by-step instructions and using business type problems, he/she is exposed to practical uses of LOTUS 1-2-3. Since each assignment should take about two hours to complete, the student receives 10 to 12 hours of LOTUS and "keyboarding" experience.

REFERENCES

(1) Smith, Peter, Dunn, Samual (July, 1985) Tomorrow's University: Serving the Information Society - Getting ready for the year 2000. Educational Technology p. 5-11

(2) D'Souza, Patricia V. (August, 1985) Computer Literacy in Today's Society. Educational Technology p. 34-35

THE ORGANIZATIONAL ROLE OF FOURTH GENERATION TECHNOLOGY

Leslie Aucoin
Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104-6366

Stephanie Barrett
Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104-6366

ABSTRACT

This paper presents the key results of an analysis of the organizational effects of a fourth generation information system. Although by conventional criteria, the system was considered successful, we observed errors of commission, errors of omission, and failures to resolve problems. We concluded the most significant impediment to effective exploitation of the technology was inadequate user education. Based on our analysis, we recommend four criteria for effective user education: progressivity, role-orientation, debiasing, and skills for management of IS.

INTRODUCTION

Organizations are facing severe problems in effectively applying information technology. There are many symptoms of the "software crisis," and trends indicate the situation will deteriorate unless fundamental changes are made. Many alternatives for coping with these difficulties are being explored. A promising approach, and one to which many organizations are turning, is the use of fourth generation tools (4GTs).

The potential benefits of employing 4GTs are many, and expectations are high. Impressive claims have been made regarding the ability of 4GTs to radically improve the economics of information systems development. If the promises of fourth generation technology are fully realized, the ramifications may extend far beyond the information systems arena. The effects on organizations may be dramatic.

Post-implementation studies of the effects of 4GTs are vital if we are to profit from our experiences and learn to manage fourth generation technology successfully. Unfortunately, evidence about the effects of 4GTs is largely anecdotal in nature. Few extensive, systematic post-implementation studies have been conducted and reported. To fill this gap, we have begun conducting such studies. Since the effects of a major technical intervention upon an organization can be subtle and far-reaching, we are using the case study method. This paper discusses the key results obtained in one organization.

THE CASE STUDY

BACKGROUND

One of the organizations that we studied is a prominent, privately funded biomedical research institution, BioRes (a pseudonym). During the last fifteen years, BioRes has experienced rapid growth. Currently, it receives income of about $20 million, predominantly in federal research grants. The combination of BioRes's rapid growth and the federal budget crisis have necessitated improved planning, budgeting, accounting, financial control, and reporting of information.

In 1981, BioRes commissioned a comprehensive review of its administrative information processing system. Its old, third generation system was batch-oriented and had evolved in a piecemeal fashion over a number of years. Data were stored in over 30 "flat" files and processed by more than 50 separate programs. The review identified a number of serious deficiencies of the old system. Batch orientation introduced processing and reporting delays. Also, the system was incomplete, inflexible, expensive to operate and change, and transaction rather than management oriented.

BioRes decided to purchase a new, custom-designed system that would be constructed using 4GTs. The heart of the new system was a modern database management system that included an integrated data dictionary, transaction management facilities, and a powerful ad hoc report writer. In addition, generalized tools for screen and dialog management, security enforcement and monitoring, and database auditing were employed.

RESULTS

System "Success"

Based upon the most commonly employed criteria for evaluating information systems, BioRes's new information system has been quite successful. First, the system has been used and useful. Second, its users are generally satisfied and exhibit positive attitudes about information technology. No evidence of resistance to the system was observed. Finally, the system is cost effective.

Concerns

However, further analysis reveals there are significant causes for concern. The new system is being used suboptimally. Although it clearly constitutes a great improvement over its predecessor, it has not begun to be used to its full potential. We observed the following:

(1) Errors of commission

(2) Errors of omission

(3) Failure to resolve problems

Errors of commission. These occur when

users behave inappropriately because they have misunderstood how to operate the system or how the system works. For example, BioRes users erroneously believed an important subsystem was not functioning properly. As a consequence of this misunderstanding, they engaged in unnecessary, time-consuming procedures for "working around" the perceived limitation.

Errors of omission. These are missed opportunities. They occur when users fail to realize that the system can be used to achieve a goal. At BioRes, contrary to our expectations, the inherent flexibility of the 4GTs was not truly exploited. For example, no significant changes to the system have been made although there are clearly areas where refinements are warranted. Also, the report writer is substantially underutilized. Tasks that could easily be done using the system are done by hand, and users seldom customize reports for their specific needs.

Failure to resolve problems. BioRes demonstrated a striking lack of aggressiveness in identifying and addressing problems. For example, BioRes expected the system to dramatically shorten the monthly closing process. Delays in receiving inputs from other departments, which has been acceptable in the past, migrated to the "critical path" with the advent of the new system and prevented this goal from being fully achieved. Despite widespread disappointment concerning closing delays, new procedures for receipt of external inputs -- which could easily have been instituted -- were not developed.

The lax approach to problem detection and correction also permitted misunderstandings to persist. For example, we mentioned that users incorrectly believed a major subsystem was malfunctioning. This misunderstanding would have been cleared up long ago if BioRes had taken an aggressive approach to correcting the perceived error.

RECOMMENDATIONS

The BioRes experience suggests the most significant impediments to effective usage of fourth generation technology are within users' minds. Users did not sabotage or resist the system for Machiavellian reasons. Rather, they were genuinely confused about the system and how to exploit its potential. Thus, user education emerged as a crucial issue.

For many years, training has been recognized as an important factor in the success of management information systems. Mere acknowledgement of its importance, however, has not been sufficient. A better understanding of the content of good training is sorely needed. Drawing on our experience with BioRes, we suggest criteria for user education. To distinguish our view of training -- which is quite broad and oriented toward learning -- from more restrictive approaches, we prefer to use the term "education" rather than "training." Table 1 indicates how the training criteria address the concerns identified earlier.

PROGRESSIVITY

Two kinds of errors can be made in teaching a complex system. If the focus is too narrow, users will learn to operate the system without gaining true understanding. If too much is presented too quickly, users will be overwhelmed and confused. Education should take place in stages. The stages should be progressive -- moving from easy to difficult, common to rare, and concrete to abstract. For example:

menu options ⟶ command interpreter
common task ⟶ unusual task
canned report ⟶ ad hoc report

Role-Orientation. Partly as a corollary to the principle of progressivity, education should be tailored to the various job roles within the organization. This philosophy of role-orientation also applies to user manuals. All too often, as was the case at BioRes, the information as individual needs to truly understand the system is scattered throughout a bulky, poorly-indexed manual.

Debiasing. Errors of omission are particularly insidious. They are difficult to predict and detect. Furthermore, they strike at the very heart of perhaps the greatest organizational benefit of 4GTs -- their ability to increase an organization's flexibility and adaptability.

Errors of omission are frequently fostered by faulty, implicit assumptions. To develop an effective strategy for combating errors of omission, it is helpful to know whether the erroneous assumptions can be classified according to any patterns or themes. At BioRes we identified too such themes: anchoring on past experiences and reliance on the "electronic filing cabinet" metaphor.

The users at BioRes seemed to suffer from what might be termed a "generation gap" or a "third generation mindset." In the absence of information to the contrary, they generalized from experiences with third generation technology. For example, they assumed that changes to the system would be time-consuming and expensive. Consequently, a "what we see is all we can get" attitude was adopted.

People relied heavily on the metaphor of the system as an "electronic filing cabinet." This notion was consciously reinforced by the system's designers who named the major retrieval option the "pull" command. One would expect the popular file cabinet metaphor to alleviate anxiety and facilitate understanding by appealing to a concept familiar to all. This appeared to be true. No one appeared to be "fearful" of the system, and people generally relied on the system to quickly find information about a single logical item (like a purchase order, a check, or an account). Unfortunately, the metaphor seemed to be taken too literally. Many people were not consistently able to "go beyond" this metaphor and recognize the potential of the computer for selecting information based on specified attributes, relating information from various "files," and performing arbitrary calculations.

In effect, users appear to be biased by both past experiences and the file cabinet metaphor. Therefore, we feel an important educational goal is "debiasing." Techniques that might be used include marketing (New! Improved! ...), horizon-expanding scenarios, and idealized planning.

SKILLS FOR MANAGEMENT OF INFORMATION SYSTEMS

Any computer-based information system is embedded in a larger system that consists of people and procedures in addition to hardware and software. Adding or modifying a computer-based information system is not a simple process like changing a tire. It requires mutual adjustments between the information system and its containing super system. At BioRes, no formal process existed for identifying and making these adjustments. Consequently, many problems were not resolved. This experience highlights the importance of teaching the skills needed for effective management of an information system. Particularly critical is the need for post-implementation audits with follow-up.

TABLE 1

CRITERIA	error of commission	error of omission	failure to resolve problems
progressivity	X	X	
role-orientation	X		
debiasing		X	
management of IS	X	X	X

CONCERNS

DPMA And The Student: A Challenge To The Instructor

Patricia Waters
Auburn University At Montgomery
Montgomery, Alabama

ABSTRACT

The purpose of "DPMA And the Student: A Challenge to the Instructor" is to encourage faculty to start a student chapter.

Student DPMA chapters, sponsored by local chapters, have operated under the guidance of DPMA International since 1968, yet many students are unfamiliar with DPMA.

Students in a campus chapter learn first-hand about data processing issues and careers from business professionals. They enjoy many benefits of regular DPMA membership such as Data Management magazine and participation in seminars and conferences.

Auburn University at Montgomery (AUM), started a student chapter in 1984 with the help of Dr. Carl McDevitt, Head of the Information Systems Department. He worked with his local chapter, DPMA International, the students of AUM, and others.

Chapter meetings were patterned after those of the sponsoring chapter. Social aspects of the chapter as well as educational programs were considered.

Faculty members, especially those involved with the sponsoring chapter, were encouraged to support the students while students were encouraged to attend meetings of the sponsoring chapter.

By year-end, the AUM Chapter had grown and prospered. Students had used their computer skills to document the year's activities in order to brief incoming officers. A budget surplus funded a year-end party.

Educators who bring a student chapter to their school will receive ample help from DPMA International and will find it a rewarding experience for everyone involved.

INTRODUCTION

Data Processing Management Association (DPMA) has existed since 1951 and currently has over 47,000 members. Educators and the business community have long recognized it as the most prominent organization in information processing for managers. But many students, even the best students, may never have heard of it.

The growth of the information systems field and increased use of computers in education, science government, and industry has led to a surge in the number of students registering in IS classes. At the same time, deans and department heads have found it difficult to keep pace with changing concepts in information storage and retrieval as well as the demand for more sophisticated software courses. To complicate matters further, knowledge in the computer field is often obsolete by the time a textbook is printed. So, continuing education is essential for the educator and student alike. This is where DPMA has played a significant role.

STUDENT CHAPTER BENEFITS

Since 1968 DPMA has provided the means for establishing student chapters in colleges and universities. Today this is the fastest growing segment of DPMA. Whether or not you are currently a member of DPMA, you have an opportunity to introduce your students to its benefits. Students in DPMA, through their link with a local sponsoring chapter, are in direct contact with local data processing managers, systems analysts, programmers, educators and vendors. They learn first-hand about data processing careers as well as specific job openings in the community. This involvement gives them an important advantage over students who are not affiliated with a professional organization.

DPMA provides many services and programs that can enhance the college experience. Data Management, the monthly magazine which is received by all members, provides students with articles on management, technical issues, and current trends. It is also a source of information on the seminars and conferences to be held regionally, nationally, and internationally. Students may usually attend these events at reduced rates. Registration for the 1985 International Conference in Houston, for example, was $25.00. There were several career development seminars given during a special student program.

The Certificate in Data Processing (CDP) is the widely acknowledged designation for information processing professionals. Serious students will have opportunities to prepare for the examination at the same time as the members of the local chapter. With membership, students have many other opportunities to earn recognition by working individually and with their chapter.

THE BEGINNING

In the Fall of 1984, Auburn University at Montgomery (AUM), took steps to develop a link with business through DPMA. It was felt that through contacts made in the business community the student could be taken from the classroom to the "real world" and back again. The confidence, skills and professional attitudes developed through the DPMA experience would prepare students to face the keen job competition they would encounter at graduation.

The head of the Information Systems Department, Dr. Carl McDevitt, played an important role in bringing DPMA to AUM. As a first year member of the Montgomery chapter, he found that the group was eager to share its experiences with his students. The chapter volunteered to sponsor the student group and he agreed to serve as the faculty advisor.

He then contacted DPMA International Headquarters seeking advice on starting a student chapter. A short but helpful bulletin titled "How To Start A DPMA Student Chapter And Coordinate Its Operation" is available from them. More specific guidance on student chapter operation is contained in the 47-page DPMA Student Chapter Handbook. Both may be ordered at no charge from DPMA International, 505 Busse Highway, Park Ridge, IL 60068-3191, (312) 825-8124.

Next, Dr. McDevitt verified that AUM met DPMA's requirements for starting a student chapter. To qualify for a chapter of DPMA, the school must be an accredited two or four year institution offering an Associate or Bachelors Degree in Information Processing or a related field.

The permission and support of the Dean of Business was then secured. This was necessary since a letter from the Dean was required by DPMA International confirming that the student chapter rules conform to those which govern other student organizations at the school. Endorsement by the Dean also helped the students expedite the process of obtaining a meeting place, open a bank account, and obtain audio-visual equipment for their programs.

STUDENT INVOLVEMENT

The year that a chapter if formed, it was discovered, is the most time consuming because of the dual responsibilities of setting up the structure of the organization and promoting its activities. At AUM, an IS student who had considered joining the local chapter as an individual student member, encouraged a few others to join her and volunteer their help. It was important to involve these students early in the planning so that the chapter truly represented the student point of view. A high grade point average or technical expertise were not considered important for students to participate. Those first students were selected to take part because they came forward enthusiastically and showed the willingness to spend the time needed to coordinate the effort on campus.

FIRST MEETING

The first meeting of the members of the new chapter was particularly important. Prior to the meeting the Student Government Association (SGA) was contacted for information regarding the school's requirements for setting up a club on campus. Since nearly 40 other campus clubs had already been established, there was very helpful information available. SGA provided the group with sample constitutions which, along with the suggested student constitution and bylaws provided by International in the handbook, greatly simplified that requirement.

The students created a simple announcement that stated that a meeting would be held in the Dean's conference room to discuss the establishment of a student chapter. It was posted in the Math and Business computer labs and on classroom bulletin boards. A copy of the announcement was sent to all instructors inviting them to attend and encourage their students to participate. By using the Dean's conference room, rather than a vacant classroom or the Student Center, a certain aura of importance and professionalism was created. This is critical since Student DPMA is a transitional phase between young people's student and professional lives.

At the initial meeting it became evident that the election of officers should take place as soon as possible so that decisions could be made concerning the chapter meeting time and place and what dues would be charged. Plans were also needed for the first educational presentation so that student interest would be sustained.

Dues were kept to a minimum. International currently requires $15.00 annual dues for student membership. The student chapter added a small amount for postage, poster and office supplies, and refreshments. Since all speakers at DPMA meetings appear without pay, although transportation and meals may be provided, the budget was minimal. A memento such as a coffee mug or paper weight is presented to each speaker in appreciation for their program.

SPONSOR INVOLVEMENT

The faculty advisor took the lead in the first few meetings, then maintained an important role as liaison with the sponsoring chapter once the new officers assumed more responsibility. His support, and that of the Dean, gave the students confidence and a sense of continuity, especially in the early stages of the chapter's growth.

The student officers and committee heads attended several meetings of the Montgomery chapter to observe the structure of the meetings and programs. The sponsoring group issued an open invitation for all student members to attend the program portion of their monthly meeting. They provided for three students to attend the dinner program for half the regular cost. Due to evening classes, busy schedules and jobs, many students were not able to participate in these events every quarter but eventually most of them had a chance to visit a local chapter meeting or event.

STUDENT MEETINGS

Because students had many evening classes and varied work schedules, time for meetings was a problem. Friday evening was selected and the meeting was kept strictly to one hour, although socializing following the meeting was promoted. Since 70% of the students at AUM work full or part-time, providing little opportunity for socializing, the social aspect of the club was considered important. Many students who were boisterous between classes and in the lab (that is, they acted like undergraduates) became surprisingly reticent at chapter meetings (that is, they began to act like thoughtful, young professionals). It was found that holding the meetings in a small conference room or office rather than a classroom was more congenial. Refreshments were served before the meeting began to help promote an informal atmosphere. The development of social skills was seen as an important benefit of the chapter.

Students were encouraged to invite prospective members to every meeting. Pamphlets provided by International such as "This is DPMA" and "Pot O' Gold" were available on the refreshment table along with membership applications.

STUDENT PROGRAMS

The student chapter needed guidance in putting together their first few programs. By asking their chapter sponsors, they discovered that speakers were available from the chapter, campus and local business community. The Chamber of Commerce provided a speaker's bureau brochure. Video seminars and films were available from International.

Speakers invited by the sponsor chapter often agreed to present a program for the student group. In one instance the students obtained an excellent speaker on fourth generation languages. He was later asked to present his program to the sponsoring chapter and it was one of the highlights of the year!

Several faculty members at AUM were members of the sponsoring chapter. The students invited them to each meeting and social event along with instructors in the Information Systems and Decision Science Department. These visitors gave credibility to the meetings, offered helpful suggestions, and recommended speakers and topics.

One of the most successful student programs was an informal gathering with university Information Systems faculty. It was held in the Student Lounge with comfortable seating arranged in conversational groupings, dimmed lights and music. Students talked with the instructors about curriculum and careers and enjoyed refreshments. Instructors discovered who their serious students were and students benefitted from getting to know their instructors outside the structure of the classroom. Instructors were later invited to give programs on some of their favorite topics or research studies.

LOCAL CHAPTER PROGRAMS

Many local chapter programs were exceptionally meaningful for students who attended. Rear Admiral Grace Hopper, the 79 year old computer pioneer, was invited to speak before DPMA members, students and guests. It was a major event for students to meet an important historical figure who is featured in their computer textbooks. Many had their texts autographed or had photographs taken with Admiral Hopper.

Another local chapter program focused on the five area universities and their Information Systems curricula. Since many members of the local chapter were contemplating becoming students again or possibly instructors, the topic was of interest to all.

During a local vendor showcase, hosted by the local chapter, students set up displays, manned booths and represented AUM with a display and literature from the Admissions Office.

YEAR ONE

By the end of the first year, students had put the chapter roster in a data base, used word processing to prepare minutes of the meetings, and prepared posters using graphics software. A permanent bulletin board was established near the Computer Lab for announcements, cartoons, classified ads, and pertinent articles from Data Management.

The chapter found itself with a healthy budget surplus. Many supplies and refreshments had been donated and the officers had been very conservative in spending the chapter funds. A small amount was retained to ensure a smooth transition for the upcoming year. The remainder was spent to provide a "Fun and Games" pizza party after final exams. The games were the computer games that no one had time to play during the school quarter.

Notebooks had been purchased to organize the minutes of the meetings, financial records, and all correspondence send and received during the year. All information had been carefully kept to aid the incoming officers. Since some of the current officers and members had graduated, it was important that continuity be maintained through this documentation. The notebooks were presented to new officers at a special planning session attended by outgoing and incoming officers.

CONCLUSION

The Student Chapter at AUM has grown to over 30 members who are carrying on the tradition by preparing for the second annual "Fun and Games" pizza party.

The founding officers have graduated and taken responsible positions in their field. Some have joined the Montgomery Chapter where the faculty advisor is now President.

The local chapter now sponsors three student chapters. It has set up a scholarship fund to which its members contribute at each monthly meeting. Each year it presents an award in each school it sponsors- the Outstanding Data Processing Student of the Year. The student receives a small cash prize and has his or her name engraved on a plaque which is prominently displayed in the school.

The DPMA Student Chapter Handbook best describes what DPMA has meant for students and faculty at AUM:

> "DPMA means belonging to a group of like-minded individuals joined together for the purpose of advancing their knowledge and skills, their careers and their profession."

You can contribute to the development of your business students in specific, meaningful ways by bringing a DPMA student chapter to your campus. By involving them with businesses and business people, you will help students focus on career options which use their newly developed skills.

Through interaction with business, educators will be better able to determine what skills and concepts to teach. As schools and businesses form closer ties, all professionals in information processing will benefit. DPMA supports the bridge you build between the business and academic community.

TOWARD THE CONCEPT OF INFORMATION MANAGEMENT SCIENCE FROM MINOR PROFESSION TO MAJOR SCIENCE

Andrew S. Targowski (616) 383-1965
Earl E. Halvas (616) 383-1908
Department of Business Information Systems
College of Business
Western Michigan University
Kalamazoo, MI 49008

ABSTRACT

Training of graduates in information management science in academia is not keeping pace with industrial practices, as evidenced by (a) technological tools found in the two environments, (b) computer languages being taught vs. used, (c) the level of systems training incorporated into programs, and (d) the lack of ongoing communications between academia and the business/industrial environments.

Data processing as a profession is relatively young, and has not in itself emerged as a major profession, but is dependent upon other academic disciplines for its identity and direction. This is partially due to a lack of rigourous rules and concepts needed to form a foundation for a more specific science.

Specific statements of the mission, purpose, and goal of IM/S ought to be incorporated into the IM/S organizational structure, which would ultimately form the basis for an information school in management theory.

Of considerable impetus to such development would be adapting a more generic term, "informatics," for communicating information management science data among computer specialists from computer engineering, computer science, and information management science.

SYMPTOMS OF PROFOUND CRISIS IN THE DP INDUSTRY AND IN ACADEMIA

Nothing is wrong when a crisis is recognized in a science-based environment, such as the data processing industry. Any crisis is a healthy symptom of progress, since in the crisis state there are usually contradicting forces promoting a status quo or progressive situation. These contracting forces help define the crisis; otherwise, the crisis might never be recognized. Hence, James Martin's definition of the crisis situation in DP as "the period before the French Revolution" when lots of people said, "We've got to change" is less dramatic in a sense of Martin's message(1) and more profound in a sense of the hypothesis presented in this paper.

Martin in "An Information Systems Manifesto" dramatically declares that "it is a crime to train any new graduate today in COBOL or BASIC"(2) and that universities should train their graduates in the new tools of fourth generation languages that shorten the system development life-cycle. This message is positive, since modern and advanced tools should be supported. However, in information processing, which produces knowledge, there is no saturation point in the area of tools development. According to Martin, the current crisis in the DP industry is a permanent one. There will always be situations where industrial applications of new information tools will be far ahead of academic instruction. In the past we experienced at universities a shortage of COBOL compilers, which we now have in full supply. Today we observe the shortage to be the case with fourth generation languages. In the future it will be true of the tools of the fifth generation for informational problem solving. It is, therefore, a utopian hope that universities will educate specialists with superior knowledge and experience, which will be far ahead of industrial practice.

There exists, however, a more profound crisis in DP, in computer manufacturing industries, and in academia, which is not identified by Martin. This crisis is a lack of communication between the DP industry and academia, and is especially acute in business schools/colleges that have a relatively undeveloped computer- and particularly software-oriented academic environment.

An example of this miscommunication is seen in James Martin's roundtable teleconference with academic computer science representatives.(1) When Martin charges them with his message and they reply: "MIT students do not write in COBOL, BASIC, or Ada. They usually write in C or in LISP." Martin, with no additional comment, switches to application-oriented issues of developing data bases. His "accusation" and "crisis warnings" should eventually be addressed not to computer science departments in arts and sciences colleges, but to MIS/IS/CIS departments in business schools/colleges. However, Martin, who invokes the storm, appears not to be aware of that need.

Computer science departments are very well established in the academic environment. They have the longest tradition among computer-oriented departments, having been established in 1960. Some of those departments also are well supported by the computer industry, which recognizes the long term value and effect of such support. Since the ACM members in comparison to DPMA members are exceptionally articulated, it was natural that Martin had chosen computer scientists for that roundtable teleconference. It was also "typical" that during the discussions the computer scientists spoke on behalf of MIS/IS/CIS departments.

The MIS/IS/CIS departments (later called CIS) are very young; they appeared on the stage about 15 years after CS departments. Of considerable concern to the end users is the fact that CIS departments tend to educate only COBOL system analysts-programmers, not software or systems scientists.

The CIS curriculum produces application systems advocates whose knowledge is based upon academically simulated activities, and is modified by the trial and error of individual practice. By contrast, CS departments educate computer professionals rigorously prepared in technical problem-solving based on specialized scientific knowledge.

DP AS A MINOR PROFESSION

In a major book on the professions, published in 1970, Wilbert Moore embraced Alfred North Whitehead's distinction between a profession and an avocation. An avocation is "the antithesis to a profession."(4) Furthermore, Nathan Glazer recognizes the "major," "near-major," and "minor" professions.(5) Medicine and law are recognized by the author as major professions. Engineering and economics are close behind these. The major professions are grounded in systematic, fundamental knowledge, of which scientific knowledge is the prototype. They are distinct from such "minor" professions as social work, librarianship, education, and town planning,(6) which lack specificity in their foundation.

In attempting to categorize the DP profession, one becomes acutely aware that it is based on less rigorous rules, and is dependent on other academic disciplines such as economics, finance, management (which are also nonrigorous), and on engineering. The DP profession suffers from shifting, ambiguous ends, from unstable institutional contexts of practice, and from the huge influence of computer vendors; DP is therefore unable to develop a base of systematic, scientific, professional knowledge--or at least has not yet done so.

The state of the art of the DP profession is problem-solving, unsupported by specialized scientific knowledge. In the development of human resources for DP, the emphasis is put on the improvement of "skills" to solve concrete problems. An example might be developing the ability to generate code without conventional programming. Martin's corpus delicti examples of "skills" in use of theory and methods to solve DP practical problems should come later, after the CIS student has learned the basics of information management science. Students cannot apply skills of CIS development and maintenance until they have learned scientific knowledge. Skills per se are an ambiguous, secondary type of knowledge. Shon's position is that there is something disturbing about calling them "knowledge" at all.(7)

The DP profession is oriented toward means, not ends of economic action. DP specialists look for the answer "How to do it?" before they have answered the primary question "What to do?" Consequently, business schools/colleges produce COBOL analyst/programmers who have a limited knowledge of operational and strategic management, quantitative methods, and systems science. We must remember, however, these graduates are entering an industrial environment which is being converted into a "programmerless" one. When "ends" are confusing and conflicting as they are in DP, there either are no problems to solve, or everything is a problem! "Problem setting" is a process in which, interactively, we identify the things to which we attend, and <u>frame</u> the context in which we will attend to them.(8) This is the main promise of the systems approaches, which in DP are limited to the structured design. The structured approach is useful for the organization of the development process. But when this approach is applied to "framing" the nature of an application system, the correctness of the system is lost, since the two-dimensional structure flattens the system.

Techniques used in the DP profession are esoteric. They come from an unfixed content of professional knowledge. One role of the DP profession is to design (or engineer) the scope of information, application software, and computer configurations. However, since DP specialists are educated at business schools/colleges, they are not equipped with the more rigorous knowledge of engineering. Furthermore, there is little research undertaken in schools/colleges of business that would change the situation. A flood of too easily written textbooks creates another obstacle to the development of the DP profession.

EMERGENCE OF A NEW PARADIGM

We have to look at the reality of information management (IM) using a new paradigm.(3) We have to shift to a new set of references for the DP specialist. This new paradigm: the theory and foundation of IM must be significantly advanced of current practices and must be based on rigorous knowledge. This would make possible the movement of the DP (minor) profession to the status and criteria of the major profession. The first consequence of this upward mobility is that the CIS curriculum at business schools/colleges becomes de facto a curriculum of business engineering, much the same way as industrial engineering emerged as a cross-sectional curriculum in engineering colleges.

The CIS curriculum must incorporate the engineering approach to information management, i.e. the building of information systems via computer tools. This would augment the existing behavioral approach, in which we teach packages for user applications, and the scientific or problem-solving approach, wherein we incorporate mathematics-based formulas and algorithms.

One should expect some change in the student admission requirements and policies for programs in business systems engineering; more emphasis should be placed on the analytical capabilities of prospective students. The proposed name, business systems engineering, may sound out of place in business colleges. However, the same process takes place at the colleges of arts and sciences, i.e., the computer science departments are actually involved in system software engineering.

The second consequence of the upward mobility of the DP profession is that the scope of CIS or IM science curriculum should be developed after a scope of IM science has been determined, even roughly. Otherwise, the DP profession would have a difficult time applying standards to evaluate the quality of the curriculum, through the evaluation of CIS graduates in the marketplace.

The third consequence is that the DP profession, being a "design-in-the-social-context"-oriented discipline, must be equipped not only in problem-solving knowledge and skills but also in "problem-setting" capabilities. "Problem-setting" is problem solving within a broader context of reflective inquiry. Knowing in-action is the present paradigm in the DP profession; however, the aforementioned shift will focus the attention on reflection-in-action.(9) In other words, the "common sense" (as a tool of knowing in-action or learning through practice) is extended to the experience of surprise, a tool of reflection in-action. As Shon states: "When intuitive, spontaneous performance yields nothing more than results expected for it, then we tend not to think about it."(10) This broader context of the DP profession's interest must come from the more rigorous and hierarchical knowledge that has to be developed in the scope of information management science.

MISSION AND CONTENT OF INFORMATION MANAGEMENT SCIENCE (IM/S)

One mission of IM/S is to increase the scope of information, knowledge, and wisdom at all levels of CIS use--for the benefit of individual citizens, organizations, and particular users involved in information handling and decision making.

A purpose of the IM/S use is its expected contribution to the improvement of individual users' and organizations' performance until some synergetic results can be achieved.

A goal of IM/S is to help in developing human intellectual capabilities, so he can use them for problem setting and problem solving in society and the economy.

A full structure of IM/S is shown in Figure 1.

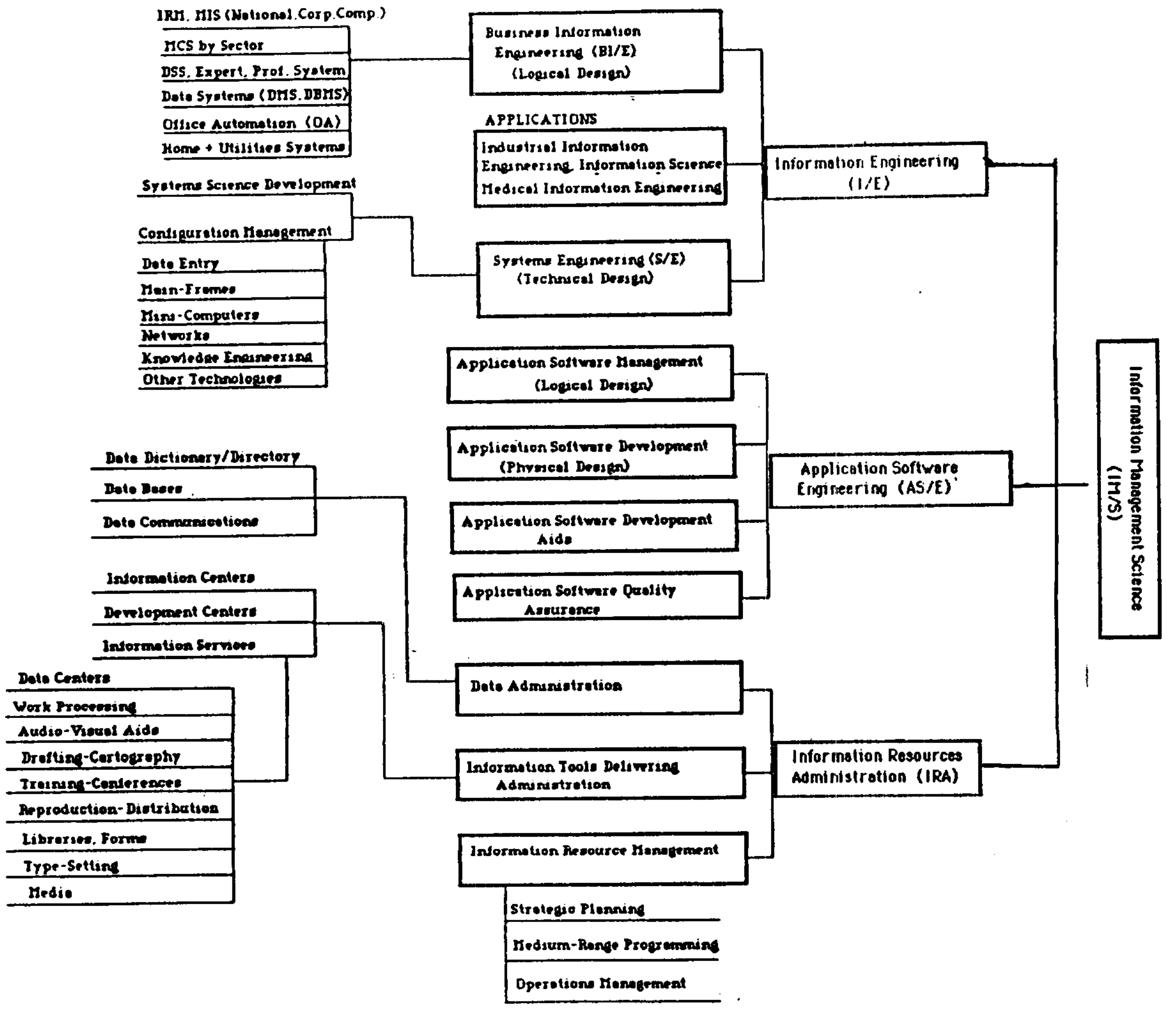

IRM. MIS (National.Corp.Comp.)
MCS by Sector
DSS. Expert. Prof. System
Data Systems (DMS.DBMS)
Office Automation (OA)
Home + Utilities Systems
Systems Science Development
Configuration Management
Data Entry
Main-Frames
Mini-Computers
Networks
Knowledge Engineering
Other Technologies
Data Dictionary/Directory
Data Bases
Data Communications
Information Centers
Development Centers
Information Services
Data Centers
Work Processing
Audio-Visual Aids
Drafting-Cartography
Training-Conferences
Reproduction-Distribution
Libraries, Forms
Type-Setting
Media
Business Information Engineering (BI/E) (Logical Design)
APPLICATIONS
Industrial Information Engineering, Information Science Medical Information Engineering
Systems Engineering (S/E) (Technical Design)
Application Software Management (Logical Design)
Application Software Development (Physical Design)
Application Software Development Aids
Application Software Quality Assurance
Data Administration
Information Tools Delivering Administration
Information Resource Management
Strategic Planning
Medium-Range Programming
Operations Management
Information Engineering (I/E)
Application Software Engineering (AS/E)
Information Resources Administration (IRA)
Information Management Science (IM/S)

Figure 1. A Structure of Information Management Science - 1986

CONCLUSION

Having defined the mission, scope, and methods for information management science, it should be easier to identify the necessary research, knowledge, curriculum, and professional standards.

BIBLIOGRAPHY

1. "DP and Academia: The Communication Gap," Communication of ACM, Vol. 28, No. 3, 1985, pp. 256-262.
2. "Excerpts from: An Information Systems Manifesto," Communication of ACM, Vol. 28, No. 3, 1985, p. 263.
3. Kuhn, Thomas S. The Structure of Scientific Revolutions. Chicago: The University of Chicago Press, 1970, p. viii and p. 85.
4. Shon, Donald A. The Reflective Practitioner. New York: Basic Books, Inc., 1983, p. 22.
5. Shon, p. 23.
6. Ibid., p. 23.
7. Ibid., p. 28
8. Ibid., p. 40
9. Ibid., p. 54.
10. Ibid., p. 56.
11. Targowski, Andrzej. Informatyka. Warsaw: PWE, 1980, p. 20.

MASTERY OF CONTROL BREAKS REPORTING LOGIC

George P. Grill, Ed.D.
Department of Information Systems & Operations Management
School of Business & Economics
University of North Carolina at Greensboro
Greensboro, NC 27412-5001

ABSTRACT

Control break reporting logic is one of the complex concepts for the beginning COBOL student. This paper presents a modular approach to teaching control break logic. Each module performs a specific processing function of control break reporting. An analysis of developing good control break programs is based upon the program development cycle.

INTRODUCTION

Introducing control break reporting logic is usually the first abstract concept begining COBOL students have difficulty in mastering. Control break logic is founded on the premise that basic COBOL concepts--file structure; input/output operations; arithmetic operations, especially accumulating totals; comparing operations; and nested IF statements--are learned and thoroughly understood. With this foundation, the logic of control break reporting can be more easily understood and quickly mastered by the beginning COBOL student before designing and writing programs requiring control breaks.

Control Break Logic

Prior to the introduction of control breaks, data was read and processed from an input record to print a detail line. This type of printing is often referred to as detail printing--one printed line from one input record. Basic programs involving input/output only, accumulating totals, and comparing are often detail printed. A final total line is usually printed on the report.

Generally, business applications require in addition to a final total line other intermediate totals or sub-totals. Printing intermediate totals are controlled by the program as either detail printing or group printing. Group printing or indication suppresses the printing of repetitive data, such as employee social security number, employee name, and employee number. Group printing requires more structured programming logic. Also, group printing improves clarity, readability, and brevity of the report.

Control Break Reporting

A control break report is a summary report containing intermediate and final totals. Selected fields (control fields) within the input record cause the printing of total lines. When these control fields change, a control break occurs and a total line is printed. There may be a single-level control break or multiple-level control breaks.

Comparisons are initially made on the control fields from major to minor designations. When comparisons are not equal, printing of intermediate totals are from minor to major. For example, if the three control fields are country (major), state, and city (minor), totals for the specific city would be printed first, then state totals and finally country totals. The major field/key is the highest level of totals and the minor keys are the lowest levels.

Control break programs are based on accumulators. When there is no control break, the specific input data is accumulated. If control breaks occur, the value of the lowest-level accumulator is printed. The accumulator's value is then added to the next higher level accumulator, finally generating a final total accumulator when all the records have been read. Thus, a minor accumulator is added to an intermediate accumulator. The intermediate accumulator, if necessary, is added to the major accumulator. The major accumulator is then added to the final accumulator.

When the value of an accumulator is printed, then that accumulator must be reset to zero. However, be certain that the accumulator has been added to the next higher level accumulator before resetting the accumulator to zero.

Control Break Programming Modules

To illustrate printing totals generated from group printing processing, the following program creates an office payroll summary report. The input of the program is a file of office personnel records. The format of these records is illustrated in Figure 1. The input data records containing the division number, employee number, employee name, weekly gross income, and weekly payroll deductions, are shown in Figure 2. Figure 3 presents the print chart output layout of the Payroll Report.

A control break occurs when the value in a specific field in the record changes from the value found in that field in the previous record. When the employee number changes, a group indication detail line containing the division number (unless already printed), employee number, employee name, monthly gross income, and monthly net income is printed.

When the division number changes, the total monthly gross and total monthly net incomes for the specific division are printed. After all records have been processed, a final total line is printed.

The following programming modules illustrate the processing logic needed to design and code the office payroll summary program (Figure 4):

Module: A000-Main-Routine. After the first record has been read and before the record is processed, move the division number and employee number to compare areas. The employee name is also moved to a save area as the employee number must correspond with the employee name. The values in the compare areas will be used later to determine if a control break has occurred.

Module: B100-Process-Pay-Records. The first detail record is then processed as follows:

1. Compare the division number in the input area to the division number in the compare area. When they are equal, no control break has occurred. However, when they are not equal, a control break has occurred and a division total line is to be printed (see Module D200-Print-Division-Line).
2. Compare the employee number in the input area to the employee number in the compare area. When they are equal, no control break has occurred. However, when they are not equal, a control break has occurred and an employee detail line is to be printed (see Module D100-Print-Detail-line).
3. Add the employee's weekly gross income to the employee's gross income accumulator and add the employee's weekly deductions to the employee's weekly deductions accumulator. This addition processing is executed for each record read. These accumulators are used to accumulate the monthly gross income and monthly dedutions for all of the records processed with the same employee number.
4. Read the next record into the input area. Then, repeat Steps 1-4.

Module: D100-Print-Detail-Line. When the employee number in the input area is not equal to the compare employee number, the following coding should take place:

1. Move the employee number from the compare area to the detail line.
2. Move the employee name from the save area to the detail line.
3. Move the employee gross income accumulator to the detail line.
4. Compute net income and move to the detail line.
5. Add employee gross income accumulator to division gross income accumulator.
6. Add employee weekly deductions accumulator to division weekly deductions accumulator.
7. Print the detail line. (The division number was moved to the detail line from the heading module (see Module C100-Print-Headings).
8. Move the employee number in the input area to the employee number in the compare area.
9. Move the employee name in the input area to the employee name in the save area.
10. Reset the employee gross income accumulator to zero.
11. Reset the employee weekly deductions accumulator to zero.
12. Process the record stored in the input area (see Module B100-Process-Pay-Records).

Module: D200-Print-Division-Line. When the division number in the input area is not equal to the compare division number, the following coding should be made:

1. Move the division number in the compare area to the division total line.
2. Move the division gross income accumulator to the division total line.
3. Compute the division net income and move to the division total line.
4. Add the division gross income accumulator to the total gross income accumulator.
5. Add the division weekly deductions accumulator to the total weekly deductions accumulator.
6. Print the division total line.
7. Move the division number in the input area to the division number in the compare area and to the detail line.
8. Reset the division gross income accumulator to zero.
9. Reset the division weekly deductions accumulator to zero.
10. Process the record stored in the input area (see Module B100-Process-Pay-Records).

Module: E100-Print-Final-Total-Line. After all records have been read and processed, the control break processing for the last group of records must be executed. After the last control break is processed, the final totals are processed and the final total line is printed (see Module A000-Main-Routine).

Control Break Problem Analysis

Figure 2 shows the monthly gross income and net income of office personnel by division and employee numbers. The major field is the division number; the minor field is the employee number. The employee name is saved to correspond to the employee number in the compare area. Final totals of gross and net incomes are printed.

To produce the office payroll summary report, the five phases of a program development cycle should be carefully analyzed before attempting to write the control break program.

Phase 1: Program Specifications. What is the problem of this program? Is it a single-level or multiple-level control break program? Is it detail printed or group printed? Do you understand the input record format? Is the format of the output report clear? Are all the program specifications included? Are you familiar with the input record layout as shown in Figure 1 and with the output layout as shown in Figure 3?

Phase 2: Program Design. Based upon the input records, what processing must be applied to produce the report? What specific modules are needed? Are your modules depicted in a program structure chart?

Phase 3: Program Processing. To identify the step-by-step logic to be performed, have you prepared a structure chart, flowchart, or pseudocode to solve the problem? Have you performed a walkthrough of your logic?

Phase 4: Program Coding. Based upon Phases 1-3, are you ready to code the program as shown in Figure 4?

Phase 5: Program Debugging/Testing. Have you compiled your program? Are all your syntax errors corrected? Does the test data as shown in Figure 2 produce an accurate report, as illustrated in Figure 5?

CONCLUSIONS

Developing good program structure is based on good program design. Program logic specifies the order in which instructions will be processed and executed. A properly designed modular program is easily coded and implemented.

Control break reporting concepts can be readily understood and mastered if the COBOL student carefully develops the logic behind control breaks. With this mastery, the student can continue to learn more advanced and complexed COBOL concepts. The program development cycle should be followed in designing and writing programs.

REFERENCES

Brown, Gary D. Advanced ANS COBOL With Structured Programming. New York: John Wiley & Sons, 1977.

Medley, Don B., and Eaves, Ronald W. Programming Principles With COBOL I. Cincinnati: South-Western Publishing Co., 1984.

Shelly, Gary B., Cashman, Thomas J., and Forsythe, Steven G. Structured COBOL. Flowchart Edition. Brea, California: Anaheim Publishing Company, Inc., 1985.

Stern, Nancy, and Stern, Robert A. Structured COBOL Programming. 4th ed. New York: John Wiley & Sons, Inc., 1985.

Yourdon, Edward, and Constantine, Larry L. Structured Design. 2d ed. New York: Yourdon, Press, 1978.

Figure 1: INPUT RECORD FORMAT FOR OFFICE PAYROLL SUMMARY REPORT

Position	Field
1 - 3	DIVISION NUMBER
4 - 6	EMPLOYEE NUMBER
7 - 26	EMPLOYEE NAME
27 - 31 (2 decimal places)	WEEKLY GROSS INCOME
32 - 36 (2 decimal places)	WEEKLY PAYROLL DEDUCTIONS
37 - 80	NOT USED

Figure 2: TEST DATA FOR OFFICE PAYROLL SUMMARY REPORT

```
100123CASE, TRELLES GLENN 6000004000
100123CASE, TRELLES GLENN 7000010000
100123CASE, TRELLES GLENN 6000004000
100123CASE, TRELLES GLENN 8500012500
100145ICARD, REBECCA MABEL3000006000
100145ICARD, REBECCA MABEL3000006000
100145ICARD, REBECCA MABEL3000006000
100145ICARD, REBECCA MABEL3000006000
100167LACKEY, FRED SAMUEL 4500005000
100167LACKEY, FRED SAMUEL 4500005000
100167LACKEY, FRED SAMUEL 4500005000
100189MANN, GLORIA MELISSA7750012500
100189MANN, GLORIA MELISSA7750012500
100199NIKONGE, BESSIE ANNE9654312345
200223CHENG, EDDY HOWARD  5000004000
200223CHENG, EDDY HOWARD  5000004000
200223CHENG, EDDY HOWARD  5000004000
200223CHENG, EDDY HOWARD  5000004000
200235FEW, AMANDA BARBARA 7802009473
200251HILL, ROBERT PHILLIP2093700800
200257MULLEN, TOM PAUL    3955006241
200257MULLEN, TOM PAUL    3955006241
200257MULLEN, TOM PAUL    3955006241
200257MULLEN, TOM PAUL    3955006241
200269PONS, TIM FRANK     5800011030
200269PONS, TIM FRANK     5800011030
300458COLLINS, KATHY WEBB 9999909999
```

Figure 3: OUTPUT LAYOUT OF OFFICE PAYROLL SUMMARY REPORT

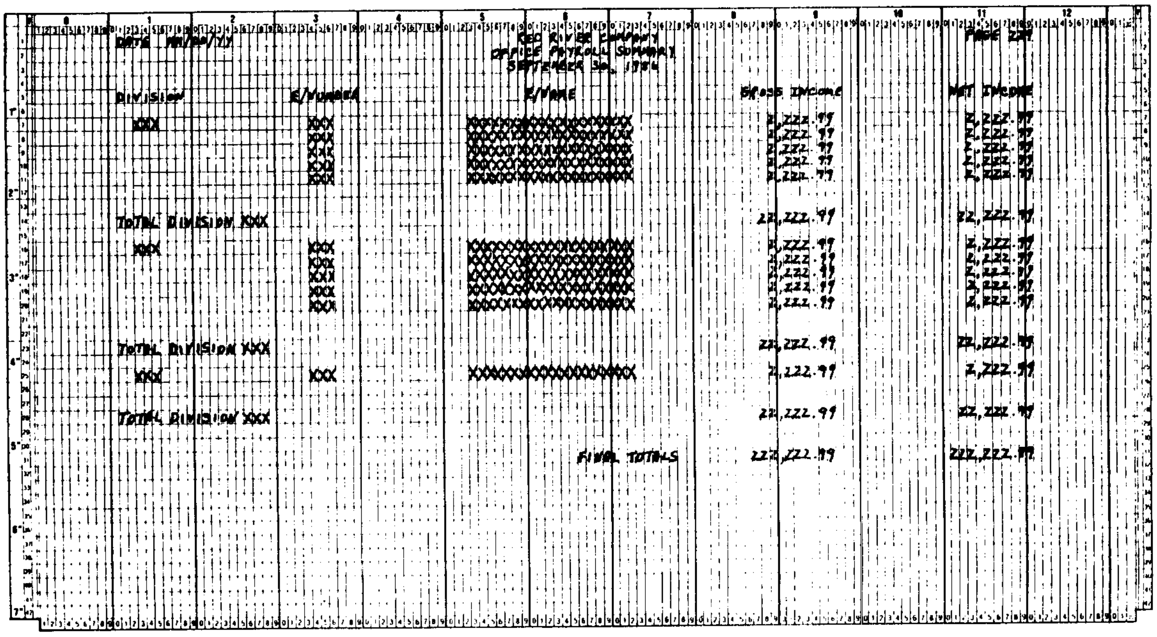


Figure 4: SOURCE PROGRAM TO PRODUCE OFFICE PAYROLL SUMMARY REPORT

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  PAYROLL-PROGRAM.
 AUTHOR.  GEORGE P. GRILL.
 INSTALLATION.  UNC-G.
 DATE-WRITTEN.  SEPTEMBER 15, 1986.
*
*****************************************************************
*  THIS PROGRAM READS A FILE ON OFFICE PERSONNEL, DETERMINES THE
*  NET INCOME FOR EACH EMPLOYEE, AND ACCUMULATES THE GROSS PAY AND
*  WEEKLY DEDUCTIONS FOR EACH EMPLOYEE IN EACH DIVISION.  A FINAL
*  TOTAL IS ALSO PRINTED.
*****************************************************************
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  VAX-11.
 OBJECT-COMPUTER.  VAX-11.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT PERSONNEL-FILE-IN
       ASSIGN TO "CLASSDIR:PAYROLL.DAT".
     SELECT PAY-FILE-OUT
       ASSIGN TO "PAYROLL.RPT".
*
 DATA DIVISION.
 FILE SECTION.
 FD  PERSONNEL-FILE-IN
     LABEL RECORDS ARE OMITTED
     RECORD CONTAINS 80 CHARACTERS
     DATA RECORD IS PERSONNEL-RECORD-IN.
 01  PERSONNEL-RECORD-IN.
     05  PF-DIVISION-NBR                  PIC 999.
     05  PF-EMP-NBR                       PIC 999.
     05  PF-EMP-NAME                      PIC X(20).
     05  PF-WEEKLY-GROSS-INCOME           PIC 999V99.
     05  PF-WEEKLY DEDUCTIONS             PIC 999V99.
     05  FILLER                           PIC X(44).
 FD  PAY-FILE-OUT
     LABEL RECORDS ARE OMITTED
     RECORD CONTAINS 132 CHARACTERS
     DATA RECORD IS REPORT-LINE.
 01  REPORT-LINE                          PIC X(132).
*
 WORKING-STORAGE SECTION.
 01  HEADING-LINE-1.
     05  FILLER                           PIC X(10)       VALUE SPACES.
     05  FILLER                           PIC X(6)        VALUE "DATE  ".
     05  HL-1-MONTH                       PIC XX          VALUE SPACES.
     05  FILLER                           PIC X           VALUE "/".
     05  HL-1-DAY                         PIC XX          VALUE SPACES.
     05  FILLER                           PIC X           VALUE "/".
     05  HL-1-YEAR                        PIC XX          VALUE SPACES.
     05  FILLER                           PIC X(34)       VALUE SPACES.
     05  FILLER                           PIC X(17)       VALUE "RED RIVER COMPANY".
     05  FILLER                           PIC X(37)       VALUE SPACES.
     05  FILLER                           PIC X(5)        VALUE "PAGE ".
     05  HL-1-PAGE-NBR                    PIC ZZ9         VALUE SPACES.
     05  FILLER                           PIC X(12)       VALUE SPACES.
 01  HEADING-LINE-2.
     05  FILLER                           PIC X(55)       VALUE SPACES.
     05  FILLER                           PIC X(22)       VALUE "OFFICE PAYROLL SUMMARY".
     05  FILLER                           PIC X(55)       VALUE SPACES.
 01  HEADING-LINE-3.
     05  FILLER                           PIC X(57)       VALUE SPACES.
     05  FILLER                           PIC X(18)       VALUE "SEPTEMBER 30, 1986".
     05  FILLER                           PIC X(57)       VALUE SPACES.
```

```
01  HEADING-LINE-4.
    05  FILLER                          PIC X(10)        VALUE SPACES.
    05  FILLER                          PIC X(8)         VALUE "DIVISION".
    05  FILLER                          PIC X(13)        VALUE SPACES.
    05  FILLER                          PIC X(8)         VALUE "E/NUMBER".
    05  FILLER                          PIC X(20)        VALUE SPACES.
    05  FILLER                          PIC X(6)         VALUE"E/NAME".
    05  FILLER                          PIC X(20)        VALUE SPACES.
    05  FILLER                          PIC X(12)        VALUE "GROSS INCOME".
    05  FILLER                          PIC X(13)        VALUE SPACES.
    05  FILLER                          PIC X(10)        VALUE "NET INCOME".
    05  FILLER                          PIC X(12)        VALUE SPACES.
01  DETAIL-LINE.
    05  FILLER                          PIC X(12)        VALUE SPACES.
    05  DL-DIVISION-NBR                 PIC 999          VALUE SPACES.
    05  FILLER                          PIC X(18)        VALUE SPACES.
    05  DL-EMP-NBR                      PIC 999          VALUE SPACES.
    05  FILLER                          PIC X(16)        VALUE SPACES.
    05  DL-EMP-NAME                     PIC X(20)        VALUE SPACES.
    05  FILLER                          PIC X(16)        VALUE SPACES.
    05  DL-GROSS-INCOME                 PIC Z,ZZZ.99.
    05  FILLER                          PIC X(16)        VALUE SPACES.
    05  DL-NET-INCOME                   PIC Z,ZZZ.99.
    05  FILLER                          PIC X(12)        VALUE SPACES.
01  TOTAL-DIVISION-LINE.
    05  FILLER                          PIC X(10)        VALUE SPACES.
    05  FILLER                          PIC X(15)        VALUE "TOTAL DIVISION ".
    05  TDL-DIVISION-NBR                PIC 999.
    05  FILLER                          PIC X(59)        VALUE SPACES.
    05  TDL-GROSS-INCOME                PIC ZZ,ZZZ.99.
    05  FILLER                          PIC X(15)        VALUE SPACES.
    05  TDL-NET-INCOME                  PIC ZZ,ZZZ.99.
    05  FILLER                          PIC X(12)        VALUE SPACES.
01  FINAL-TOTAL-LINE.
    05  FILLER                          PIC X(65)        VALUE SPACES.
    05  FILLER                          PIC X(11)        VALUE "FINAL TOTALS".
    05  FILLER                          PIC X(9)         VALUE SPACES.
    05  FTL-GROSS-INCOME                PIC ZZZ,ZZZ.99.
    05  FILLER                          PIC X(14)        VALUE SPACES.
    05  FTL-NET-INCOME                  PIC ZZZ,ZZZ.99.
    05  FILLER                          PIC X(12)        VALUE SPACES.
01  WS-DATE.
    05  WS-YEAR                         PIC XX.
    05  WS-MONTH                        PIC XX.
    05  WS-DAY                          PIC XX.
01  PROGRAM-INDICATORS.
    05  EOF-IND                         PIC X            VALUE "N".
01  PRINT-CONTROLS.
    05  LINE-COUNT                      PIC 99           VALUE 60.
    05  PAGE-NUMBER                     PIC 999          VALUE ZERO.
    05  PROPER-SPACING                  PIC 9.
01  TOTAL-ACCUMULATORS.
    05  EMP-GROSS-INCOME-ACCUM          PIC 9(4)V99      VALUE ZERO.
    05  EMP-WEEKLY-DEDUCTIONS-ACCUM     PIC 9(4)V99      VALUE ZERO.
    05  DIV-GROSS-INCOME-ACCUM          PIC 9(5)V99      VALUE ZERO.
    05  DIV-WEEKLY-DEDUCTIONS-ACCUM     PIC 9(5)V99      VALUE ZERO.
    05  TOTAL-GROSS-INCOME-ACCUM        PIC 9(6)V99      VALUE ZERO.
    05  TOTAL-WEEKLY-DEDUCTIONS-ACCUM   PIC 9(6)V99      VALUE ZERO.
01  WORK-AREAS.
    05  WA-NET-INCOME                   PIC 9(6)V99      VALUE ZERO.
01  COMPARE-AREAS.
    05  CA-DIVISION-NBR                 PIC 999          VALUE ZERO.
    05  CA-EMP-NBR                      PIC 999          VALUE ZERO.
01  SAVE-AREAS.
    05  SA-EMP-NAME                     PIC X(20)        VALUE SPACES.
```

```
*
***************************************************************************
*  THE A000-MAIN-ROUTINE ACCEPTS THE DATE, READS THE FIRST RECORD,
*  PERFORMS THE B100-PROCESS-PAY-RECORDS, THE D100-PRINT-DETAIL-LINE,
*  THE D200-PRINT-DIVISION-LINE, AND THE E100-TOTAL-LINE.
***************************************************************************
*
 PROCEDURE DIVISION.
 A000-MAIN-ROUTINE.
     OPEN INPUT   PERSONNEL-FILE-IN
          OUTPUT  PAY-FILE-OUT.
     ACCEPT WS-DATE FROM DATE.
     MOVE WS-MONTH TO HL-1-MONTH.
     MOVE WS-DAY TO HL-1-DAY.
     MOVE WS-YEAR TO HL-1-YEAR.
     READ PERSONNEL-FILE-IN
          AT END
             MOVE "Y" TO EOF-IND.
     MOVE PF-DIVISION-NBR TO CA-DIVISION-NBR.
     MOVE PF-EMP-NBR TO CA-EMP-NBR.
     MOVE PF-EMP-NAME TO SA-EMP-NAME.
     PERFORM B100-PROCESS-PAY-RECORDS
          UNTIL EOF-IND EQUAL "Y".
     PERFORM D100-PRINT-DETAIL-LINE.
     PERFORM D200-PRINT-DIVISION-LINE.
     PERFORM E100-PRINT-FINAL-TOTAL-LINE.
     CLOSE PERSONNEL-FILE-IN
           PAY-FILE-OUT.
     STOP RUN.
*
****************************************************
*  PERFORM FROM A000 TO PROCESS PAY RECORDS.
****************************************************
*
 B100-PROCESS-PAY-RECORDS.
     IF LINE-COUNT GREATER THAN 50
        PERFORM C100-PRINT-HEADINGS.
     IF PF-DIVISION-NBR NOT EQUAL TO CA-DIVISION-NBR
        PERFORM D100-PRINT-DETAIL-LINE
        PERFORM D200-PRINT-DIVISION-LINE
     ELSE
        IF PF-EMP-NBR NOT EQUAL TO CA-EMP-NBR
           PERFORM D100-PRINT-DETAIL-LINE.
     ADD PF-WEEKLY-GROSS-INCOME TO EMP-GROSS-INCOME-ACCUM.
     ADD PF-WEEKLY-DEDUCTIONS TO EMP-WEEKLY-DEDUCTIONS-ACCUM.
     READ PERSONNEL-FILE-IN
          AT END
             MOVE "Y" TO EOF-IND.
*
****************************************************
*  PERFORM FROM B100 TO PRINT HEADING LINES.
****************************************************
*
 C100-PRINT-HEADINGS.
     MOVE SPACES TO REPORT-LINE.
     ADD 1 TO PAGE-NUMBER.
     MOVE PAGE-NUMBER TO HL-1-PAGE-NBR.
     WRITE REPORT-LINE FROM HEADING-LINE-1 AFTER PAGE.
     WRITE REPORT-LINE FROM HEADING-LINE-2 AFTER 1.
     WRITE REPORT-LINE FROM HEADING-LINE-3 AFTER 1.
     WRITE REPORT-LINE FROM HEADING-LINE-4 AFTER 2.
     MOVE 6 TO LINE-COUNT.
     MOVE 2 TO PROPER-SPACING.
     MOVE PF-DIVISION-NBR TO DL-DIVISION-NBR.
```

```
*
*****************************************************
*  PERFORM FROM B100 TO PRINT DETAIL LINE.
*****************************************************
*
 D100-PRINT-DETAIL-LINE.
     MOVE SPACES TO REPORT-LINE.
     MOVE CA-EMP-NBR TO DL-EMP-NBR.
     MOVE SA-EMP-NAME TO DL-EMP-NAME.
     MOVE EMP-GROSS-INCOME-ACCUM TO DL-GROSS-INCOME.
     COMPUTE WA-NET-INCOME = EMP-GROSS-INCOME-ACCUM - EMP-WEEKLY-DEDUCTIONS-ACCUM.
     MOVE WA-NET-INCOME TO DL-NET-INCOME.
     ADD EMP-GROSS-INCOME-ACCUM TO DIV-GROSS-INCOME-ACCUM.
     ADD EMP-WEEKLY-DEDUCTIONS-ACCUM TO DIV-WEEKLY-DEDUCTIONS-ACCUM.
     WRITE REPORT-LINE FROM DETAIL-LINE AFTER PROPER-SPACING.
     MOVE 1 TO PROPER-SPACING.
     ADD 1 TO LINE-COUNT.
     MOVE ZEROS TO EMP-GROSS-INCOME-ACCUM
                   EMP-WEEKLY-DEDUCTIONS-ACCUM.
     MOVE PF-EMP-NBR TO CA-EMP-NBR.
     MOVE PF-EMP-NAME TO SA-EMP-NAME.
     MOVE SPACES TO DETAIL-LINE.
*
*****************************************************
*  PERFORM FROM B100 TO PRINT TOTAL DIVISION LINE.
*****************************************************
*
 D200-PRINT-DIVISION-LINE.
     MOVE SPACES TO REPORT-LINE.
     MOVE CA-DIVISION-NBR TO TDL-DIVISION-NBR.
     MOVE DIV-GROSS-INCOME-ACCUM TO TDL-GROSS-INCOME.
     COMPUTE WA-NET-INCOME = DIV-GROSS-INCOME-ACCUM - DIV-WEEKLY-DEDUCTIONS-ACCUM
     MOVE WA-NET-INCOME TO TDL-NET-INCOME.
     ADD DIV-GROSS-INCOME-ACCUM TO TOTAL-GROSS-INCOME-ACCUM.
     ADD DIV-WEEKLY-DEDUCTIONS-ACCUM TO TOTAL-WEEKLY-DEDUCTIONS-ACCUM.
     WRITE REPORT-LINE FROM TOTAL-DIVISION-LINE AFTER 3.
     MOVE ZEROS TO DIV-GROSS-INCOME-ACCUM
                   DIV-WEEKLY-DEDUCTIONS-ACCUM.
     MOVE PF-DIVISION-NBR TO CA-DIVISION-NBR
                             DL-DIVISION-NBR.
     ADD 3 TO LINE-COUNT.
     MOVE 2 TO PROPER-SPACING.
*
***************************************************
*  PERFORM FROM A000 TO PRINT FINAL TOTAL LINE.
***************************************************
*
 E100-PRINT-FINAL-TOTAL-LINE.
     MOVE SPACES TO REPORT-LINE.
     MOVE TOTAL-GROSS-INCOME-ACCUM TO FTL-GROSS-INCOME.
     COMPUTE WA-NET-INCOME = TOTAL-GROSS-INCOME-ACCUM - TOTAL-WEEKLY-DEDUCTIONS-ACCUM.
     MOVE WA-NET-INCOME TO FTL-NET-INCOME.
     WRITE REPORT-LINE FROM FINAL-TOTAL-LINE AFTER 3.
```

Figure 5: OUTPUT REPORT FOR TWO-LEVEL CONTROL BREAK, GROUP PRINTED OFFICE PAYROLL SUMMARY

```
DATE  09/15/86                          RED RIVER COMPANY                                  PAGE   1
                                      OFFICE PAYROLL SUMMARY
                                        SEPTEMBER 30, 1986

DIVISION            E/NUMBER               E/NAME                 GROSS INCOME          NET INCOME

  100                 123           CASE, TRELLES GLENN             2,750.00             2,445.00
                      145           ICARD, REBECCA MABEL            1,200.00               960.00
                      167           LACKEY, FRED SAMUEL             1,350.00             1,200.00
                      189           MANN, GLORIA MELISSA            1,550.00             1,300.00
                      199           NIKONGE, BESSIE ANNE              965.43               841.98

TOTAL DIVISION 100                                                  7,815.43             6,746.98

  200                 223           CHENG, EDDY HOWARD              2,000.00             1,840.00
                      235           FEW, AMANDA BARBARA               780.20               685.47
                      251           HILL, ROBERT PHILLIP              209.37               201.37
                      257           MULLEN, TOM PAUL                1,582.00             1,332.36
                      269           PONS, TIM FRANK                 1,160.00               939.40

TOTAL DIVISION 200                                                  5,731.57             4,998.60

  300                 358           COLLINS, KATHY WEBB               999.99               900.00

TOTAL DIVISION 300                                                    999.99               900.00

                                              FINAL TOTALS         14,546.99            12,645.58
```

From Education to Business and Back: A Look at Retraining to Suppliment the Computer Science Faculty Vacuum

Chris R. Brown
University of Evansville

Abstract

This paper reports the experiences of a former elementary school educator and successful businessman who retrained to teach at the college level within the field of computer science. It also describes some of the characteristics common to teaching, business and computer science and what makes these characteristics advantageous to a retrainee.

INTRODUCTION

There has been a lot of literature directed toward retraining to help fill the faculty vacuum now being experienced in many computer science departments. Most of the writings and the retraining programs are geared toward college level instructors who may be in departments currently suffering declining enrollments or in departments seemingly related to computer science, such as mathematics, business or the sciences. There may be an alternative to luring college level faculty from their chosen disciplines. Pre-college teachers and those who have left teaching because of the problems within the elementary and secondary educational environment may offer another pool from which to draw candidates for retraining. As one example, I would like to share with you my background and retraining experience. Perhaps, it may offer some insights about this otherwise untapped pool.

BACKGROUND

As a youth of the sixties, I inherited its humanistic ideals and liberal "save the world" philosophy, and, being of parents that teach at elementary and university levels, I regarded the teaching profession with the highest of esteem. I had a desire to teach from my first experience in front of a classroom in a course entitled Career Decisions. The enjoyable classroom experience and familiarity with the lifestyle of teachers lead to the pursuit of a B.A. in Elementary Education. Upon graduation, I discovered that there were no teaching positions available. This was a little unsettling; however, remaining firm in convictions, I began masters work. Some teaching was available by "subbing" in the local school system. Surely, a position would open. After all, there was a shortage of male elementary teachers. This situation may have been a first practical business lesson in supply-side economics. Not to mention, the first of several reasons for abandoning a career in education.

During the time spent on Masters work, I had contact with many other education graduates. Conversations with those that had acquired teaching positions furthered a growing disillusionment with the teaching profession. For the most part, they exhibited a high regard for teaching; however, the professional environment was not all it was cracked up to be. Associates relayed tales of the lack of parental and administrative support, virtually no self-direction, diminishing esteem for the "profession" and the economic hardships created by the poverty level starting salaries. But being single and still enveloped with a shaken, though not yet shattered, opinion of education, I maintained my resolve. The turning point, which caused a realistic evaluation of a career in education, was a decision for marriage and family development. On the positive side, there was my undiminished love of teaching, but the scales were tipped by the outrageous economic ramifications and the haunting adage of "Those who can, do. Those who can't, teach."

BUSINESS EXPERIENCE

After critical self-assessment and evaluation of the job market, I decided that some level of management seemed the appropriate avenue to follow. Having little previous experience other than part-time retail sales, military food service training and an inbred dedication to professionalism, I procurred a management trainee position with a local fast food concern. Believe it or not, the salary was nearly equivalent to the starting teacher salaries. Management skills are very similar to those a teacher develops through formal training and experience. Effective management envolves the ability to plan ahead, the ability to motivate personel and the ability to train and evaluate your subordinates. Obviously, there are other criteria; however, most business professionals would agree that these skills are of primary importance. Since I had these basic skills, I advanced to Assistant Manager in a relatively short time. After six months of learning the ropes I was promoted to Executive Store Manager.

The year and a half I spent as manager was an enlightening experience. During that time, my management skills and food service knowledge base were honed by trial and error, a number of trips to the library and educational seminars offered by various professional organizations. Being able to train, evaluate, motivate and plan were prerequisite job skills. Success as a store manager led to promotion to Supervisor. I spent a year in this position, now training management personel. My other job responsibilities included maintaining company established policies and controls, unit wide promotional implementation and managerial supervision. I was next promoted to Director of Operations. Up until now, training had been a major portion of my job responsibilities. But as Director of Operations, training had less emphasis. Most of my duties were administrative. Fortunately, my education background did not let me down, my coursework in statistics was of particular benefit coupled with my "from the mailroom up" background made my project feasability and profitability projections realistic and reasonably accurate. Although my success was imminent, I still missed life in the classroom (what little I had known).

About two years later, I "got creative" and bought the company. Since the purchase, the company has and is doing well. I decided that I had disspelled the adage "Those who can do..." Sounds easy doesn't it. Actually not all former or present educators would find a business career as a suitable alternative. Success is a matter of timing and the ability to shut off concern for the personal problems of employees. The effective manager must appear supportive of his subordinates, but foremost in his mind must be profitable performance of operations. When dollars become the measure of success, success becomes rather superficial; hence, I felt a need to gain substance.

WHY COMPUTER SCIENCE?

The computer is now an integral part of the business environment. Though initial contact with the computer was during my undergraduate days, actual envolvement was initiated when the company I had been working for made the transition from utilizing a computer service bureau for bookkeeping and accounting to a "in-house" system. My limited undergraduate experience, along with numerous conversations with a friend who owns a custom software shop familiarized me with the potential of the machine beyond merely clerical duties. Many operational functions were developed and implimented. It became obvious, that there was definitely a need to learn more about the computer. After acquiring the company, this need became more than just something to consider. I began learning about the computer by reading, playing with the machine and talking with anyone that might improve my knowledge base. The more I learned, the more I wanted to learn. I was hooked! I finally enrolled in a programming class at the University of Evansville. I had a little apprehension about the course because I heard that it was the course used to cull out those undergraduates merely testing the waters of computer science. If I did well, I decided I would pursue computing science as far as it would take me, provided it would not too radically effect the business. As it happened, I found the course, though time consuming, very enjoyable.

What factors contribute to a former teacher / businessman finding success in computing: Those factors that lead to a successful transition from education to business seem concurrently prerequisite to success in computing. Programming requires the ability to carefully analyze a problem, plan a method of resolving that problem, implementing that plan, evaluating the plan and revising the plan based upon evaluations. These are no more than the same tasks performed by a teacher or a businessman during their daily routine. A teacher must analyze student needs in relation to the material being taught, plan a method of meeting those needs, and implement the plan. A businessman facing a project must first analyze the project, devise a plan.... Another common factor relates to the ever-changing computing environment. Currency and accuracy of information ofter determine success within all three areas. A teacher must keep ahead of his students and a businessman will often beat out compeditors by being aware of new trends and procedures. Ingenuity is yet another common attribute. The accomplished teacher or businessman must be innovative. Though not as blatant as other elements, new approaches to problem solving often segregate the truly successful. Also success within all three areas requires a modicum of common sense. This factor is not easily quantified; however, common sense or logic is needed by the teacher, the businessman and the programmer. Professionalism could be described as one's ability to pay attention to details inherent to each field whether subtle cues as to a students lack of comprehension, in teaching, or selecting the right product to stimulate sales in business, or in computing, the ability to detect a syntax error or revise an algorithm. Though not directly related to programming, another factor common to the three areas is superior communication skills. Because a reasonable balance of these components are found within persons of each category, the transition from field to field is generally blessed with commensurate success. It is not to say that all will meet with success; however, the likelihood of accomplishment seems greater than persons from other fields of endeavor.

RETRAINING

I next made inquiries of the School of Graduate Studies at U.E. regarding degree programs. Upon learning of the Masters program in Computer Science Education, a program conceived to retrain college faculty to teaching computer science, I hardly believe it. Here was an opportunity to learn about computing and as a portion of the degree requirements, return to the classroom as an instructor at the college level. Teaching at the college level is an entirely different professional environment than that of primary and secondary levels of instruction. Many questions arose. Would teaching still be as satisfying as it was in the past? Could I handle the coursework in the program? Would I fit in with the other candidates? Could I finagle the retraining schedule? After much careful consideration and a temporary reorganization of my job responsibilities within the company, I enrolled. At the very least, I would gain a better understanding of the computer.

The Masters of Science in Computer Science Education degree program offered at the University of Evansville can be described in four basic components, previous experience being the first. The candidate's past envolvement with the computer, whether formal course work or verifiable proficiency via independant study, must afford him with a fundamental knowledge of programming. The more proficient his programming, the more likely his success with the next component, Summer I. Summer I is an intense study of seemingly advanced programming, utilizing Pascal and PL/I programming languages. The other areas studied are systems analysis, computer hardware, systems programming, and the Computer Science curriculum. Component three takes place over the academic year. It involves teaching at least two Computer Science courses and writing two papers of publishable quality on some aspect of computer science utilizing the experience and knowledge gained about the discipline. The final component is Summer II. During this second summer, the candidate will take two courses in comparative programming languages and additional computer science electives.

Reflecting after the first summer, I realize that much of the stress and anxiety induced by trying to learn so much, so fast has been mitigated by memories of the comeradery established with other candidates and a strong sense of accomplishment. The plethora of information disseminated in the ten week time period was, to say the least, overwelming. Retention seemed impossible. In fact, it is a little difficult to categorize what was learned. The computer related knowledge base is there. Thus far, it has been excellent foundation from which to grow. Many of the questions that arose prior to enrollment have been answered. Apparently, the coursework could be handled. Perhaps, not to standards ordinarily acceptable, but this was attributable to the condensed timespan. With few exceptions, the other candidates were in a similar state of mental disarray. The mix of backgrounds they brought included Mathematics, Sociology, English Literature, Education ranging from elementary to college level, and others. By far, the most common background was mathematical. While each background type had his or her specific strengths there appeared to be offsetting weaknesses. Virtually all candidates drew together to assure success by drawing off each others strengths and supporting each others weaknesses. Finagling my business schedule had been a little tricky but fortunately, the company personel had been well trained and a key person was due for promotion.

TEACHING AT THE COLLEGE LEVEL

The remaining question that came up prior to enrolling in the M.S.C.S.E. program was: "Would teaching still be as satisfying as it was in the past?" Now, having taught courses in programming and Business Computer Systems, the answer to the question is "yes". Teaching still holds its exhilarating allure. The experience was, in a word, fun.

The first course taught was Introduction to Computer Business Systems, the equivalent of DPMA's CIS-1. During this course, utilization of my business experience, teaching experience and formal training in computing synergisticly provided for an effective experience for the student. The course is very general in nature, and virtually all areas of computing are encountered and investigated to some degree. Researching for lectures offered an excellent opportunity to fortify the knowledge gained over the previous summer and clarify areas of confusion. The classroom experience was challenging. Anything could come up during discussion. After establishing rapport and earning the students' respect, these discussions were stimulating and rewarding for both student and teacher. By blending business experience with educational experience, a wide variety of classroom activities were possible, ranging from "hands on" computer exercises to guest lecturers from the local Data Processing business community. The students gained much from the variety of experiences. The business background afforded the students with many real life examples, as well as, contact with some resources unavailable to other faculty. The education background allowed for effective, comprehendable presentation of the material to be covered.

The other area which I taught was an introductory programming course for non-majors. It was less of a challenge than the CIS-1 course. Though discussion and questions were encouraged, the course was more or less one directional. The true challenge was to motivate the the students to develop a consistant, effective and logical approach to program development. Again, my business background provided practical applications and my education background proffered understandable presentations. By fully understanding the "why's and wherefore's" of program development, the students succeeded in generating programs that were well constructed and functional.

Educational and business experience were both of immeasurable import to these successful teaching experiences; however, education was by far more useful. A good foundation in educational methodology diminishes the problems that a person with only a business background encounters during the transition to academia. Additional research is generally required to suppliment lessons with practical applications for the educationally oriented teacher but the business retrainee needs to spend far more time learning how to effectively develop a lesson, how to create a functional evaluation instrument or, more important, how to meet the individual learning needs of the students.

CURRENT CONCERNS

Several questions come to the reader's mind: How can the retrainee simultaneously teach and further his formal study of computer science? How much formal study is necessary? Will the retrainee be accepted in the realms of higher education? Is there a market for someone with his background?

The college environment allows for both teaching and learning. Many colleges have positions where in-depth pursuit of computer science would be possible while teaching undergraduate courses within the discipline. As knowledge is increased, courses of more complexity can be added to the teaching load. It is important to have an available source of structured learning for the retrainee to formalize his educational background within the field.

Can anyone really ever learn enough about his chosen field of endeavor? Not really, especially within a field as rapidly changing as computer science. There are those who would learn only enough to satisfy the teaching of a specific course but this certainly requires more than having taken that course. To facilitate a holistic learning experience the instructor must be able to relate course material the student has acquired within other courses. My computer science education degree is just a beginning, and I am already anticipating a second degree within computer science.

Acceptance from colleagues, in any field, must be earned. Given a commitment to professionalism, an intense drive to learn and the desire to be accepted, credability will come in time, it is really dependent upon the environment. Finding a flexible, progressive environment is of utmost import for anyone seeking to gain acceptance within a particular field.

Where there is a need, there is a market. Retrainees from virtually any background should be able to find a position of some kind within the field of computer science education. As industry continues to cause attrition within C.S. faculties, there will be an increasing need to tap every available resource for qualified instructors. Much attention has been directed toward the retraining of college faculty into the field of computer science. Why limit ourselves to these educators, when other educational devotees are available?

CONCLUSIONS

With all of the concern regarding the staffing of computer science departments, there is a need to investigate numerous sources from which to fill this increasing void with stable, well trained professionals. Perhaps this paper will offer some insights into the education pool. Granted, many may be so disillusioned that they are not retrainable or they may not have the luxury of available time to be retrained for teaching at higher levels. Those that have left the field may not wish to reenter. Many may feel the computer is not their bailiwick. Perhaps, with the current state of elementary and secondary education, we should not lure away those dedicated professionals that remain; however, can not the same be said of industry luring college faculty?

INTRODUCING MICROCOMPUTERS TO MANAGERS

C. David Rolfe and Cynthia E. Johnson

When Computers Successfully Penetrate the Managerial Levels

Soon after the initial introduction of microcomputers, vendors, consultants and business men and women have argued the question of computer usage at all levels within the organization. One buzzword has been used time, and time again, with no clear cut definition -- the executive workstation. The intent here is more of an attitude, than a grouping of hardware and software at a manager's desk. The question has been pondered: will computers be used by managers? It is not, however, a question of will -- a difference of philosophy exists here. Eventually, computers will be in almost every manager's office; the question, therefore, is when.

Few of us were in the workforce when the use of an automobile for transportation replacing the horse and buggy was the exception rather than the rule. Additionally, a very limited number of managers admit to having no telephone near their desks. Eventually, the latest in industrial and technological advances find their way into every aspect of the corporate environment. We are already beginning to see the use of computers in business. It has served millions as a useful tool to complete repetitive tasks. Younger professionals (ages 25-40) continually find means of analyzing data and manipulating text or accessing information bases using a computer.

Data processing professionals must view the issue in terms of a time frame -- WHEN computers are introduced at the various levels of management. Armed with this philosophy, data processors must identify ways to turn that time frame to our advantage. All executives will not use the computer. This may be to our advantage. Too often, we tend to speak in absolutes. "Every one has to have a computer on his or her desk to survive in corporate America!" Nor is it necessarily important for those who eventually choose to have a computer to understand how the computer works, rather how they may use the computer to their advantage. Our challege, then, is to offer the appropriate amount of training and support WHEN the time is right.

Data processing professionals have the advantage of gaining an ally or have the risk of creating an enemy in a position of strength with the initial introduction of the microcomputer in the manager's domain. The introduction of the computer to a manager had better be correct the first time. There are no second chances with a first time user at that corporate level. Our goal is to turn these first time users into second time users.

Three points of importance should be considered when introducing the computer to upper management. (1) The knowledge of the manager, (2) The trainer's own experience, (3) The trainer's skills (not to be confused with training). The introductory meeting should aid the manager in overcoming his/her reluctance and anxiety without boring them, damaging their dignity, or making them defensive. The upper level manager wants to know how the computer fits inot the big picture. He or she is not concerned with clerical details; rather in the relationship between computing and productivity.

Higher level managers may tend to be more conservative in attempting to implement a new concept such as computers in their offices. They will at some time look down the corporate ladder and see junior executives and subordinates climbing, well equipped in knowledge and skills from the technological revolution. Remember, the Silicon Valley cliche: "Never underestimate the power of outmoded habits to survive."

UNDERSTANDING THE MANAGERS' NEEDS

In order to understand a manager's needs, it is important to divide the category into upper-level managers and lower-level managers. For our purposes, an upper-level manager is one whom develops and implements strategies, policies, and procedures. These are the decision-makers in the corporation and job functions are, therefore, highly unstructured. Lower-level managers, on the other hand, are concerned with day-to-day operations of the corporation as well as direct supervision of employees. The duties of a lower-level or line manager are very structured.

A recent study completed by the authors found that 33 percent of upper level managers use personal computers at work while 17 percent of the same managers are using personal computers at home. This is an indication of the direction being taken by many corporate executives.

There are three basic personalities of managers; entrepreneurial, directive, and participative. Entrepreneurial managers are informed, tough, risk-takers. These managers actively search for new opportunities because they are innovative, creative, aggressive, and impulsive. Entrepreneurial managers are achievers because they set challenging goals, seek to improve their performance, and they value results. Therefore, they are constantly exploring avenues which enable them to meet their new challenges head-on. State-of-the-art technology is an aid to an entrepreneurial type.

Directive managers are nearly opposite of entrepreneurial. The directive manager is authoritarian requiring absolute power. It is appropriate to use his/her status and power in deliberate decision-making policies. Directive managers are risk adverse making the use of technologies sensible only after they have stood the test of time and been proven by others. Job centered, task-oriented managers are extremely formal in their work attitude.

The managers who view life and work as a social responsibility are known as participative. This managerial personality is trusting, reliable, and always accountable, although he/she is assertive by his/her own right. This type of manager can be persuaded, given the appropriate degree of explanation and trust.

What does all this mean to data processors? With a thorough understanding of the manager's personality type, the trainer may prepare to "face the enemy with both barrels loaded." Knowledge of the personality type one is about to encounter will obviously make the encounter a positive experience. Remember, the initial contact will determine the volume of use of the computer at any management level following that encounter.

In addition to the personality of the individual manager, one must be aware of the underlying thoughts of managers in general. Managers want to have control no matter what their personality type. Some view the use of the computer as a limiting factor in retaining control. The organizational hierarchy appears to become flattened because many people have easier access to information.

Managers prefer the spoken word. Without a doubt an idea, thought, or directive is better expressed face to face. The words may be the same but body language plays a crucial role in communicating, no matter how subtle. Facial expressions, voice fluctuations, and stance all tell the listener as much as or more than the actual words which are being shared. Memoranda and electronic mail contain words which dispel a message but are much more open to interpretation without the influence of the nonverbal aspects of communication.

Managers are busy. They do not have the time, nor the inclination, to sit at a terminal for hours upon hours learning a particular system whether software or hardware. Nor do they wish to take time to sit in on training lectures. There are telephone calls to make, people to see, meetings, budgets, and so on.

Then there is the minor detail of the manager's image. Put a keyboard in front of a manager and immediately he/she thinks of a clerical rather than a managerial orientation. The manager begins to worry that he/she lacks the difficulty to understand information such as keystroke sequences, computer disciplines, and applications knowledges necessary to operate the contraption. Using the computer may diminish the manager's value to the company. After all, a manager primarily makes decisions. Why have a manager if a subordinate could do the job as well or possibly even better? There is also the matter of performance anxiety. The manager has been in charge for so long and now has to worry about doing seemingly meaningless tasks.

Finally, a manager addresses business perspectives and issues which usually do not include word processing or spreadsheet analysis. A successful data processing trainer would be wise to engage in activities to protect the manager's image of himself or herself.

TRAINING THE MANAGER IN THE USE OF MICROCOMPUTERS

Earlier we mentioned that knowledge, experience, and skills are needed to introduce the computer into the manager's office. What are these knowledges, experiences, and skills needed to successfully introduce the computer into the manager's office? They entail a combination of knowledge of computer technology teaching skill, and specific business experience. For the computer to make a successful impact in the manager's office, there are three things that must take place:

(1) The system must be of the utmost simplicity with a high degree of ease of use.

(2) The benefits of use must be obvious.

(3) The executive must believe that the information made easily available by the computer in turn makes the executive more efficient and thus more effective.

The system introduced into the manager's office will not be one that allows the user to enter and manipulate large amounts of data. Instead it will be one that allows that user access to existing issues.

Two areas create a gate through which the successful trainer must pass: indifference and anxiety. Managerial indifference must become enthusiasm; managerial anxiety must be transformed into confidence. The vehicle for this transformation to take place has been discussed herein. We need to know the executive and we must understand the executive's needs.

In addition, there has to be some determination of the level of usage to which the executive can be taken. If you are not certain as to the extent of detail to advance, then stick to the basics. Simplify everything. Managers are looking for intensely user-friendly machines -- machines which do not require thinking. The acronym for the trite expression KISS is always an aid to the data processing trainer in preparing to introduce computers to top managers. Our version of the KISS approach will help you gain an advocate in the upper corporate levels. Keep it short; simple!

Once those determinations have taken place, develop an extensive plan which also allows a high degree of flexibility. Never sacrifice specifics for flexibility. Assume nothing, and be aware of how the training is going. Stop before you meet resistance. Executives want to learn how to solve the problems they have. The last thing they want is to have new problems created for them by a machine.

Time is of the essence, too. There is an obvious need to condense the learning process to the shortest possible time span. Learning a little at a time is much better than spending a lot of time now and not being able to look at it again for several weeks.

Knowledgeable of the general rules of training, the data processing professional may now select one of three basic approaches to the training cycle:

(1) Computer Based Training

(2) Hiring an outside firm

(3) In-house training

The primary aim of a computer based training package is to get the user up and running on a specific package or machine. A well-designed CBT package avoids intimidating the learner, yet provides enough practical knowledge to do meaningful tasks upon completion of the program. Four benefits can be derived from computer based training. Training costs are reduced even though the initial price of the tutorial may seem expensive, the package can be reused time and time again. The package allows you to reach a larger user population without additional costs. It also opens scheduling possibilities without additional costs. Finally, computer based training would help reduce travel costs for companies with remote offices. On the other hand, CBT packages often fail to allow the user to make mistakes, which is actually a valuable learning experience in itself.

Trainers independent of the corporation can offer some benefits. Generally speaking, the outside trainer automatically has the respect of the trainees because of the lack of familiarity of the trainer. The quality of work is usually higher, too, because the trainer devotes full time to that particular function. However, the cost of a customized course for an organization can run as high as $250,000.

In-house training offers the greatest versatility and returns the greatest variance in results. The in-house trainer generally knows the company politics and how to deal with them. At the same time, however, the in-house trainer will have to work with the students once they finish their training. For that reason the in-house trainer should be cognizant of a few points.

(1) Remember a trainer is someone who teaches the manager the right keystrokes, whereas an educator shows the user how to focus on solving business problems through the right selection and proper use of a system.

(2) The trainer needs to interact with the user, to personalize the experience and make it fun, and to anticipate the user's needs. For example, do not assume that the manager can type.

(3) Be prepared to let the manager ask questions both simple and difficult. Show them a manual and say "Do page 1, exercise 1" and you will probably get thrown out of the office.

(4) It is generally best to train the manager one-on-one in his or her office. This avoids some office politics.

(5) Do not overload the manager with what the computer can do; start with something simple. Remembering that executives need qualitative information that has strategic value and will get them to interact with others.

(6) Finally, use the easy-to-use features available on computers such as Icons, mouse, and windows. Build the system to work the way the manager works.

Good systems will be "user seductive." They will minimize the time required to do routine tasks allowing the executive more time for involvement in the decision making process so necessary to being an effective manager. Data processing trainers can introduce good systems, hence, computers, to managers with the knowledges of the managerial personality style, the needs of the managers, and the appropriate training method.

BIBLIOGRAPHY

Harrison, William D. "Working the Human Side," COMPUTERWORLD, May 21, 1984, ID15-ID24.

Hollis, Robert, "Real Executives Don't Use Computers," BUSINESS COMPUTER SYSTEMS, July, 1984, pp. 41 - 42.

Kiechell, III, Walter, "Why Executives Don't Compute," FORTUNE, November 14, 1983. pp. 241-246.

Linder, Jane, "Harnessing Corporate Culture," COMPUTERWORLD, September 23, 1985. ID1-ID10.

Nulty, Peter. "How Personal Computers Change Managers' Lives," FORTUNE, September 3, 1984, pp. 28-40, 44-48.

Verrando, Robert E. "Let Computers Enhance Executives Environment," INDUSTRY WEEK, March 18, 1985, p. 14.

MIS with a Public Flavor: A Graduate Program in Public Information Systems

Frank W. Connolly, Ph.D. and Thomas J. Bergin, Ph.D.
The American University
Washington, D.C.

This paper presents an overview of an MIS curriculum designed for Public Administration students. It is an adaptation of the ACM's Information Systems model (4), and reflects the major concerns of the National Association of Schools of Public Affairs and Administration (NASPAA).

INTRODUCTION

For government agencies the computer is more than a decoration or symbol of sophistication. As with most private sector organizations, government agencies at federal, state and local levels cannot function without computers. Agencies depend on computers to process their administrative work, to massage the volumes of data that provide the grist for their work, and to facilitate the delivery of services.

Computers have been used in government since the 1940's, although computers have not been a curricular requirement in public administration. The National Association of Schools of Public Affairs and Administration (NASPAA) published a report (1, 2) proposing four critical factors that ought to be reflected in educating public officials. It proposes that Public Administration education should:

1. assure that all public management students have an appreciation for computers and computing;

2. recognize that public organizations require different levels of computer awareness, skills, and knowledge;

3. encourage public management programs to develop computer curricula; and,

4. encourage the integration of computing topics and tools into public administration course offerings.

Implied in the recommendations is the belief that public administration is sufficiently different from business administration to justify separate teaching units. Extending this assumption is the belief that MIS in public administration is sufficiently influenced by the public environment and focus that it is best taught in school whose primary focus and faculty expertise is public sector. The basis for these assumptions is the web of public laws, public interests, executive orders, and organizational structure that place unique demands and limits upon practitioners (e.g. the dichotomy of adhering to both sunshine legislation and privacy requirements, the question of file matching based upon government records and files). The differences between public administration and business administration not only justify different teaching units, but require different approaches to the analysis and design of computer systems.

Similar to schools of business administration, there is an assumption that schools of public administration can meet the critical need for educating current public administration faculty regarding computers and computing.

LEVELS OF KNOWLEDGE

Kraemer and Northrup (3) define four levels of computer knowledge. This schema provides a framework for examining the applicability of a public sector curriculum in MIS. The four levels of knowledge are:

1. Computer Consciousness - which involves an introduction to computers and computing for the uninitiated.

2. Management Use of Computing - which encompasses the skills and knowledge required for managers to apply computing capabilities productively within existing organizational environments.

3. Management of Computing - which involves knowledge of those policies, techniques and procedures employed by managers to provide and control computing capabilities and services.

4. Management Information Systems - which involves understanding the tools and techniques required by information system specialists for designing and implementing information systems.

The American University (AU) is one of the top ranked colleges of public administration in the country. By instituting a graduate program in public

INITIAL REQUIRED SEQUENCE

55.601 Introduction to Public Information Systems
55.603 Applications Development
55.605 Systems and Information in Public Organizations
55.607 Quantitative Analysis for Information Systems
55.609 Human Relations and Information Systems

APPLICATION OPTION (12 hours)	MANAGEMENT OPTION (12 hours)
55.631 Information Systems Analysis	55.641 Information Resource Management
55.633 Information Systems Development	55.643 Managing the Systems Development Life Cycle
55.635 Microcomputer Applications	55.645 Project Management
55.637 Data Base Management Systems	55.647 End User Computing

ELECTIVE COURSES
(6 hours)

55.671 Telecommunications and Information Systems
55.673 Information Systems Evaluation and Acquisition
55.677 Advanced Quantitative Management
55.679 Selected Topics in Information Systems & Technology

CAPSTONE REQUIREMENTS
(9 hours)

55.700 Public Information Systems Policy
55.701 Workshop in Public Information Systems
55.703 Seminar in Information Systems Policy and Implementation

Figure 1. Overview of TPIS Curriculum

sector management information systems, it hopes to enhance its reputation by addressing all four levels of the Kraemer and Northrup model.

The new masters level program in Technology and Public Information Systems (TPIS) offers a Master of Science degree for individuals interested in the development and management of computers and information technology in the public sector. The program offers two concentrations to address the varied student population that AU attracts. One concentration offers in-depth courses in computer and allied technologies for those with little prior technical experience. The second offers courses in the management of those technologies for students with significant prior experience and appropriate education.

The products of the program will be analysts, specialists, and managers, able to participate in the design, development, and implementation of information systems in public organizations. Moreover, the graduates will be able to contribute to the policy processes surrounding the use and management of computers and information systems in the public sector.

THE PROGRAM

The program is broken into four segments: an initial sequence of required foundation courses, a concentration of technical or management courses, electives, and a three-course capstone sequence. The complete program is forty-two (42) semester hours. Figure 1 outlines the curriculum showing the relationships and sequencing of courses.

PRE-PROGRAM REQUIREMENTS

One of the assumptions of this program is that students entering the program have had some exposure to computers, either as students or as employees. Indeed, it is assumed that students in the future will have had greater exposure and experience than present students. For this reason, non-credit skill modules/workshops are available for students lacking skills in such areas as word processing, spreadsheets, and statistical packages, as well as in technical writing and graduate research skills.

AU's graduate students at present, come from a wide variety of backgrounds. Many students are federal or local government employees who find the computer entering their professional domain (such as engineering or accounting) and need to learn more about it. Some students are professionals wanting to change disciplines, (i,e, they wish to become analysts and specialists working with public sector information systems). Another segment are students who come directly from undergraduate studies, most of whom did not study information systems in their undergraduate programs. The final part of our student population are from Third World nations, studying computer and information technology so that they can build information systems in their own countries. The purpose of skill modules is to reduce the gap between students of different backgrounds, allowing all students to fully participate in common coursework.

INITIAL REQUIRED SEQUENCE

The initial sequence of courses is:

55.601 Introduction to Public Information Systems

55.603 Applications Development

55.605 Systems and Information Concepts in Organizations

55.607 Quantitative Analysis for Information Systems

55.609 Human Relations and Information Systems

These courses provide students with the foundation skills necessary to participate in the analysis, design and development of information systems in the public sector. The five courses offer a balance of skills -- technical, organizational, and analytic -- essential for success in MIS and their academic program.

55.601 Introduction to Public Information Systems examines the concepts underlying the design, development and use of public information systems. It includes hardware and software concepts, the systems development life cycle, program development, and an introduction to information systems in public sector organizations.

55.603 Applications Development is perhaps the most unique course in the program, because it requires students to design and develop programs in a high level language using structured programming tools and techniques. Its purpose is to immerse students in the program development process to develop an awareness of program development techniques and requirements. Students design, code, test and document programs using mainframe computer facilities to gain some hands-on insight into the major mystery of the computer, i.e. the programs that drive it. The objective is not to develop programmers (that is one of the purposes of our undergraduate programing). Rather, a thorough grounding in a records-based language, such as COBOL, builds the foundation to the informed use of software tools such as data base management systems. The study and use of a language also provides students with some insights into the systems development life cycle which cannot be acquired in any other way. Finally, students gain some confidence when they understand the programming process -- confidence which will be most helpful when they are placed in management positions.

55.605 Systems and Informations Concepts in Organizations presents and illustrates the major concepts and techniques that comprise the systems perspective with a special focus on public information systems. To many individuals, a benefits delivery system is composed of a computer and little else. This course places the computer (as machine) in a systems context within a public environment. Students learn the theory behind management and systems.

55.607 Quantitative Analysis for Information Systems examines a broad range of quantitative tools and techniques used in the creation of information systems. Students utilize statistical software in both mainframe and microcomputer environments.

55.609 Human Relations and Information Systems introduces students to the human relations aspects of public sector information systems. It focuses on the managerial, social and political dynamics that determine behavior in public organizations, and how these factors can be applied to the design and operation of public information systems.

CONCENTRATIONS

After completion of the required courses, students choose either the Application option or the Management option. The Application Option provides students with the skills and knowledges necessary to plan and create information systems. The Management Option focuses on the skills and techniques to effectively oversee the information resources, policies, and personnel of public agencies.

Application Option

The Application Option is intended for students with little prior education or experience in using computers. Courses in this option require students to actively participate in systems development activities as theoretical discussions alone are insufficient for students planning to participate in these processes as information professionals. The four courses in the Application Concentration are:

55.631 Information Systems Analysis

55.633 Information Systems Development

55.635 Microcomputer Applications

55.637 Data Base Management Systems

55.631 Information Systems Analysis examines the methods and processes of systems analysis with reference to public sector information systems. Emphasis is on analytic tasks relating to systems development, and the organizational and technological context within which these analytic tasks are addressed. Topics include structured analysis techniques and tools, feasibility studies, cost benefit analysis techniques, and functional specification preparation.

55.633 Information Systems Development provides an in-depth examination of the methods and processes of system design and implementation. Technological, managerial and organizational factors that comprise the public sector information systems environment are emphasized. Topics include structured design techniques, prototyping approaches, quality assurance techniques, applications software development, and appropriate levels and types of documentation and training necessary to the successful implementation of public information systems.

55.635 **Microcomputer Applications** focuses on effectively using and evaluating microcomputer software. The Applications Development course has already introduced students to the world of mainframes, and this course provides students with the other half of the hardware experience they need.

55.637 Data Base **Management Systems** examines the analysis, evaluation, design, implementation, and use of mainframe-level data base management systems. Topics include data base architectures, host language interfaces, query languages, data administration concepts, and data base administration techniques.

Management Option

The Management Option is reserved for students with strong technical skills, as well as experience in developing information systems. Courses in this option provide insights into the administration of technical organizations, the technical aspects of non-technical organizations, and the relationships between the various levels of administration in large public organizations. The four courses in the Management Option are:

55.641 Information Resource Management

55.643 Managing the Systems Development Life Cycle

55.645 Project Management

55.647 Managing End-User Computing

55.641 Information Resource Management focuses on information as a resource in public organizations. Strategic issues in planning, designing, and implementing (or acquiring) information systems and technologies are identified. Particular attention is paid to organizational structures and managerial alternatives. Topics include strategic management of information, federal policies impacting information resources management, and the alternatives for structuring information resources.

55.643 **Managing the Systems Development Life Cycle** examines the systems development life cycle (SDLC) and provides a framework for managing the systems development process. Attention is paid to political, economic, and organizational factors. Topics include managing the different phases of the systems development live cycle, the integration of management and decision theory, and the problems of power, leadership, conflict and control in technical organizations.

55.645 **Project Management** examines the tools and techniques for managing public sector information systems projects. It focuses on both the qualitative and quantitative aspects of managing projects throughout the systems development life cycle. Topics include estimating and benchmarking techniques, workload analysis, planning and scheduling large scale development projects, task tracking (using automated methods) and project evaluation and control.

55.647 Managing End-User Computing examines the use of computers by non-computer personnel. It focuses on managing the introduction, growth and institutionalization of end-user computing tools such as microcomputers, query languages, LAN's, and fourth generation languages. Topics include microcomputer-based tools and techniques, mainframe-based tools and techniques, strategic planning considerations of end-user computing, standards development, management and implementation, and the management of information centers

CAPSTONE COURSES

The capstone courses serve a number of purposes: first they provide students with a thorough grounding in the federal policies affecting information system development and management in the federal sector; second, students from the application and management concentrations have an opportunity to work together on a real-world project; and, third, students have an opportunity to discuss both technical and non-technical issues relating to public information systems. The courses in the capstone sequence are:

55.700 Public Information Systems Policy

55.701 Workshop in Public Information Systems

55.703 Seminar in Public Information Systems

55.700 **Public Information Systems Policy** examines public sector information systems policy, focusing on the structures, laws and regulations that affect information systems and technology in the public sector. The course analyzes the roles of government agencies in determining and carrying out information systems-related policy. Topics include the role of government in the development and regulation of the information industry, laws and regulations affecting public information systems procurement, design and use, and the role of central management agencies and the Congress (OMB, GSA, NBS and GAO).

55.701 **Workshop in Public Information** Systems focuses on the management of information systems for the public sector. Students participate in the analysis, development and implementation of an information system for a public agency, a not-for-profit organization or an educational institution.

55.703 Seminar in Public Information Systems is a research oriented seminar requiring a major research paper relevant to current issues in public sector information systems. Class discussions focus on current and emerging issues affecting technology in public organizations, professional ethics, including the ethics of information systems development and use, the evolution of information technologies, and the future of information technology. Because of American University's location, frequent guest lecturers from federal agencies are involved.

ELECTIVE COURSEWORK

The following courses are offered as advanced elective courses. Students may choose from these electives, or from courses outside their chosen concentration, to fulfill degree requirements. The content of these is obvious from the course titles:

55.671 Telecommunication and Information Systems

55.673 Information Systems Evaluation and Acquisition

55.677 Advanced Quantitative Management

55.679 Selected Topics in Information Systems and Technology

The **55.677 Advanced Quantitative Management** course examines advanced quantitative techniques for managing public sector information systems. Management science techniques including simulation and modeling are major aspects of the course, as well as discussions of a number of formal models that can be used in analytic decision-making.

55.679 Selected Topics is an open-ended offering that allows faculty to present topics of interest to both students and the faculty member. For example, course might include Fourth Generation Languages, Security and Privacy, or Technology Transfer.

SUMMARY

There is little debate as to the future of MIS. The foregoing describes a unique MIS degree program -- one with a public administration flavor. It is intended to serve a unique student population by reflecting the organizational context in which the MIS will function. The principles of MIS are impacted by this environment, and therefore the education of those who will manage it in public organizations ought to reflect this uniqueness.

REFERENCES

1. Kraemer, K.L. (Chair) (1986) "Curriculum Recommendations for Public Management Education in Computing," Social Science Microcomputer Review, 4 (1), pp 1-37.

2. Bergin, T. J. (1985) "Curriculum Recommendations for Computers in Public Management Education," Proceedings of the Eighth National Conference on Teaching Public Administration, St. Louis, MO., April 24-26, pp. 41-50.

3. Kraemer, K.L., and Northrup, A. (1984). "Computers in Public Management Education: A Curricula Proposal for the Next Ten Years," Public Administration Quarterly, Vol. 8, No. 3, Fall 1984.

4. Nunamaker, J.F., Couger, J.D., and Davis, G.B. (1982). "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs," Communications of the ACM, Vol. 25, No. 11, November, 1982.

END-USER TRAINING: THE WEAK LINK IN THE MICROCOMPUTER MARKET

Susan Helms, Hardin-Simmons University, Abilene, Texas

ABSTRACT

This paper summarizes some practical issues in end-user training in the microcomputer environment, including: (1) the need for training to achieve a growth market, (2) how to select and prepare trainers, (3) what components need to be included in the training, (4) choices of instructional media, (5) cost considerations, and (6) training evaluation.

WHY END-USER TRAINING IS IMPORTANT

Computer sales growth has traditionally come from new markets rather than from replacement, upgrading, or add-ons. One reason for this is that new developments in the computer sector tend not to replace older equipment, but to create new demands and new niches. The minicomputers did not replace the mainframes, and the microcomputers generally do not replace mini or mainframe operations.

With the numbers of microcomputers currently in place, the growth market in this arena must lie significantly among current non-users of computers. Figure 1 gives a rough model for the levels of computer expertise of potential purchasers of microcomputers (and/or microcomputer software and peripherals). While it is not easy to tie exact numbers or proportions to the figure, it should be obvious that many more potential micro users are to be found in Level I than in Levels II or III at this time.

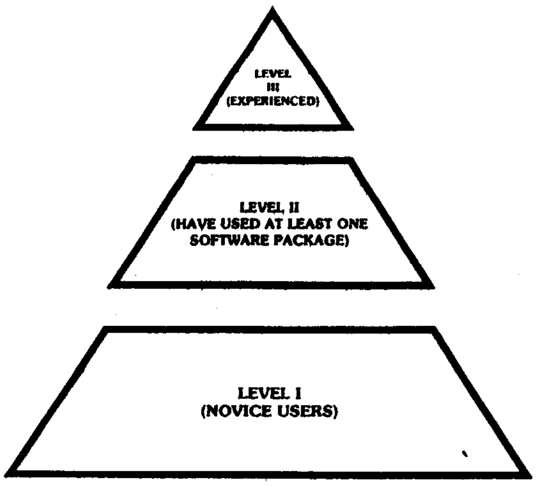


Figure 1 The End-User Population

Two important conclusions regarding end-user training can be made from the representation in Figure 1: First, since most of the knowledgeable computer users already have micros, the computer-resistant, novice user market is more wide open, and training can provide a key to overcoming many knowledge-related buyer barriers in this market segment. Second, training today needs to be pitched at the appropriate user level -- not expert training or novice training for all users.

Even when microcomputers are available, only 25% of large corporations are providing employee training [1], and obviously even this is not possible for the smaller companies. In this context, end-user training can provide a competitive edge for the vendor by "promoting product awareness, increasing customer satisfaction, and lowering technical service costs" [2].

DESIGNING END-USER TRAINING PROGRAMS

Where to start the training is one of the first decisions to be made when developing a training program. A glance back at Figure 1 suggests that the nature of the training audience will be critical in this decision. For true novices it will do little good to jump right into the intricacies and options of the software they must learn to use. Somewhere, somehow they must first learn a little about the hardware, how to use a floppy, how to save their work, perhaps even a little about the operating system, some keyboard training, etc.

What to include in the training also depends a great deal on the level of expertise of the individuals being trained [3]. Level I trainees will have to start with loading the program from the disk, how to use any menus, and work through the options, data entry, etc. Level II users might need the same sort of start as the Level I users, but need less repetition, less "hand holding", while Level III users typically want a much more summarized approach along the lines of "how do I get into it", and "what are the peculiarities I should know about". In "live" training these differences can be handled by separate sessions, and with training manuals they are often handled at two levels by a "quick start" section for knowledgeable users, and a step-by-step tutorial for novices.

Emphasizing specific applications utilized by the group being trained is critical to the transfer of general training to the specific needs of the user. All training should use real examples (actually enter real data, carry out the various processing steps, make the decisions about how the program will proceed, etc.). In the case of "live" training, the examples should be taken from the users' own applications if possible [4].

Show error messages, and how to recover in addition to the normal operation/options. For all levels of users it is important to know what to do (or NOT do) when errors occur. Particularly for Level I users it is important to show typical mistakes, what messages result, what must be done, and what should not be done.

Use a logical, top-down approach to designing the training program. Just as modular top-down design leads to better software, it also is a good approach to use in building the training program [5]. The training (whether live, video, computer based, or in manuals) should work its way logically through the options available to the users. It is important to keep early portions of the training as simple as possible for novice users - show the simplist or most common applications and build skill at that level before proceding to the more complex or esoteric options. When possible, emphasize applications the particular user(s) will be utilizing, and leave out materials they will not be encountering for a while.

Training materials should be accessible at any point, so that the more experienced users do not have to plod through from the beginning.

WHO SHOULD DO THE TRAINING

There are some serious problems with the common practice of having the sales force responsible for the training. Often the sales people are, naturally, more concerned with their next sale than in doing a thorough job of training the last purchaser. The personality traits and skills that make the best sales people are not necessarily the same traits and skills needed by an effective trainer.

Skills and traits needed by a good trainer include good written and oral communications skills (materials must be both developed and presented in most cases), technical competence (both in the subject matter and in using the software/hardware involved), enthusiasm, and patience [6, 7,8,9].

To find (or develop) good trainers identify the exact skills your training program requires (oral or written communication, accounting or engineering or whatever kind of general background, and the general level of computer skills required for the particular training task) and evaluate candidates on these specifics (have them write a simple "how to", give a short oral presentation, etc.) [6]. If not dealing with experienced trainers, some training for the trainers will be essential -- be sure to include how to deal with adult learners, understanding the background and perspective of the users, how to design and organize materials, and some practice and evaluation in developing materials, and in making presentations [8].

Trainer evaluation will be essential to a successful training enterprise [9]. Critical areas to look for include pace of the training, clarity of presentation, and whether or not the individuals being trained felt comfortable with the trainer.

TRAINING MEDIA

Instructor-led training with lots of hands-on work has a big advantage in overcoming computer resistance with novice users [10]. Additionally, live training can most easily be adapted to a variety of user experience levels. In many instances, however, logistics and cost considerations preclude exclusive use of live training, and all training media have some specific advantages that make them the preferred choice in some cases. For many software products and turn-key systems the best approach may involve the careful selection of a mixture of media to cover general background, specific applications, and updates, all at a variety of user levels.

Video training materials can be especially effective in showing the user exact sequences of operation/data entry and the results, all of which can be viewed in a self-paced or group mode -- and repeated as often as necessary. Since it will be used repeatedly, a great deal of effort and preparation can go into a video presentation. Caution must be employed to utilize the video medium effectively, while not trying to make the result into an "entertainment" product [11].

The computer itself is the best training medium in many applications, especially when the newly-available co-resident courseware authoring systems are utilized. While an especially effective tool for the Level II users, the computer based lessons may well prove initially frustrating to the complete novice, and not move fast enough for the Level III user [12]. Like video training materials, the multiple use prospects and self-pacing of CBI (computer based instruction) materials allow for more extensive preparation time and expense.

Even simple audio cassette tapes have been shown to be very effective in helping Level I users get through the initial phases of using simple, menu-driven software [13].

DO' AND DON'TS

Do train enough people (moral support and back-up).

Do train the right people (actual users and their supervisors) [14].

Do use examples specific to the users' applications.

Don't assume "it's obvious" (it almost never is).

Don't "gee whiz" the users (looking smart isn't as important as making the user feel confident) [15].

Do keep it simple.

Do train in the "hands-on" mode (their hands!).

Do show what can go wrong and how to recover.

Don't try to cover too much in too little time.

Do make the training a positive experience for the users [16].

COSTS AND LOGISTICS

Though end-user training is essential in the micro environment, it is equally essential to be able to determine the costs of training and to recover those costs. Some of the training costs are obvious, while others may be more difficult to determine accurately [17,18]. Only an accurate picture of potential costs can permit a reasonable choice among training media, and clear cost information is necessary before approaching the difficult question of cost recovery. While good training programs may "pay for themselves" in reduced support costs, and greater customer satisfaction, the actual dollars spent need to be accounted for either as a "bundled" item in the cost of the product, or as a separate item for the consumer to pay for.

EVALUATION OF TRAINING

Evaluation of training program/materials is the only way to determine their effectiveness. Whether elaborate post-training questionnaires, or simple follow-up phone calls are used, the people being trained can provide valuable insight into the strengths or weaknesses of the program/materials [19,20]. Following the evaluation, the training materials/program should be subject to ongoing revision as the user mix changes, product updates are completed, etc.

REFERENCES

1. Training Today - Training, the Missing Link, Training, 22(9), pp. 10-13, 1985.
2. Zemke, Ron, Customer Education: The Silent Revolution, Training, 22(1), pp. 27-39, 1985
3. Training Today - Instructional Design: The Basics, Training, 22(8), pp. 73-76, 1985.
4. Bezdek, Jiri, How to Teach Technical Subjects to Nontechnical Learners, Training, 22(4), pp. 73-81, 1985.
5. Good, Tom W., Building Presentations: A Top Down Approach, Training, 21(6), pp 50-55, 1984.
6. Clark, Ruth C., and Kyker, Phyllis, How to Select Good Technical Instructors, Training, 22(12), pp. 54-62, 1985.
7. Hanley, Marybeth, Technical Knowhow vs Teaching Skills, Data Management, 24(5), pp 14-15, 1986.
8. Kobel, Terri, and Farson, Alice, Poof - You're a Trainer, Training and Development Journal, 39,(1), pp. 106-107, 1985.
9. Caldwell, Robert M., and Marcel, Marvin, Evaluating Trainers: In Search of the Perfect Method, Training, 22(1), pp. 52-59, 1985.
10. Callaghan, David R., Realistic Computer Training, Training and Development Journal, 39(7), pp 27-29, 1985.
11. Martin, Peter, Training by Video: How to Shoot Yourself in the Foot, Training, 22(4), pp. 39-41, 1985.
12. Pepper, Jon, Creating Corporate Training Courseware, PC Magazine, 5(1), pp. 197-199, 1986.
13. Zemke, Ron, Teacher on a Tape: Audiotutorials for Computer Literacy, Training, 21(4), pp. 73-79, 1984.
14. Spitzer, Dean, 20 Ways to Energize Your Training, Training, 22(6), pp. 37-40, 1985.
15. Pop Quiz: Do's and Don'ts of Presentations, Training, 22(6), pp. 16 & 83, 1985.
16. Zemke, Ron, and Gunkler, John, 28 Techniques for Transforming Training Into Performance, Training, 22(4), pp. 48-63, 1985.
17. Shanks, Mark D., Hands-On Training: How Much Equipment Do You Need?, Training, 22(8), pp. 55-59, 1985.
18. Weinstein, Laurence M., Collecting Training Cost Data, Training and Development Journal, 36(8), pp. 31-34.
19. Kirkpatrick, Donald L., Techniques for Evaluating Training Programs, Training and Development Journal, 33(6), pp. 78-82, 1979.
20. Komras, Henrietta, Evaluating Your Training Protrams, Training and Development Journal, 39(9), pp 87-88, 1985.

CORPORATE INFORMATION MANAGEMENT: COURSE INTEGRATION OF INFORMATION AND MBA MAJORS

H. Charles Walton, University of Baltimore
Richard J. Lorette, Loyola College

The strategic use of information systems has received increased emphasis in the information systems community of late. Dickson's survey of The Society of Information Management's corporate members concluded that integration of information systems into the corporate strategic plan was viewed as the most important information systems issue. (1)

In surveys of Baltimore area executives, Walton and Lorette found that executives viewed information managers as unprepared for executive responsibilities. (2) The major deficiencies dealt with corporate understanding, human relations, responsiveness to corporate needs, and policy development. Primarily because of these perceived deficiencies, executives had increasingly given control of information systems development and operations to functional area managers. In addition, executives and managers felt insecure with information related matters.

Because of the specific needs expressed, this course was designed for three populations:

(1) the prospectives managers who needed an understanding of information policy,
(2) the potential information managers who needed a more strategic perspective, and
(3) the functional managers with growing responsibilities in the information systems area.

McFarlan and McKinney's view - that important issues were related to integration of technology in the firm - was accepted as the central theme. Studies consistently found that, if information systems were to be effective, understanding of corporate culture and individual behaviour were of paramount importance. (4) Since these areas were seen as major deficiencies among information managers, corporate culture and behaviour received heavy emphasis. McFarlan and McKinney also stressed strategic relevance, corporate culture, contingency management, and a phase approach to technology assimilation. (5) From previous experience with Cash, McFarlan, and McKinney's text, we believed that two information technologies seemed to require more depth; the technologies included were database and telecommunications.

The course was organized in a case format. This approach has been accepted by AACSB and was advocated by Peter Kauber in "A Meta Approach to MIS Research," the outstanding paper award winner at the 1985 Conference of the National American Institute of Decision Sciences. This paper recommended the use of case studies as a method for conducting research in MIS. In addition to cases, the course presented a conceptual framework or covering theory. The cases used were "Harvard Cases" or author-developed cases which followed similar formats. Generally, the cases were twenty pages or more in length and required one-and-one-half to two hours of class discussion.

All students were required to submit written answers to three to five questions on the assigned case immediately before each case discussion began. The graded responses were returned at the next class meeting. Course final grades were based upon: (1) the graded responses to the individual cases, (2) a written "in class" final case analysis, and (3) a subjective contribution evaluation.

Response to the course was excellent from both the MBA and MS(Information Systems) students. The mixture of different populations appeared to be extremely beneficial. While both degree groups profitted from the diverse student mix, the information students' added insights into the elements of corporate culture and their enhanced organizational understanding were clearly the most significant student gains.

The student population mix might alter the course response. The mix consisted primarily of working MBA students, working information majors, and foreign information students. The information majors generally were employed at lower managerial levels - with a few notable exceptions - than the MBA students. Those exceptions were experienced middle level information managers.

The course outline, with case subject areas, was developed to meet the requirements of the three groups. The course outline was:

(1) Introductory lecture without a case

(2) Comprehensive introductory case
(3) Non-technical case stressing corporate culture, strategic assessment, and Nolan's phase concept (3)
(4) Database planning and implementation
(5) Teleprocessing
(6) Office automation
(7) Centralized-decentralized information organizational structures
(8) End user computing
(9) IS manager requirements and the effect of IS mission change
(10) IS planning
(11) Controls, chargeback, standards
(12) Project management methodology
(13) Project management and application portfolio management
(14) Security and EDP auditing
(15) Make-buy and personnel development
(16) Comprehensive "in-class" case analysis

We believe the course significantly benefitted information systems majors by increasing their sensitivity to managerial, policy and corporate culture aspects of information management. We believe, also, that functional area managers and general area managers were better prepared for information oversight and policy responsibilities.

References

1. Dickson, G., R. Leitheiser, M. Nechis and R. Wetnerbe, "Key Information Issues of the 1980s," MIS Quarterly, 8:3, September 1984.

2. Lorette, R. and C.Walton, "Problems With Definitions of End User Computing," Interface, 7:3, Fall 1985.

3. Nolan, R., "Recharting Business and Computing in the Decade Ahead," The Consultant, Digital Equipment Co., January-February 1985.

4. Vitalari, N., and G. Dickson, "Problem Solving for Effective Systems Analysis," Communications of the ACM, 26:11, November 1983.

5. McFarlan, F. and J. McKenney, Corporate Information Systems Management - The Issues Facing Senior Executives, Richard D. Irwin, Inc., Homweood, Illinois, 1983.

6. McKenney, J. and F. McFarlan, "The Information Archipelago," Harvard Business Review, Sept.-Oct. and Nov.-Dec. 1982.

7. Kauber, P., "A Meta Approach to MIS Research," Proceedings of the American Institute of Decision Sciences, 1985.

THE DESIGN AND DEVELOPMENT OF EDUCATIONALLY SOUND COMPUTER ASSISTED INSTRUCTION

Janet D. Hartman
Illinois State University

ABSTRACT

The development of computer assisted instruction is a systematic process which merges the principles of system design with those of curriculum design. A computer assisted instruction package should be well-designed, the objectives to be met and the audience for which it is designed having been defined. The author should determine the functional design of the package and then proceed to develop the subtopics to be covered. Once these tasks have been completed, specific objectives should be written and sequenced. Having completed this process, the author can use the logical subdivisions and sequenced objectives to design the menu and the screens. In the next phase a programming language or authoring system must be selected. Once the programming starts the author should follow some guidelines in developing screens which result in a good computer instruction package. He should consider such principles as (1) avoiding cluttered screens, (2) allowing the user to have control of exits from the program and screen transitions, (3) placing similar items on screens in consistent positions, (4) using graphics and sound to enhance the package, (5) providing clear and accurate introductions and directions, (6) presenting material in small steps with smooth transitions between steps, (7) personalizing to the user if possible, and (10) providing consistent, appropriate feedback. Once the programming is complete, the author should design and develop any ancillary materials. The final product should be pilot tested so that needed modifications and revisions can be made. The resulting package should be both useful and educationally sound.

INTRODUCTION

As the computer becomes more commonly used as a classroom tool, the need for educationally sound software continues to rise. While the need in the classroom will continue to grow, a new market for the consumption of educational software will also grow. This is the home market. Forecasters predict that this market will even surpass the classroom market.

In the beginning, computer assisted instruction was delivered via a mainframe without benefit of color, graphics, or sound. The cost for development and delivery was prohibitive for many school systems. Thus, the use of computer assisted instruction in the classroom was very limited. The diverse capabilities of the microcomputer, coupled with its portability and reasonable cost, have led to the development of computer software with many useful and attractive features which is within the reach of most schools. Often these packages are developed commercially by a team of experts over a long period of time. In some cases, however, the entrepreneur, the classroom teacher, has an interest in developing software to support instruction in his own classroom. How does he begin to plan the enterprise, carry it through, and possibly market his product?

Since the development of a computer assisted instruction package is a systems development project, many of the useful strategies and tools which analysts and programmers use to create systems can be used effectively. Couple the design tools with some principles of educational curriculum and instruction design and the result is an educationally sound, technically well-developed package. The principles for the design and development of educational software which are be discussed below are focused on the development of microcomputer courseware, but could be applied to the development of educational courseware using another medium, for example, interactive video.

THE DESIGN PHASE

Every systems development project starts with a great idea. The concept which the courseware author hopes to develop should be examined to determine if it can be delivered well via the computer. In doing this the author should <u>consider the objectives within the curriculum</u> which are to be met and how the computer can be used to deliver the content better than, or equally well as, other methods.

Once a topic has been identified, the author must consider the audience which the package will serve. Things to include might be the special needs of the user, the reading level of the user, the age level of the user, and any other characteristics which may be useful to consider when developing the software. For example, "cute" packages which use a lot of graphics for reinforcement may be appropriate for young children, but not for college freshmen. Children without reading skills may need menus which utilize graphics, but adults probably do not. The matching of the product with the proposed consumer may go a long way in selling the product. At this stage of development the author should also consider any prerequisite skills which will be needed by the potential user of the package.

Having decided on a topic and identified user characteristics, the developer should concentrate on the functional design of the package. Will the package be drill and practice? a tutorial? a simulation? a test? Each strategy has some unique characteristics to be considered in development. Some of these are discussed below.

Drill and practice programs should ideally present a variety of levels of practice sets, with the option to select a variable number of problems for each session. The format may or may not be a game. The user should get immediate feedback on his response and be informed of his success rate upon completion of an exercise set. The main idea is to develop a flexible package which can be used by a variety of individuals in sessions of different lengths, and also by the same individual on several occasions. Drill and practice programs which provide for practice of one concept a fixed number of times with no variation in format or problems have limited use.

Tutorial programs should present concepts clearly in a logical manner encouraging user interaction at regular intervals. Concepts should be developed in small steps with smooth transitions between the steps. Ideally, tutorial programs will allow for branching to alternative paths (e.g. to a help screen, to a review section) based on the user's response. Users should be provided with an option to page back so that previous screens can be reviewed if necessary.

Simulation packages should present realistic, accurate scenarios which provide ample opportunity for the user to explore. Based on user input, the program should allow for alternate choices and give the user feedback regarding his choice. By far, simulation packages are the most difficult to create.

Some types of computer assisted instruction emphasize problem solving. Students are asked to use appropriate problem solving strategies to find answers to posed problems. This emphasis may appear alone as an instructional strategy or be embedded in a tutorial or simulation package.

Some computer assisted instruction may include several strategies within the same package. For example, a tutorial may be followed by a drill and practice session or a test.

After determining the functional design of the package, the author should use a top down design philosophy to break the topic to be covered into parts. Each component identified may end up being a single program selected from the menu or it may be one block of the larger program. The author should use a method such as Gagne's task analysis to subdivide the package. Since many computer assisted instruction packages tend to be large and unwieldy to program in one chunk, the development of a task analysis can assist the user in breaking the package up into manageable components and suggesting a possible sequence for presenting the subtopics. This process may also give the author some insight into organizing the menu of choices from which the user may ultimately select. A useful tool for representing the model of the package and the relationship of the parts to each other is the structure or hierarchy chart. This chart will provide the programmer and the user with a sense of the overall structure of the package and also the subcomponents to be covered.

Within each subtopic, specific objectives should be formulated. These objectives will further define the framework of the package and the sequencing of ideas within the package. Stating the objectives behaviorally (one of the tools from educational design) will help the developer to (a) develop the logical design of the package and (b) develop any testing tools--pre or post. For example, objectives can be useful in determining whether the sequence of ideas will be presented deductively (rule then practice) or inductively (discovery by using examples). Objectives will also be useful in identifying areas of strength and weakness when the package is evaluated.

Once subtasks have been sequenced and specific objectives have been written, the author can depict the overall design of the package showing the components to be covered and the flow of information through the package. Systems development tools such as the structure chart, data flow diagrams and flow charts can be utilized to graphically present the relationships between components and the flow of information. These documents will provide valuable assistance to the programmer who will carry out the design and will also be useful in developing written information which may appear in later documentation (e.g. user's guide, product description sheet).

Having spent a great deal of time designing the content and its sequence, perhaps 50% or more of the life of the project, it is time to begin the development phase. The author must choose an implementation language or system. High level languages provide for more flexibility of capabilities within a program, but they are usually difficult to learn for the beginner. An alternative is an authoring system which is usually relatively easy to use, but limits the format of the questions and text. Authoring languages, like PILOT, provide a middle of the road approach, since they were developed particularly for the task at hand; they still require that a language be learned, however. In selecting a language the author should consider portability, particularly if he intends to publish his software.

THE DEVELOPMENT PHASE

The author may not choose to program his design, but work with a programmer who will carry out his design. Whether he chooses to program it or not, it may be useful to have designed the screens which will carry out his design. While this effort is quite time consuming, it makes the programming much easier and also any later modifications will be easier. A sheet which shows an exact replica of the screen and has room for some additional material can be duplicated for each screen to be created. Using this template, an exact layout of text and graphics can be made. In a language like BASIC in which words may be split indiscriminantly from one line to the next, this may save hours spent in debugging to make a screen visually correct and pleasing. Correct and incorrect responses along with appropriate feedback for each can be listed below the screen layout with any specific instructions for handling them, for example, incrementing the correct or incorrect answer counter. Branching instructions can also be handled.

While all possible phases and techniques for development cannot be discussed, what follows are a number of helpful hints for the author and/or programmer.

(1) Cluttered screens are difficult to read. Double space lines on the screen. Limit the number of lines on the screen.
(2) Allow the user to control screen changes. Avoid using time loops for paging control unless timing is important to the objective of the package. Employing a prompt which allows the user to move to a new screen is a much better option for paging control.
(3) Consistently place similar ideas and questions in similar positions on the series of screens, for example, placing prompts for paging control at the bottom. Also, provide for a consistent method of responding to questions.
(4) Allow points where the user can exit if he wishes. Special keys to return to the menu or strategically placed questions for exit should allow the user to feel in control.
(5) Use graphics and sound to enhance the package only. Remember that in a group environment sound can be potentially distracting.
(6) Think of the user (and the programmer!) in accepting responses. Choose a response

format that can be used and checked easily. Unless spelling is important allow for short, possibly one key, responses.

(7) Provide for clear, concise directions in the package. Instructions for responding, any special keys, and any special characters (/ for divide, for example) should be particularly addressed.

(8) Present material in small steps providing for a smooth transition from screen to screen.

(9) Use inverse video, flashing and other "bells and whistles" sparingly.

(10) Personalize the package by periodically employing the user's name.

FINAL CONSIDERATIONS

Once the program has been written, the author is not finished. The module should be utilized in pilot tests and evaluated to determine where necessary changes may need to be made. The author might use one of several evaluation instruments which are available in order to do this systematically. Before submitting an educational software package to a publisher for possible mass distribution, the software should have accurate content, have been tested with actual users, any modifications made, and it should be free of errors. It should also be copyrighted. Any accompanying user's guide, help facilities, or documentation should also have been completed and tested. Developing a package for computer assisted instruction is a demanding process, but can well be worth the effort if it is well designed.

INTEGRATED OFFICE TECHNOLOGY: WHERE SHOULD THE EMPHASIS BE?

Dr. Lorrie Steerey
School of Business and Economics
Eastern Montana College
Billings, MT 59101

ABSTRACT

This paper discusses integrated office technology and the impact it is and will have on the office environment. It further discusses the impact of our concentrations in this area and where our concentrations should be focused.

Office automation is often referred to as the "office of the future". It is a key factor in bringing about more effective information handling at all levels of the organization (Mortensen, 1984). It involves more than just automating the administrative services personnel (clerical and secretarial).

Office automation is "generally considered to refer to the use of integrated computer and communications systems to support administrative procedures in an office environment (Olson and Lucas, 1982:838)."

Because of the heavy reliance on computers to automate the offices, many companies have placed this function under the MIS area in the organization. By N. Dean Meyer's definition, President of the Society of Office Automation Professionals,

> Office automation is more than data and word processing. It integrates a number of technologies and disciplines, including data processing and database management, word processing and administration, voice and data telecommunications, teleconferencing and electronic mail, management science, library science, facilities and space planning, human resources and training, organization design and development, socio-technical systems, ergonomics and occupational safety, psychology and sociology, and diffusion of innovations and political science...Have I forgotten anybody? Probably. As we evolve, other disciplines will grow in importance. For example, office automation also can depend on labor relations, cybernetics, artificial intelligence and more.

The push to automate can be attributed to the rapid increase in the volume and cost of office work and to the decline in the number of trained clerical workers.

> It is estimated that the U.S. offices produce 600 million computer printouts, 234 million photocopies and 76 million letters each working day. In addition, the business letter which cost less than $1 to produce in 1940 has been estimated to cost around $15 today (Seaward, 1983:27).

The shortage of clerical workers can be attributed to the expanded career opportunities for women. As women enter into other career areas, fewer and fewer of them are entering the clerical workforce. This shortage is predicted to accelerate to the year 2000.

This trend to office automation is dramatized by the fact that 85 percent of the Fortune 500 companies have formalized plans to automate their offices. These systems will include word processing, personal computers, electronic mail systems, and other office automation tools. Medium-sized and small companies have less formalized plans, but they also are delveloping strategies to automate (Olcott, 1984).

CLERICAL VS. MANAGEMENT PRODUCTIVITY

The first office automation systems concentrated on the administrative support areas of the office. Word processing systems were one of the first office automation systems to be implemented on a large scale. These areas of the office are easy to automate since they are highly proceduralized. They often involve repetitive tasks that can be easily counted.

As office automation advances it must

concentrate on automating the professional/managerial positions. In 1982, U.S. businesses spent $1 trillion on white-collar personnel and $600 billion of that compensated knowledge workers (Poppel, 1982).

MEASURING INCREASED PRODUCTIVITY

> Early studies of office automation directed almost exclusive attention on methods for increasing the productivity of clerical workers. However, current studies tend to focus major attention on problems related to the increase of productivity of managers (Mortensen: 1984).

Office productivity is often criticized for having only 4 percent increase while during the same period manufacturing productivity increased 80 percent. One of the reasons often quoted for this slow increase is the inadequate capital investment of office workers.

CHANGING COMMUNICATION PATTERNS IN THE AUTOMATED OFFICE

The automation of the office is moving us from an office which is paper-based to one which is electronically-based. This change is also changing the source of information from that of a person to that of the computer. These two fundamental changes will affect the communication patterns in the office.

Pava believes that as technology becomes cheaper and more powerful, the office will become more functional and integrated. Thus eliminating some of the traditional office equipment like the typewriter and telephone. He further states that the challenge of management is to redefine problems and articulate general policies that encourage others to view technology as more than an elixir (1984:76).

Mynett supports this view point when he states: "The challenge of the actual office of the future lies in translating opportunities of technologies to practicable realities for each of our unique business environments and human resources (1983:162)."

Perhaps the greatest impact will be upon the secretarial profession. One thing we know is that typing which was once the real pillar of secretarial work is diminishing as an activity. However, nine out of ten PSI secretaries believe that office automation has had a positive effect on the secretarial profession and will allow secretaries to be more productive (1983).

As the computer becomes the source of information and not people, how will the communication patterns change? What effects will this have on office personnel?

Will this increased reliance on the computer change the power structure in the office? Will those who have the information obtain the power? What will happen to those who do not automate?

PREPARING FOR THE AUTOMATED OFFICE

Identification of the needed skills to survive in the automated office has become an area of concern. As managers using a powerful work station are able to draft a report, create and insert charts and tables, revise, and print the text without leaving their desks or waiting for work to come back from other people, such as secretaries or staff analysts who once would have been involved, (Uttal, 1982:178) these managers will need to possess different skills than their counterparts did five or ten years ago. Liddle, vice-president of Xerox, points to four major acitivities of managers: the communication of data and ideas that are still in a pliable state and being worked on by groups of professionals; the creation of documents; the filing and retrieval of documents; and the distribution of written work.

How will the "office automation" software packages effect the skills needed by office especially professional personnel? With these packages professional personnel are responsible for maintaining their own "electronic filing system." They must index the record, file the record, and retrieve the record. In addition, they are responsible for purging their own records. What skills are needed so that these professionals are still effective and valuable computer space is not wasted with old or unneeded duplicate copies?

OVERCOMING RESISTANCE

Resistance to office automation occurs with both managers and administrative support staff. In both cases, education and participation in the decision making process are crucial. There is no substitute for feeling like part of the team. Everyone involved needs to participate in the decisions. As office automation is planned mangers should discuss plans and ask employees for ideas (McArthur, 1985:128).

Administrative support staff is generally quite favorable toward office automation. A recent study reports that secretaries were generally satisfied with the nature and the variety of their work. Further, nine out of ten report that office automation has beneficial effects which permit them to be more efficient and productive. However, both secretaries and executives agree that secretaries should be more involved in the process of equipment selection (1983:116).

Managerial reactions to office automation are not so easily identified or quite so positive. Managers do not readily accept office automation. Olcott reports these attitudes in a recent article. This includes the results of a study showing that executives and managers who do not use computers doubt whether they (computers) would affect their own productivity. In addition, managers have started to delegate computer-oriented tasks to support staff. He supports the idea that managers are frusterated with office computers. The findings indicate that managers are getting tired of inputting numeric data (Olcott, 1985:7). While using computers tends to glamorize clerical jobs, this is not true managerial jobs. Other blocks to office automation include the fear of computers and the resistance to change. The market and software must now turn their concentration to managerial software (Olcott, 1985:7).

ENVIRONMENTAL CONCERNS

Arthur D. Little, Inc. reports that "by 1990 between 40 and 50 percent of all Americans will be making daily use of electronic-terminal equipment." Other predictions have included that by 1990 about 80 percent of the labor force in the United States will interact with computers and over 50 percent will need some knowledge about computers (Mortensen, 1984:92). With this growing use of CRTs, there is a growing concern for the hazards associated with their use.

The integration of office automation into the office must also include the environmental aspects. This includes the systems furniture, lighting, interior design, and environmental controls. How much each of these contributes to the hazards of office automation is not really known.

The radiation effects suffered from daily use of CRTs is only one health hazard of concern. Non-radiation concerns include: eyestrain, back and shoulder problems, head and neck problems, arm and wrist problems, and leg problems. "Half of the secretaries who use a CRT display report that they suffer from eye strain and back or neck problems. One-fourth experience headaches or other types of stress (1983:112)."

Another area of concern is stress. As the office continues to automate and as the push for higher productivity accelerates, how will office personnel be affected?

RESTRUCTURING OF THE OFFICE

> Technology, as well as the economy, also is causing change within our office environments. Both hardware and software available to use serve to create, eliminate and alter the job titles and descriptions with which we have long been familiar (Jones, 1983:24).

As office automation becomes more prevalent, some fear that it will create a whole new set of problems for clerical workers. Since these jobs are easiest to automate, will computer technology reinforce the ghetto boundary lines for secretaries? Production in the clerical functions of office automation can become factory-like.

Nussbaum fears that office automation will increase discrimination.

> Automation of the office will bring in a whole host of completely new jobs. Now, if we were a fair society, a society that didn't practice discrimination in employment, we would find men and women equally distributed throughout these new jobs. But we're not seeing that... What office automation often means is that clerical workers are paid less for doing more (1983:32).

Secretaries and managers have different perceptions on the future role of the secretary in the organization. The managers tend to see the secretary as advancing to the status of technician, while the secretary sees the role as emerging into a management training position. This clashing of views was reported by Kanter of Yale (Kleinschrod, 1983: 7).

> the survey's biggest finding lies in the clash of views between secretaries and managers on what OA should do for them. Dr. Kanter said secretaries want OA to change their role, bringing them into more of a team relationship with their bosses. Managers, however, wish to obtain current roles, and in some cases actually want OA to keep other workers at a distance and to not increase communication with them.

SUMMARY

Concentrating on clerical and secretarial areas does not solve the problem of office productivity. Office automation must now concentrate on managerial productivity. Productivity depends on two things: (1) how much time is spent on a particular task, and (2) how important that task is to achieving the mission of the organization (Bullen, 1983:6-8).

In order for managers to accept office automation they must become "office automation literate". And, "until computer manufacturers can produce systems that managers see as indispensable, the full productivity benefits of office automation will remain largely theoretical (Olcott, 1985:7)."

BIBLIOGRAPHY

"Automation May Increase Discrimination: A Conversation with Karen Nussbaum," Office Administration and Automation, pp. 32-35, April 1983.

"How Secretaries Related to the Age of Information," The Office, p. 112 and 116, May 1983.

"A Conversation with John Connell, "Office Administration and Automation, pp. 52-54, May 1985.

Baldwin, Darrell F., Following the Natural Path to Real Office Automation," The Office, p. 166+, May 1984.

Bullen, Christine V., "Hang On, the Ride is Just Beginning," The Secretary, pp. 6-8, June/July 1983.

Connell, John J., "Managing Information in Tomorrow's Office," The Office, pp. 110-111, January 1983.

Grant, Kenneth A., "The Folklore of Office Automation," The Journal of Office Systems Research, pp. 1-6, Fall 1982.

Jones, Hester S., "Planning and Designing the Office Environment," The Office, p. 24, February 1983.

Kleinschod, Walter, "State of OA Report: Partly a State of Confusion," Office Administration and Automation, p. 7, November 1983.

McArthur, Donald W., "What Good Is Technology If Your Are Faced With Disenchanted Employees?," The Office, p. 128, January 1985.

Mortensen, Erik, "Office Automation; Agenda for training and Organizational Change," Conference, Office Systems Research Assocation, 1984.

Munson, Vernell K., "Automation Defined: Teaching Old Companies New Tricks," The Office, p. 19+, February 1984.

Mynett, Jack W., "Turning Our Technology Into Practical Reality," The Office, p. 162, January 1983.

Olcott, William A., "OA: Is the Message Getting Through to American Business?", "Office Administration and Automation, p. 7, May 1984.

Olcott, William A., "Many Executives and Managers Resist Computers--Can Anything Be Done To Turn This Around?," Office Administration and Automation, p. 7, June 1985.

Olson, Margrethe H. and Henry C. Luca, "Implications for Research and Practice," Communication of the ACM, pp. 838-847, November 1983.

Pava, Calvin H. P., "Tommorrow's Office Differenct but How?," The Office, p. 76, November 1984.

Poppel, Harvey L., "Who Needs the Office of the Future," Harvard Business Review, p. 147, November/December 1982.

Ressenbaum, Li da, "Health Effects of Video Display Terminals: the Nonradiation Problems," The Journal of Office Systems Research Assocation, pp. 7-21, Fall 1982.

Seaward, Marty Robertson, "Awakening to Office Automation," Management World, pp. 27-29, May 1983.

Uttal, Bro, "What's Detaining the Office of the Future?," Fortune, p. 176+, May 3, 1982.

BUILDING EFFECTIVE DECISION SUPPORT SYSTEMS IN TURBULENT ENVIRONMENTS

MADJID TAVANA
LA SALLE UNIVERSITY
PHILADELPHIA, PA 19141

AND

SASSAN HEJAZI
PENNSYLVANIA STATE UNIVERSITY
UNIVERSITY PARK, PA 16802

ABSTRACT

This paper presents ways a Decision Support Systems Builder tackles problems in turbulent environments by problem observation, information processing, thinking, and model building.

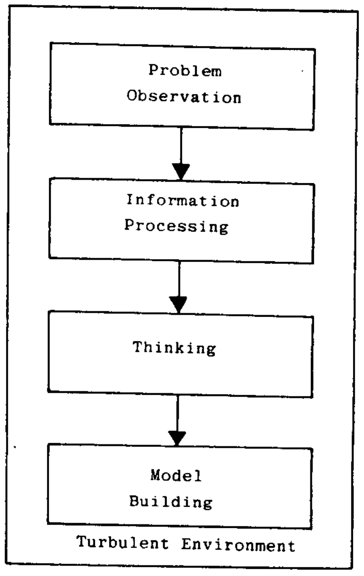
INTRODUCTION

A main problem in designing Decision Support Systems (DSS) is that the environmental contexts in which orginizations exists are themselves changing-at an increasing rate, under the impact of technological change. We live in a society with a turbulent environment. Emery and Trist (1973) describe turbulent environments as the most complexly textured environments in which individual orgainizations, however large, cannot adapt successfully simply through their direct interactions.

The degree of turmoil and chaos and the amount of information in the environment has been increasing at an exponential rate. Any orginization to survive in such an environment has to develop effective and up-to-date DSS. Failure to produce such systems might contribute to that organization's demise.

The DSS Builder should devise systems to respond to environmental changes which affect various levels and types of decision processes. Whether or not an organization can deal with it's turbulent environment depends on the ability of it's DSS Builder's full understanding of external and internal variables that enter the decision making processes. The nature of the DSS, however, would vary depending on the type of applications. DSS can be utilized in either institutional or ad hoc environments for varying managerial levels such as operational, or strategic planning. Nature of decision making at each level can also vary from structured to unstructured.

This paper discusses ways a DSS Builder tackles problems. Discussions will be divided into four topics: DSS Builder as an observer, as an information processor, as a thinker, and as a model builder. (Exhibit - 1) Several issues are raised for each of the approaches being used.



DSS builder as an observer. The first thing to be done before DSS Builder can solve a problem is the recognition of the problem itself. He has to scrutinize the situation of the problem and should be able to describe it as clear as possible. The more information gathered from observation, the more useful it is for subsequent steps.

There are two issues that should be raised here concerning DSS Builder's perception of a problem or a system. The same system observed by different people is very likely to be described differently. Sometimes the descriptions given by each observer varies so much that we do not know they are talking about the same system. These discrepancies depend very much on each individual's background, experience, culture, and education. One eminent Russian Scientist pointed that the moon, as described by American and Russian astronauts, looks different. In some cultures, people do not have a word for 'snow' since they have never experienced any, while other cultures like Eskimos have more than ten words for 'snow'. Volcanic explosion may be described as an act of God by some tribemen, while it is obvious that it is a nutural phenomenon.

The DSS Builder frequently relies on the descriptions provided by the decision makers. Benneth (1976) indicates that decision makers have trouble describing a decision making process. He suggests that the designer should not require the decision maker to describe the decision making process before the DSS is built. The DSS, on the other hand, should help a decision maker to conceptualize a problem. The DSS Builder's observation should take into account specific styles, skills, and knowledge of decision makers. Carlson (1979) proposes that DSS should be flexible enough to be applied to a variety of styles and designer should build the system so can be tailored for each decision maker.

DSS Builder tries to percieve the system the way he thinks it is, which may or may not be correct. Such biasness will certainly create discrepancies in the way he or she sees a system. This bianess will be the first kind of error blended into his problem even before he models or solves it. In order to reduce the level of biasness in the observation stage, the DSS Builder, therefore should understand the interactive forces among various subsystems within the overall system.

Another issue in DSS Builder's perception of a system is his limited sensory capabilities. Due to this limited perception, he might find a problem too complicated or even too simple for fitting it into the overall observation. As suggested by Keen (1980), this problem can be reduced by utilizing a User-Builder interactive approach within the pre-defined system. The observer should allow for future problems and issues to be arised in future by the user.

<u>DSS builder as an information processor.</u> DSS Builder's capacity to process information and solve problems is finite. The environment is extremely complex that he can not make decisions without having constraints imposed upon his environment to help make it managable. It is therefore impossible for the behavior of a single, isolated individual to reach any high degree of rationality. The number of alternatives he must explore is so great, the information needed is so vast, that givens or premises must be created and accepted if he is to behave rationally.

In approaching a social problem, DSS Builder does not have the ability to consider all the independent variables that describe the behavior of the problem since the number of variables is enormous. He has set a boundary for his problem subjectively, consider all the variables within this boundary, and disregard the rest. Variables within the boundary are those that have more influence and more relevance to the outcome than the rest which are ignored. In doing so, DSS Builder transforms a real problem to a reduced, approximated, and simplified one.

One classical example that is always quoted is Newton's approach to formulate law of gravitation. His simple formula that describes the motion of a free falling object has the form of $v=\frac{1}{2}gt$. In deriving this formula, Newton ignored all interactions of objects in the universe except for the one - the interaction between the falling body and the earth. Had he considered more than two ojbects at the same time, he would never reach this closed form solution.

The methodology noted above should be used in analyzing each specific sybsystem. The DSS Builder, however, should recognize that every DSS is itself a subsystem of some larger system. DSS Builder must not try to construct a subsystem to meet certain objectives when they really need to be working with a broader system. For example, let us consider a customer order shipping system in a manufacturing environment: this system includes shipment planning, order entry, accounts receivables, inventory control, and production planning. A narrow focus on just one of these subsystems will not tend to best results.

<u>DDS builder is a thinker.</u> DDS Builder's thinking process can be categorized into two classes - programmed and nonprogrammed. He tries to fit his problem into a programmable process if possible. His habit, standard operating procedures, and organizational structure help create conditions for programmed decisions. Routinized problems fall into this category. The problems can be solved repeatedly by the same algorithm without major modifications. These problems are usually of institutional nature and developed by highly technical development teams. Institutional DSS usually provides support to a large number o f users.

Nonprogrammed thinking does not have defined or solid structure and problems of this type require DSS Builder's continuous interaction with users. He tends to break a nonprogrammed problem into several programmed ones if possible since the problem is easiest to be tackled in this way. He will try to fit each programmed part into the existing algorithm he has in mind and solve them within the overall system objectives. All the results from these programmed modules are combined to yield a final result. Sometimes, if the unstructured problem does not overload his information processing capability, then he needs not to break it into parts since considering the problem as a whole will always be better than breaking it into parts. Non-programmed decision processes can be associated with a variety of decision types. As indicated by Gorry and Scott Morton (1971), for instance, cash management as applies to operation control, or product planning as applies to strategic planning represent nonprogrammed and unstructured decision processes.

Simon (1960) believes that the process of non-programmed decision making will undergo a fundemental revolution because of basic discoveries about the nature of human problem solving. He predicted that subconscious or emotional variables are not as important as we think. Perhaps the complexity of the problem solving process that makes its outcome so impressive is a complexity assembled of relatively simple interactions among a large number of extremely simple basic elements.

DSS builder as a model builder. At present, the decision making modeling is performed in accordance with two schemes - the classical scheme and the new scheme.

The classical scheme roots deeply in economics. It lies on the assumption of absolute rationality of a decision maker. In fact it is limited to the consideration of the decision making process as the choice from the given set of alternative maximizing or minimizing some objective function which is the criterion of decision optimality.

In contrast, the new scheme of decision making bases on the concept of limited rationality of a decision maker. In accordance with this scheme decision making is treated as a prolonged multi-stage process. This process can be presented in the form of a sequence of the following stages: information gathering, identification of the problem, formulation of goals, generation of decision alternatives, evaluation of possible future consequences of gathered decision alternatives, and choice of an alternative. The classical scheme considerably improvishes and misinterprets economic reality. Simon pointed out: Executives and their staff spend a large fraction of their time surveying the economic, technical, political and social environment to identify new conditions that call for new actions. They probably spend an even larger fraction of their time, individually or with their associates seeking to invent, design, and develop possible courses of actions for handling situations where a decision is needed. They spend a small fraction of their time in choosing among alternative actions already developed to meet an identified problem and already analyzed in terms of their consequences.

Hence, if we regard decision making only as the choice of the best or optimal decision from among present alternatives, most important, creative stages of the decision making process are ignored.

Actually the classical scheme of decision making can be regarded as a particular case of a new one. In such case decision making center identifies problem's situation which fits well into one of existing optimization models, then formulated the objective consisting of maximizing of some scalar indicator. This scalar indicator must be expressed quantitatively and unambiguously. It is this indicator that plays the role in evaluation decision alternatives. It is assumed that information gathering performed by the decision of making center enables us to describe precisely the whole set of alternatives and also to estimate the consequences of results of these alternatives in terms of criterion of optimization. Resource spendings on gathering and processing of such information in the models of the considered class are not taken into consideration.

A variety of classical scheme for conditions of uncertainty, takes into consideration expenditures on gathering and processing of information, it in fact disregards most important creative stages of the decision making processes.

The new decision scheme provides wider possibilities to reflect the new consequences of uncertainty. The environment for the decision making center is constituted from elements of all these subsystems of the economic system. The uncertainty of the decision making center confronts is connected with the uncertainty of the decision making center confronts is connected with the uncertainty of the decision making center confronts is connected with the uncertainty of the decision making centers image of it's environment. The process of decision making for such a center is mainly the process of uncertaintly elimination in relation to the identfied problem's situation. Uncertainty of the environment dependent on the imcompleteness of information theoretically can often be removed altogether if we make an assumption concerning unbond potentials of decision makers and limitlessness of resources including time resources allocated on the solution of an emphasized problem. Under uncertainty not one but several objectives not reduced to a single scalar indicator are formulated. Information gathered with limited resource expenditures at the moment when the decision is bound to be taken enables us to formulate only an imcomplete set of feasible potentially effective alternatives.

The evaluation of consequences of generated alternatives is performed approximately, with conformity of such evaluation to reality being very low. Scarcity of resources will invalidate the substantive comparative alanysis of generated alternatives with the aim of choosing an optimal one. This fact makes inevitable the limited rationality of decision making centers displayed in the search for at least one alternative for which formulated objectives are supposedly met. Decision making processes concerning problem's situation of the same class in conditions of uncertainty are implemented sequentially utilizing all the information accumulated until the moment of decision making. For the processes of decision making in conditions of uncertainty the choice of the actual decision making moment is of great importance. Delay in decision making will permit to gather additional information, for certain parameters of environment to become apparent, but excessively delay in decision making can have negative effect.

Models based on the classical scheme are often regarded not as descriptive but a prescriptive, or normative. Therefore, in such models we are not interested in the way decision makers behave but the way they should behave. The transition of descriptive set-ups into normative ones and vice versa is quite justified and based on the fact that certain phenomena are artificial in a very specific sense. They are only because of the system's being moulded, by goals or purposes, to the envitonment in which it lives. If natural phenomena have an air of 'necessity' about them in their subserience to natural law, artifical phenomena have an air of 'contingency' in their mailability by environment. Shortcomings of the classical scheme occur not because it is normative but because it ignores many important decision making factors. Seemingly irrational real decision which are much better portrayed by the new decision making scheme than from a more general point of view resulting in tru rationality of limited rationality. Goal ambiguity, like limited rationality, is not necessarily a fault in human choice to be corrected buy often a form of intelligence to be refined by the technology of choice rather than ignored by it. The task is to single out reality rational aspects of the actual decision making processes and introduce them into normative models.

CONCLUSION

Four major roles of DSS Builder - observing information processing, thinking, and model building - have been discussed. Approaches being employed in each step have been investigated with their advantages and disadvantages being pointed out. Most issues root from limitations of DSS Builder's capacity and the way he behaves in accordance with his belief. Another important thing for any DSS Builder and analyze all the strengths and weaknesses of his approach before he starts implementing a course of action.

REFERENCES

Ackoff, R.L. (1960) Unsuccessful Case studies and why. Operations Research 8(4): 259-263

Alter, S. (1980) Decision Support Systems: Current Practice and Continuing Challenges. Reading, Mass.: Addison-wesley.

Bennett, J. (1976) Integrating Users and Decisions Support Systems. Proceedings of the Sixth and Seventh Annual Conferences of the Society for Management Information Systems, University of Michigan.

Carison, E.D. (1979) An Approach for Designing Decision Support Systems. Support Systems. Data Base 10(3): 3-15.

Emery, F.E., and E.L. Trist (1973) towards a Social Ecology. New York, NY: Plenum Publishing Corporation.

Gorry, G.A., and M.S. Scott Morton (1971) A Framework for Management Information Systems. Sloan Management Review 13(1): 55-70.

Keen, P. G. W. (1980) Mythical Man-Month Revisisted. Center for Information Systems Research working paper.

Simon, H.A. (1960) The New Science of Management Decisions. New York, NY: Harper and Row.

DSS and Strategic Management: Not a Match Made in Heaven

Betty J. Storey
James Madison University
Harrisonburg, Virginia

INTRODUCTION

The impetus for this paper was provided by a statement in a business policy textbook which indicated that upper-level managers relied predominantly on verbal communication and instincts rather than on MIS efforts to make strategic decisions for their organizations. "Formal prediction techniques such as forecasting, modeling, and using an MIS are viewed skeptically by strategists . . . ." (6) Having returned to college after 15 years to obtain a degree in management information systems, I had a vested interest in discovering if this statement was indeed accurate and, if so, where the problems exist. I began a search to examine the problem areas and identify the responsible parties. My subsequent review of the current literature has revealed a number of difficulties inherent in the area of decision support systems.

Background

In the beginning . . . the computer field consisted of cumbersome, large computers with limited capacities designed primarily to process transactions. Most data processing managers were secure in their areas of expertise, operating in a very controlled environment, providing numbers to management.

Up in the executive suites, there were managers making decisions about the company's future through a reliance on historical data (what worked before should work again), instinct, emotion, etc. Management was vaguely aware of the DP department but certainly did not consider it an organizational resource.

Then the "computer age" moved to the "information age," and technological possibilities became endless. New hardware and software products hit the market in astonishing numbers. The data processing department now became the management information systems department with the once clearly defined boundaries removed and user expectations multiplied. Management was besieged by a need to make dynamic, strategic decisions in a volatile environment. Everybody wanted to play the game, but nobody knew what the rules were.

MIS Angle

It is important to keep in mind the evolutionary nature of computer usage. Most data processing originally took place in industries' accounting departments. Computerization replaced the tedious, rote number crunching involved in many bookkeeping transactions. The models to be followed were very structured: John worked 40 hours at $2.50 an hour, FICA tax was 4%, insurance premiums were $9.50, etc. While the technology was state-of-the-art, the work to be done was simplistic--issuing John's paycheck.

In the early 1970's, the term management information systems came into existence to differentiate between transaction processing and the managing of information within the organization. The former clearly delineated tasks and boundaries started to fade. By the late 1970's, "the term decision support systems (DSS) evolved . . . . " (1) As defined by Brookes, et al., decision support system is "a management information system designed to support decision making in specific problem areas by a particular individual or class of persons." (1)

In addition to the technological advances, the work place has undergone a phenomenal change. Consider that microcomputers were not available before 1970, yet predictions for 1985 were in excess of 10 million units. (9) The day when everyone has a computer on the desk is not far away. Networking, telecommunications, information integrity, and data security are all extremely complicated areas that must be addressed by the MIS department. Accompanying these expansions is the need for training and support. User-friendly does not mean "love at first sight," and MIS will be the first to hear the repercussions if the packaged software seems alien to the user.

Management Angle

Today, management is presented with "an unforgiving and competitive environment." (2) Burch elaborates that it is not the intrigue of information systems that will force management to move forward, but the instinct for survival in a increasingly competitive marketplace, especially the foreign sector. Further, it is important to remember the unstructured nature of decision making. There are choices to be made that have not been encountered before that require timely, reliable, and relevant information.

It is not uncommon to find CEO's that still view MIS as the data processing department that merely provides numbers. Some CEO's have encountered a negative experience in IS and never give MIS a second chance. (5)

Dr. V. Thomas Dock reports on a study undertaken in 1968 by Warren Keegan addressing the information used by executives. (3) The participants were asked to identify the sources of the information they used in day-to-day applications. His findings included:
  Verbal sources provided the majority, 67%

of all important information, and 75% of that came from face-to-face conversations.

Written sources provided 27%-60% of which were from external sources (publication and information service reports).

Actual physical observation or sensory perception provided 6% of the information utilized.

Dock's subsequent study examined the sources of information utilized for decision making after an MIS came into existence in their company. His findings were as follows:

Human Sources	40%
Documentary Sources	33%
Physical Sources	27%

These findings hardly represent a marked improvement considering the technology and expertise available in the information systems field. When questioned as to their failure to maximize the use of the computer, a consensus seemed to be that if the information failed to meet any one of the following criteria: accuracy, timeliness, completeness, conciseness, or relevance; there was a distinct negative feeling toward the use of the information. Failure to meet any one of the properties was perceived as negatively as failure to meet several of the criteria.

Management is in the habit of getting "things done through and with other people," and this was also reflected in the survey. (3) Having others obtain a report for the manager rather than querying a desk-top pc was the accustomed method.

Lack of support and qualified personnel from the MIS system was a third reason stated. The majority of the managers in this study were actively involved in the planning process of their MIS system, yet they were dissatisfied with the results.

Summary

The literature confirms my initial fears that management is not fully utilizing the organization's information system to support decision theory. MIS is not developing rapidly enough to keep pace with the technological advancements and the increased needs and demands of users.

A power struggle seems evident. Management is resistant to relinquishing the old ways of strategic decision making. MIS is not evolving into DSS smoothly in part because of their reluctance to turn over control of part of the information system.

CONCLUSIONS AND RECOMMENDATIONS

Management must acknowledge that they are not doing all that they could be doing in allowing the organization's information system to enhance the decision-making strategies. CEO's must give up the "go with your gut" philosophy towards strategic planning. Today's environment is too dynamic to allow the competition to get the advantage because management is unsure of the resources available in the pc sitting on the executive desk. Management cannot allow one bad experience to be the excuse to return to the "old tried-and-true" methods. If the information being obtained is not what the CEO requires, MIS should be made aware of the discrepancy. Corporate information systems should be viewed as a primary organizational resource. It is predicted that prior to the year 2000, 40 percent of an organization's budget will be earmarked for information resources. (9) If the accounting department was not meeting its objectives or if the personnel department was failing to adequately staff the organization, upper level management would not hesitate to take appropriate action. However, the mystique of the MIS department seems to deter management from demanding what they need to support the organizational objectives.

The success stories of companies who have taken advantage of the technology available should provide a motivation to the doubtful manager that DSS is indeed the future of strategic management.

MIS, for its part, must share in the solution. The evolutionary nature of the field seems to have helped in creating the dilemma. MIS over the years has been in control of its information, but the time has come for a "radical new philosophy--sharing." (4) Information centers and decentralized processing are here to stay, and MIS must rethink its position as to control. Higher placement in the organizational hierarchy and increased utilization carry increased responsibility. Like the adolescent reaching adulthood, decision support systems is coming into its own but must accept the accompanying growing pains.

In addition to assuring that the information is available for supporting the strategic planning of CEO's, training and support may make the second critical difference in the successful implementation of DSS at the strategic planning level. Telling a vice president that a software package is user-friendly will not suffice if the users become frustrated because they are not able to successfully use the system. In his article, Gerald M. Weinberg avers that the key to software difficulties may be whether the organization has adequate support for users. (8) He discovers that where you find successful implementation, there is usually an official or often "unofficial" resident expert whom users feel comfortable calling upon for troubleshooting. MIS needs to either develop this network or establish its own. These individuals not only have technical skills but more importantly have the communication skills to help others less technically oriented. Weinberg notes that these "cultural brokers" have a "foot in each culture and act as a go-between whenever the need arises. . . . wherever two different cultures must interact but must overcome communication barriers." Weinberg recommends that MIS actively pursue these individuals whom he titles "systants" (blending system with assistant), and he sees this type of individual as a key to the success of user-friendly software.

Vice-president Irwin J. Sitkin of Aetna Life & Casualty Co., "now views his company as 'one big information processing business.' The game has become 'Get the right information to the right guy at the right time to beat the other guys out in the marketplace.'" (7) This epitomizes the goal of decision support systems for strategic planning.

Communication between management and MIS is crucial to the success of DSS. A true partnership must exist in order for there to be optimal utilization of decision support systems in strategy planning. Since the technology is there and the impetus from the environment is there, now is the time for the total immersion of the information systems within the objectives of the organization.

REFERENCES

1. Brookes, Cyril H. P., Phillip J. Grouse, D. Ross Jeffery, and Michael J. Lawrence, Information Systems Design. Australia: Prentice-Hall of Australia, 1982.

2. Burch, John G. "In a Competitive World, the Strongest Weapon is Information." Data Management, February 1986: 22-28.

3. Dock, V. Thomas. "Executive Computer Use is Doomed without Five Key Properties." Data Management December 1985: 27-30. (Original study: Keegan, Warren J., "Acquisition of Global Information," Columbia Journal of World Business March-April, 1968, 35-41.

4. Ewing, Tom. "MIS's Future: The Challenge of Change." InformationWEEK January 20, 1986.

5. Freedman, David H. "The CEO & MIS: A Promising Partnership." Infosystems February 1986: 28-30.

6. Glueck, William F. and Lawrence R. Jauch. Business Policy and Strategic Management, 4th ed. New York: McGraw-Hill, 1984.

7. Harris, Catherine L. "Information Power." Business Week 14 October 1985: 108-114.

8. Weinberg, Gerald M. "Systant: A New Name for an Old Role." 3, 12 (1985): 50-53.

9. Zmud, Robert W. Information Systems in Organizations. IL: Scott, Foresman and Company, 1983.

VALUES OF COMPUTER ASSISTED INSTRUCTION FOR STUDENTS AND TEACHERS

Pedro M. Rocafort

Undergraduate Senior
Major: Computer Science
College of Business Administration
Kent State University
Kent, Ohio

ABSTRACT

This paper presents the use of computers in education via Computer Assisted Instruction (CAI). It evaluates the feasibility of implementing CAI as an alternative mean of instruction. An overview of computers in education, and the definition of Computer Assisted Instruction are included. The paper discusses the differences of CAI from traditional modes of instruction. Conclusions and recommendations are presented.

INTRODUCTION

What can be done to provide students with a variety of instructional resources? How will the incorporation of new learning styles and modes of interaction affect our schools? As the task of education is now getting more difficult than ever, advances in technology are making it possible to improve existing modes of instruction. Computer Assisted Instruction (CAI) provides a variety of instructional strategies and delivery modes that are useful and exciting means for expanding learning opportunities. This paper will discuss how Computer Assisted Instruction can improve the students' learning capabilities and the educational environment as well.

A major emphasis in this research has been on the computer as a medium for delivery of instruction. Students need to acquire basic knowledge in order to function in society and to advance in the educational process. This basic knowledge should be transmitted as quickly, efficiently, and interestingly as possible. The computer can help to perform this task in a very positive way.

While the cost of education is daily increasing, the cost of computer technology has continued to drop dramatically. The challenge to the instructor is to become familiar with such technology . In the same way, he has to learn to incorporate it effectively in his subject area. To help in the achievement of such a task, the research breaks down into three main topics: an overview of computers in education, what is CAI, and how CAI differs from traditional modes of instruction.

Computer Assisted Instruction might be viewed as a sophisticated, important medium for instruction. But like any other media, CAI must be used where it offers an instructional advantage.

COMPUTERS IN EDUCATION: AN OVERVIEW

A remarkable aspect of education in the past four years, has been the rapid increase of computers as part of the instructional process. All indications are that the influence of computers in education will continue to increase in the succeeding years. The National Science Foundation (NSF) has calculated that elementary and secondary schools have more than 500,000 microcomputers at the present time. The NSF has also predicted that there will be more than one million microcomputers in classrooms by 1986. These numbers clearly demonstrate that "increases in computer technology and in the utilization of computers in business and industry have begun to make educational leaders aware of the urgent need for computer literacy among our students" (Gleason, 1981).

A recently published study by the Center for Social Organizations of Schools, the John Hopkins University, revealed that besides the educational impact of computers, teachers also value the social role of computers in the educational process. The presence of microcomputers in the school, teachers reported, means that:

- students are more eager to go to school
- students are able to work more independently, without teachers' assistance
- students tend to cooperate better with each other.

It has been demonstrated that our society is one where computers are playing an increasingly important role. Therefore, it is essential that students be acquainted with computers and their capabilities. Professional organizations such as the Association for Computing Machinery, the National Association of Secondary School Principals, the National Education Association and the National Council of Teachers of Mathematics have issued policy statements encouraging schools to develop instructional computing throughout the school curriculum.

This movement toward computers in education, clearly reflects one thing. As the number of students and what they need to learn increase, teaching and learning strategies, at the same time, have to be modified.

CAI: WHAT IT IS

OPERATIONAL DEFINITION

Computer Assisted Instruction is defined as an instruction medium which meets the following criteria.

The computer serves as the primary vehicle for the delivery of the instruction. It is used as a direct aid and to support the student's performance in different subjects. CAI makes use of the computer to individualize learning and assist students in learning at their own pace in a variety of fields. CAI programs can provide many strategies for individualized instruction. Drill and practice, practice tests, automated testing, tutorials, and simulations are examples of the variability of CAI within the instructional setting.

A student may use a CAI program to learn new material or to supplement some material previously taught by the instructor. Depending on the design of a particular CAI program, the student can take a pretest on a given subject. The computer will judge his performance and will take some kind of action based on the results (see figure 1). Moreover, some CAI programs allow to measure the student's performance before and/or after a specific lesson. The teacher may require the student to take a pretest and/or a posttest to check if the goals of the lesson have been achieved. In addition, individualized records for each student may be maintained by the computer.

In CAI the student and the computer are in direct communication with each other. This process is called interaction, and it places CAI into a category of instruction shared with human instructors. However, the computer is a poor substitute for effective human interaction, but it is far superior to most other nonhuman forms of instruction. A well written CAI program is capable of modifying the presentation of the instruction to match the capabilities of the student. Contrast this with other forms of instruction (a video tape, or perhaps even a classroom lecture) which do not involve interaction and proceed at a fixed pace, with no consideration for the needs of the student.

Flexibility is an outstanding CAI aspect. Depending on the program design, students have the option of taking the lesson or test during their free time at a self-chosen pace. When microcomputers are accessible throughout the campus, the student has the choice for location and thus privacy. The lack of progress in learning or testing will then not embarrass because there are no witnesses. The test and lessons can be repeated until the student is satisfied. Experience with the CAI mode of instruction has shown that "the best results come from accurately matching difficulty against performance so that students never become bored by a succession of easy tasks or overwhelmed by a succession of incomprehensibly difficult ones" (Steinberg, 1984).

According to a report by Gerald Bracey, published in the November/December 1982 issue of Electronic Learning, "students learn more when using computers than when using conventional classroom instruction" (Bracey, 1982). The report also showed that CAI saves students time in attaining the required minimum levels of knowledge and skills without a loss of student achievement. The median student time saving shown in the report was about 30 percent. Whit CAI, fast learners can attain the required levels of knowledge and skills in less time than an average or slow student.

One of the goals for education is to help students acquire as much knowledge as possible during the time they spend at school. Seen this way, CAI benefits education by increasing the effectiveness of a particular instruction for fixed periods of student time in school.

FEEDBACK

One of the unique features of CAI, and one of its most promising instructional characteristics, is its interactive capability. Each student can be required to respond to each question. The computer, in turn, can be programmed to react to each response with an individualized message. This message, called feedback, may be as simple as yes or no. It may be as elaborate as an explanation of why the student's answer is incorrect and how to get the correct answer. Moreover, feedback may be transmitted through animated graphics or a statement of the student's score.

Feedback can serve two functions in CAI: (1) to provide motivation and (2) to provide information.

Feedback as motivation

Some students are motivated by rewards for correct responses. Feedback can serve as such reward and can encourage the student to continue with the lesson. Likewise, other students like to know how they are doing in comparison to the rest of the class. Feedback about their standing, either in achievement in the particular lesson or in progress through the CAI lessons, is motivational for such students.

Feedback as information

Feedback can provide students with two kinds of information. (1) It can tell the student if the answer if right or wrong and (2) it can provide corrective information. When the computer judges a response correct, there is no need to write extensive feedback explaining why the response is correct. A simple "Right" or an equivalent is all the informational feedback needed. In contrast, when an answer is incorrect, feedback will not only inform students that they are wrong but will also provide corrective information. The purpose of feedback is not just to help students get a particular answer right, it is also to overcome errors in understanding the instruction. Since "learning is not an easy activity, the student needs every possible stimulation and aid to make learning as pleasant as possible" (Bork, 1981).

CAI: DIFFERENCES FROM TRADITIONAL MODES OF INSTRUCTION

Prior to mentioning the conditions that are most favorable for the use of computers in instruction, examples of its uses and misuses should be examined. It would be wise to try to use CAI in settings that would maximize the outcome of a particular instruction and to consider other alternatives in situations where limitations prevail.

The most obvious difference between CAI and classroom instruction is that in CAI the student is interacting with a machine rather than with another human being. The communication is different. In the classroom, only one person answers each question posed by the instructor, while others sit and listen. Some choose to respond as little as possible, and others monopolize the answering. In CAI, every student can be required to respond to every question. Some students do not respond in the classroom because they are afraid of being wrong and being embarrassed or ridiculed by the others. In CAI, learners answer in the privacy of their interaction with the computer terminal.

CAI is an individual activity, and thus the benefits of group discussion are lacking. In the classroom, students can learn from one another's responses as well as from the instructor. In CAI the lesson itself is the only source of information.

The mode of communication is different in the classroom, where students receive information both orally and visually: they hear and they see. When students are to follow a set of written directions, teachers sometimes read the directions aloud to the class and explain them in greater detail. Even when students are expected to read the directions by themselves, they can ask the teacher for clarification, if necessary. In contrast, in CAI most directions are currently transmitted visually: students see but do not hear. They depend only on reading text and interpreting graphics to find out what to do.

In the classroom students usually respond by speaking or writing on paper or blackboard. In CAI students respond most frequently by pressing keys on a typewriter-like keyset or, less frequently, by touching a specially designed display screen or even pressing a button on a paddle.

These differences between classroom learning and CAI greatly affect the responsibilities of the lesson designer. Directions must be particularly precise and understandable because the student will not necessarily have a teacher to turn to for further explanation. If the lesson does provide supplementary instruction or practice exercises, the display on the screen must communicate that information to the student; informing him that more information is available and how to access it. For example the message might be, "Press key E for additional examples". Interaction is an absolute essential in CAI. The lesson has to ask questions and judge responses to monitor the students' progress as well as to maintain their attention.

Computer Assisted Instruction is individually paced, in contrast to traditional classroom instruction, which is group-paced. In group instruction, a student who does not understand a concept must move along with the rest of the class although not ready to do so. A student who learns quickly must sit and wait for the others. Not so in CAI, where each student can work at an appropriate level of difficulty and proceed in the lesson at a pace that is appropriate for him. If a student finds trouble with a subject, remedial alternatives and extra practice are available. A student who learns quickly does not have to waste time listening to repetitions of material already learned. Such individualized instruction is an outstanding feature of CAI.

CONCLUSIONS AND RECOMMENDATIONS

It has been demonstrated that the use of computers for instruction is making a dramatic impact upon education. This reflects a step ahead in the instructional process. All indications are that the influence of computers in education will continue to increase. This movement toward computers in education is reflecting, that as the number of students and what they need to learn increase, teaching and learning strategies have to be modified.

Computer Assisted Instruction is aimed to provide effective and efficient individualized instruction in support of classroom teaching. Whatever strategy the designer for a CAI program chooses to employ, the end result will be the same; to provide the student with the necessary ability to function in a particular task. These CAI strategies will not replace the instructor, but will change his role and make it more powerful.

On the whole, the computer is one more medium for instruction. What it can do well, it can do exceptionally well. However, one must be objective and should not consider Computer Assisted Instruction the panacea for today's educational problems. There is no single solution to problems as complex as these. In return, CAI must be viewed as a substantial innovation in, and for education.

Different Courses of Action in CAI Programs

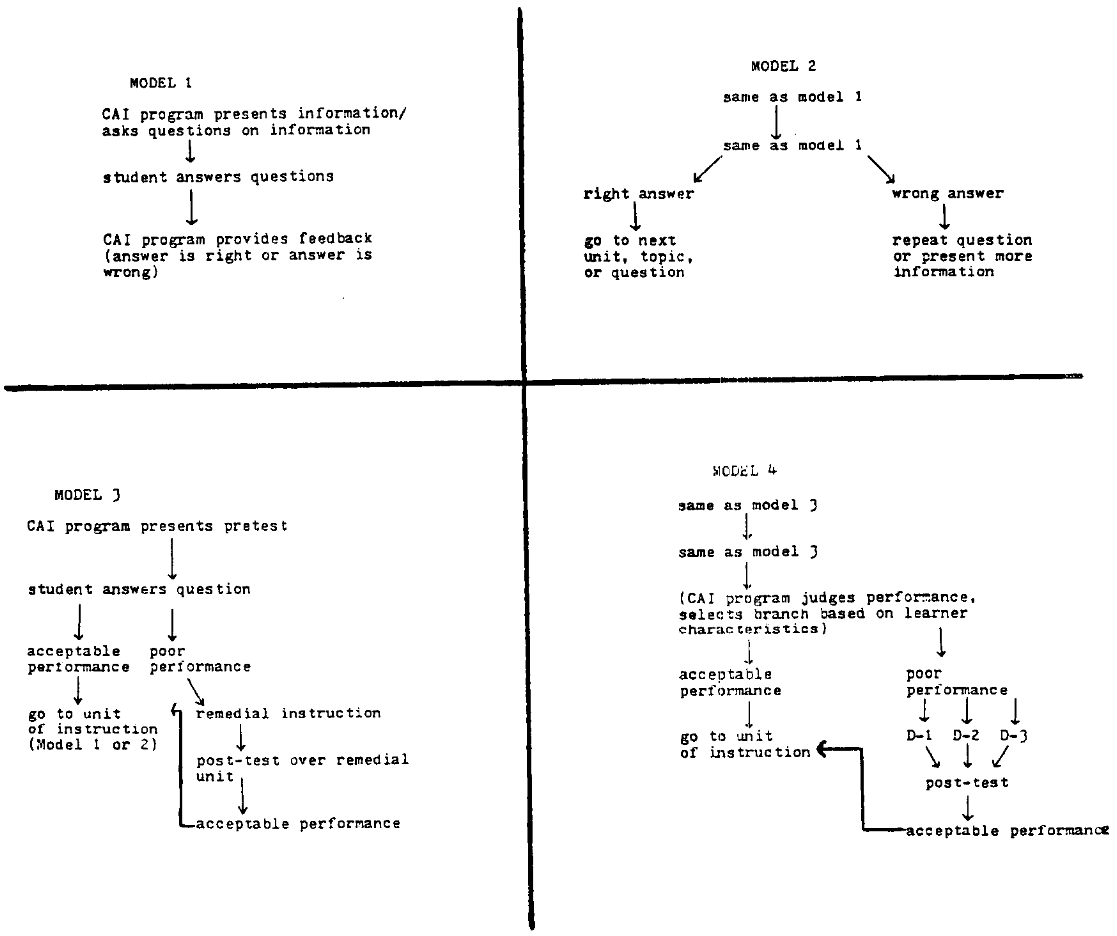

MODEL 1
CAI program presents information/ asks questions on information
student answers questions
CAI program provides feedback (answer is right or answer is wrong)
MODEL 2
same as model 1
same as model 1
right answer
wrong answer
go to next unit, topic, or question
repeat question or present more information
MODEL 3
CAI program presents pretest
student answers question
acceptable performance
poor performance
go to unit of instruction (Model 1 or 2)
remedial instruction
post-test over remedial unit
acceptable performance
MODEL 4
same as model 3
same as model 3
(CAI program judges performance, selects branch based on learner characteristics)
acceptable performance
poor performance
go to unit of instruction
D-1
D-2
D-3
post-test
acceptable performance

Figure 1

REFERENCES

Bork, Alfred. "Education and Computers: The Situation Today and Some Possible Futures." T.H.E. Journal 12 (October 1984):92-99.

Bork, Alfred. Learning With Computers. n.p.:Digital Equipment Corporation, 1981, pp. 1-29.

Bracey, Gerald W. "Computers in Education: What the Research Shows." Electronic Learning, November/December 1982, pp. 24-27.

Gleason, G.T. "Microcomputers in Education: The State of the Art." Educational Technology, July 1981, pp. 21-23.

Grayson, Lawrence P. "An Overview of Computers in U.S. Education." T.H.E. Journal 12 (August 1984):78-83.

Lear, Erich. "Computer-Assisted Instruction: Getting Started and Staying Compatible." New Directions for Teaching and Learning 15 (September 1983):49-57.

Luskin, Bernard J., Dr.; Gripp T.H.; Clark J.R.; and Christianson, D.A. Everything You Always Wanted to Know About CAI But Were Afraid to Ask. n.p., 1972.

Steinberg, Esther R. Teaching Computers to Teach. New Jersey: Lawrence Erlbaum Associates, 1984. pp. 94-142.

Taylor, Robert, ed., The Computer in the School: Tutor, Tool, Tutee. New York: Teachers College Press, 1982, pp. 53-56.

LOGICAL DATABASE DESIGN METHODS IN THE SMALL-SIZED ORGANIZATIONAL ENVIRONMENT

Chang-Yang Lin
Engming Lin
College of Business
Eastern Kentucky University
Richmond, Kentucky 40475

ABSTRACT

The generally accepted database design methodologies are impractical on the environment of small-sized organizations because they are too sophisticated to be utilized by end users. A simplier method of logical database design is developed in this paper. The method is easy to learn and can be used by end users as well as by DP professionals. One main advantage of using this method is that end users need no knowledge about semantic modeling abstractions and normalization techniques often appearing in textbooks.

I. INTRODUCTION

Textbooks on database design usually commence with an analysis of end user requirements and then reduce such requirements to a physical design through a sequence of structured steps [1]. Various techniques and procedures are utilized in each step. Most of these techniques require special skills therefore only some DP professionals and not end users are able to utilize them best.

Of various development stages, logical database design is generally considered as one major factor in determining the success of database installations. The fact that benefits from database installations in medium- or large- sized organizations have not been realized has suggested that design methodologies need to be coordinated by a top-down planning. This has increased the complexity of databases. In a small organizations where there is a lack of DP professionals, end users who have to design their own databases are not able to employ such design strategy and therefore many databases with high redundancy like a traditional file environment are likely to be created [2]. To overcome the problems, special methods are needed so that they can be easily utilized by end users.

The purpose of this paper is to develop a method for end users or DP professionals in the design of logical databases. The method is simpler and more straightforward than those suggested in textbooks. Section II discusses the environment of small-sized organizations from which four steps of logical database design are derived in Section III. Finally, Section IV presents a conclusion.

II. ENVIRONMENT IN SMALL-SIZED ORGANIZATIONS

A small organization typically has fewer and simpler functions than its larger counterpart. The number of entities are then substantially reduced to the extent that databases should not be too complicated.

The detailed, sophisticated methodologies could create several problems. First of all, it will take a lengthy operation to develop an information system by using such methodologies. End users that need immediate results from the system may not be tolerant to the situation where the development process takes too long.

Secondly, the sophisticated methodologies are too difficult to be learned by end users. End users today in many small organizations must play an ever-increasing role in the development process. They are even required to build the system themselves. The reason for this is that there is a lack of DP professionals in small organizations.

Innovative advancement in hardware technology along with the increasing availability of Data Base Management Systems (DBMSs) have simplified the end users' task. The increasing technology has reduced computer costs; more and more small shops and independent professionals such as law firms and accounting firms can afford to own small-sized computers, in particular, microcomputers. As more and more small firms own computers, the supply of DP professionals is becoming relatively shorter.

DBMSs are being developed and installed at an ever-increasing rate [3]. Among various micro-based DBMSs almost all modern systems are relational. These relational systems are easy to learn and many of them provide the facilities such as non-procedural query languages, report generators, and application aids. Given the widespread acceptance of the micro-based relational systems, a simple design method is vital to end users to enable them to take advantage of facilities provided by such systems.

Figure 1 summarizes the characteristics of the environment in a small-sized organization.

Functions:	Fewer and simpler.
People:	Lack of DP professionals. End users taking an active role in the development process.
Software:	Micro-based relational DBMSs.
Hardware:	Microcomputers.

Figure 1
Environment: Small-sized Organization

III. THE FOUR-STEP LOGICAL DATABASE DESIGN METHOD

Databases in the environment illustrated in Figure 1 should not be too complicated. The design methodologies often appearing in textbooks are certainly correct, but they are too sophisticated to be acceptable by end users. Thus, the environment in a small-sized organization requires a change in design methods.

Because tabular representation of data is easily understood by end users, an organizational data may be designed as a set of integrated tables (relations). This set of relations can be easily converted to a physical database running on relational DBMSs.

The goal of logical data modeling is then to create a set of relations without any redundancy, and from which end users can easily derive information they need. Although designing a logical database is like an art, there are patterns and procedures that can be followed to achieve the goal. The four-step process, where each step can be further divided if necessary, is derived as follows (see Figure 2).

A database is constructed to satisfy user needs of information at the present as well as in the future. Therefore,the first step is to specify user requirements. End users are better than DP professionals about specifying their own needs. In many cases, end users need certain reports to conduct their jobs. Some reports, for example, are: a mid-term deficiency grade report; an exception report on spare parts when the on-hand quantity of each item is too low. In many other cases, the user

Step 1: Specify User Requirements

* Identify reports and other documents that will be derived from a database or will serve as inputs to the database.
* List specific questions that end users will need answers from the database.

Step 2: Identify Entities and Attributes

* Identify the entity classes.
* Specify the important attributes that describe the entity classes.

Step 3: Represent Proper Relationships Among the Entity Classes

* Determine the relationships.
* Identify the intersections as new entity classes and the effects of such classes as attributes.
* Merge unnecessary entity classes.

Step 4: Review the Derived Logical Database Model

* Ensure that user requirements can be derived from the model.
* Ensure that each fact is represented in one place.

Figure 2
Logical Database Development Process

requirements can be derived from specific questions that require immediate answers from a database. Some questions, for example, are: "What student has the book Data Base Management checked out?"; "What grade did Chuck receive in the COBOL course?" The more users know about the reports and questions, the better the database will be able to be constructed. Thus, in the first step, list as many questions and necessary reports as possible.

The questions and reports are then used, in the second step, to decide what entities and attributes are to be included in the database. Each question or report might be related to various entities. An entity class is a class of distinguishable things, such as "student", "course", or "book". Once the entity classes are identified, the attributes associated with each of the entity classes are described. One rule to decide what attributes to be included in an entity class is to ask what is important about the entity classes and what we need at the time we design it. We shouldn't too worry about the future conditions because the power of DBMSs enables databases to be enlarged as new conditions arise without changing the existing applications.

The third step is to represent the relationships among the entity classes. With relationships, various entity classes are integrated so that more questions can be answered. Relationships are characterized in terms of the number of each entity class that can participate. There are three relationships that might exist: one-to-one, one-to-many, and many-to-many. For instance, "A student can take many courses" and "Each course can have many students" can be represented as many-to-many relationship between the "student" class and the "course" class. In database design, however, only one-to-many relationships are allowed to exist. The other two's are simply incomplete design patterns that must eventually be converted into one-to-manys. If a many-to-many relationship exists between two classes, an intersection class is created. This many-to-many relationship is converted into two one-to-manys where both entity classes are related to the intersection class. The intersections are used to modeling events where each of them occurs at a specific instant [4]. Date of occurrence along with the effects of intersections, thus, may be included as part of the attributes. An example of the intersection class is a "grade" class that can be used to record students' performance on the courses taken during various semesters. The one-to-one relationship indicates that there are too many entity classes. When we found a one-to-one relationship between two classes, we merge them with just one entity class.

Relations and attributes identified in the third step are combined to form a rough logical database model. This model must be reviewed before it is converted into a physical database. One major goal is to ensure that end user needs can be derived from a logical database. The review step can also mean the kind of reduction to a "normalized" logical database model. The goal is to make sure there is no redundancy in the user data, that is, each fact is stored in only one place.

If the logical database model is unsatisfactory, the feedback from the review step becomes an input of the first step, and the development process will continue until a satisfactory logical database model is derived.

IV. CONCLUSION

The four-step method is very easy to learn and can be used by end users or DP professionals to develop a logical database for small-sized organizations. One advantage of using this method is that it takes less time and effort than the methodologies illustrated in most of textbooks. Another advantage is that end users need no knowledge about semantic modeling, normalization, and other design techniques. However, a training course of a few days for end users is still needed to emphasize why entities, attributes and relationships are vital to the organization and what just they are. There is no point in making design process more complicated than necessary.

REFERENCES AND FOOTNOTES

[1] For a detailed discussion of the generally accepted design methodologies, see I. T. Hawryszkiewycz, Database Analysis and Design (Chicago, Ill: Science Research Associates, Inc., 1984), Chapters 3-7.

[2] For a discussion of the problems and characteristics of various environments, namely, traditional files, application databases, subject databases, and information databases, see J. Martin, Managing the Data-Base Environment (Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1984), p. 30-31.

[3] R. M. Curtice, and W. Casey, "Database: What's In Store?" Datamation V. 31, No. 23 (December 1, 1985):83.

[4] F. Sweet, "Objects and Events," Datamation V. 31, No. 18 (September 15, 1985):152.

DECISION SUPPORT SYSTEMS: A CATALYST FOR THE SYMBIOTIC EXECUTIVE OF TOMORROW

Dr. Kuriakose Athappilly
Associate Professor
Business Information Systems
College of Business
Western Michigan University

ABSTRACT

The evolution of information technology--from the days when data processing provided a mass of computer printouts to the present day when the manager can generate effective information by pressing a key on his/her desktop computer--calls for a new breed of executives with a new style of decision-making. These are the executives who can appreciate as well as understand the advances offered by today's information technology in terms of analytical and computing skills and knowledge. But this new breed does not come into existence as a natural outcome of this technology. What makes the formation of this new breed of executives a reality is the much heard of but "less understood" concepts of Decision Support Systems (DSS). DSS rightly understood and intelligently practiced will certainly take the definitive role of a potent catalyst to bring about what is styled as the "Symbiotic Executive of tomorrow"--the unintimidated managers, the managers who rely on analytical quantitative knowledge and computing skills rather than the so-called "hunch" or "experience." And it is the academia, especially Business Information Systems, that can properly nurture and nourish DSS in today's business environment.

INTRODUCTION

Information technology involves producing information from inputted data by ordering and manipulating the data into an organized and meaningful format. In order for information to be meaningful for decision making in general, it must contain the following characteristics: accuracy, timeliness (up-to-date), availability (in terms of response time), conciseness, decision relevance, and completeness. Through recent hardware and software advances, the quantity of information that can be generated in various applications has been improved significantly with regard to data availability and shortened response time. Accuracy is a factor which is more or less controlled by the data itself or the actions of the user. The other factors, however, have not been addressed as well by technological changes and advances up to this point. This is because while information systems and technology have evolved in terms of sophistication and speed, they have not focused on any other areas of improvement. In other words, until recently, information technology has gone through an evolution of efficiency, but has not experienced an evolution of effectiveness.

THE EVOLUTION OF INFORMATION TECHNOLOGY

As the information technology field has evolved through various generations, the sophistication and power of information systems have improved dramatically. Yet, in the middle of this technological explosion, the needs of the final users and of managers who are in need of useful information have been all but ignored. When the initial wave of management information systems emerged, they became quite popular with managers, but were best suited only for routine data manipulation and retrieval (not decision making). Operations Research/Management Science (OR/MS) systems technology focused on a little more on problem solving, but only with regard to structured situations and those problems that lend themselves more toward simple recommendations. They did not require management intuition and insight to help solve the problem. As a result, while computer equipment technology continued to increase in efficiency, the effectiveness of this technology remained limited because emphasis was still placed on the equipment rather than the needs and requirements of the users of the equipment.

The concept of decision support systems entered into the business climate and started a turnaround in the information technology.

THE THEORY OF DECISION SUPPORT SYSTEMS

The DSS is the advanced stage of the information systems evolution. The DSS is designed to interact with the user/manager's decision and thinking processes in order to help the user/manager arrive at better decisions. The underlying principle of DSS is decision support for the manager's insight and intuition rather than mandating a particular decision or solution path. The DSS provides assistance in solving semi-structured problems. The manager's judgment remains an essential element in the decision process.

The goal of DSS, meeting the needs, requirements, and specifications of the user, can be considered the ideal model for the new direction that information technology as a whole can and should take in future applications.

DSS AND THE ESSENCE OF INFORMATION TECHNOLOGY

Decision support systems can be considered the essence of the concept of information technology as applied to its fullest. This is true because the DSS concept stems from the needs and requirements of the user/manager and is used as the basis for the entire system design process. The original definition of the applications that require the use of the system through the programming and maintenance of the system, and providing for future expansion and modification are tied to the needs of the user/manager.

Subsystems of a DSS

DSS are best suited to semi-structured problem solving situations that require management insight and judgment in order to arrive at a viable decision. The DSS is made up of three basic subsystems: the data subsystem, the models subsystem, and the dialog subsystem.

The data subsystem should be able to handle data-base basics; additions and deletions, as well as maintain external data, portray logical data structures, and handle personal and unofficial data for personal judgment experimentation. The model subsystem catalogues and maintains the decision models. It can create new models and can interact with the data-base to produce decision scenarios that will assist in providing decision support. The dialog subsystem provides the linkage between the user and the system through action languages, display/presentation languages, and user knowledge reference bases to assist in the use of the system.

The prime element of all DSS is the user, not the subsystem itself. Consequently, it must generate higher quality information for the user/manager not only in terms of speed and efficiency, but also effectiveness for use in management decision making.

Adaptability of the DSS

As stated previously, the prime traits of useful information are accuracy, timeliness, availability, conciseness, relevance, and completeness. Also, it can be concluded that timeliness and response availability are indicators of system efficiency, while conciseness, relevance, and completeness are indicators of system effectiveness (accuracy being an indicator of overall quality). DSS are able to address not only those factors that influence efficiency, but also those of effectiveness because of the DSS emphasis on user requirements and needs throughout all phases of the system's design and development. Through the user's total involvement in the design of the DSS, he or she is able to provide direct input and influence regarding the decisions to be supported or information to be produced by the system. Subsequently, this will directly affect the conciseness and completeness of the information and determine the relevance of the information to the problems that are being addressed by the system.

One of the key attributes of DSS is adaptability to a variety of contexts. This means that the DSS must be flexible enough to adapt to any changes that may occur in the user/manager's needs and decision making requirements. If this flexibility is properly built into the system, adapting to changes in user needs, the problem environment, and also changes in equipment technology can take place with relative ease. As decision making needs and required applications evolve in the DSS setting, the necessary changes can be implemented into the system. Throughout the DSS design process, however, while the technological aspects have some significant role in constructing the physical system, the user/manager's needs and reasoning processes make up the logical or conceptual system. Effective design and implementation of DSS will require an integrated effort between the technicians who configure the hardware/software system and design the quantitative models and the user/managers who are in a better position to determine what data, dialog, and eventual output might be best suited to their own thinking and reasoning processes. Through this interface between the technical aspects that constitute the physical system and the decision-making processes of the user/manager that constitute the logical system, decision making is made more efficient and more effective. This emphasis on greater effectivenss in information processing means that DSS can be considered to be the tangible expression of information technology. Nonetheless, information technology and DSS are entering a still greater technological evolution which will lead to further enhancement of information efficiency and effectiveness in the future.

THE FUTURE OF DSS AND INFORMATION TECHNOLOGY

Through emphasizing effectiveness more than efficiency, flexibility, and direct user control rather than system control, and informal processing rather than formal structuring, DSS has become the ideal concept of information technology. And yet, both DSS and information technology are still going through further evolution and enhancement through advances in not only hardware and software technology, but also in the user application potential for such technology. As it is currently being applied, the concept of DSS is only a partial realization of the immense potential of the ideal DSS (which, in theory, would be totally interfaced with the manager's reasoning power and could be adjusted by the user/manager at will, whenever user needs changed).

Recent advances in hardware technology have led to a further miniaturizing of hardware components and memory chips while retaining equal or greater memory capacity and input/output/CPU processing speed and capacity. Improvements in networking capability that have taken place recently have led to expanded multi-user application of DSS, particularly where semi-structured problems are encountered at multiple levels of an organization. Further integration of software applications such as word processing, data base management, spreadsheet modeling, telecommunications, and data communications with networking and other micro systems applications will lead to the development of true "self-contained" DSS packages which can then be installed and customized to suit whatever information needs the user/manager may have. This is borne out by the current popularity of "psuedo-DSS" software packages like Lotus 1-2-3, Symphony, Framework, Enable and Jazz/MacIntosh. Along with this expansion in hardware and software will come a similar expansion in the types of business applications that can be performed through the microcomputer.

There will also be an evolution with regard to the types of systems that information technology will encompass. At the present time, there are development projects in Europe and Japan involving the following areas: microelectronics (including fiber optics), computer automated engineering (CAD/CAM), office automation, automation software, and expert systems (artificial intelligence). Potentially, the most significant of these areas for business are the areas of office automation (and related software) and expert systems. Office automation techniques are already being used in many companies, particularly with the functions of data base management, word processing, and telecommunications. Expert systems, in theory, will be most applicable not only in semi-structured situations where DSS have been quite useful, but also unstructured problem decision situations that require an extended usage of intuitive reasoning and do not lend themselves well to structured decision modeling. This new breed of system will place a still greater premium on emphasizing the user/manager's needs and specifications in order to design a system which will provide more effective support information for a still wider array of decision problems. Therefore, while the DSS concept as we know it is currently the essence of what direction information technology should be headed, it is by no means the final solution in the area. Instead, it is the launching point for a still greater evolution not only of DSS, but also of all forms of information technology in terms of new dimensions and new applications of this ever-growing and dynamic technological field.

DSS: VISION AND MISSION

The question facing managers at this time is: How do we reach this ideal goal of decision support systems? The answer lies in the formation of a new breed of managers (who may be called the symbiotic executives of tomorrow), through a sound, systematic, and thorough academic training and practical experience. This symbiotic executive of tomorrow will be the manager who possesses an understanding and appreciation of information technology and is able to combine that technical understanding with managerial skills and techniques. This will result in the mutual interdependence of managers and the technology. The model represented in Figure 1 is a guideline that may be followed by management and academia in an attempt to reach the goal of the ideal state which is presently expressed in DSS, and for the emergence of the symbiotic executive.

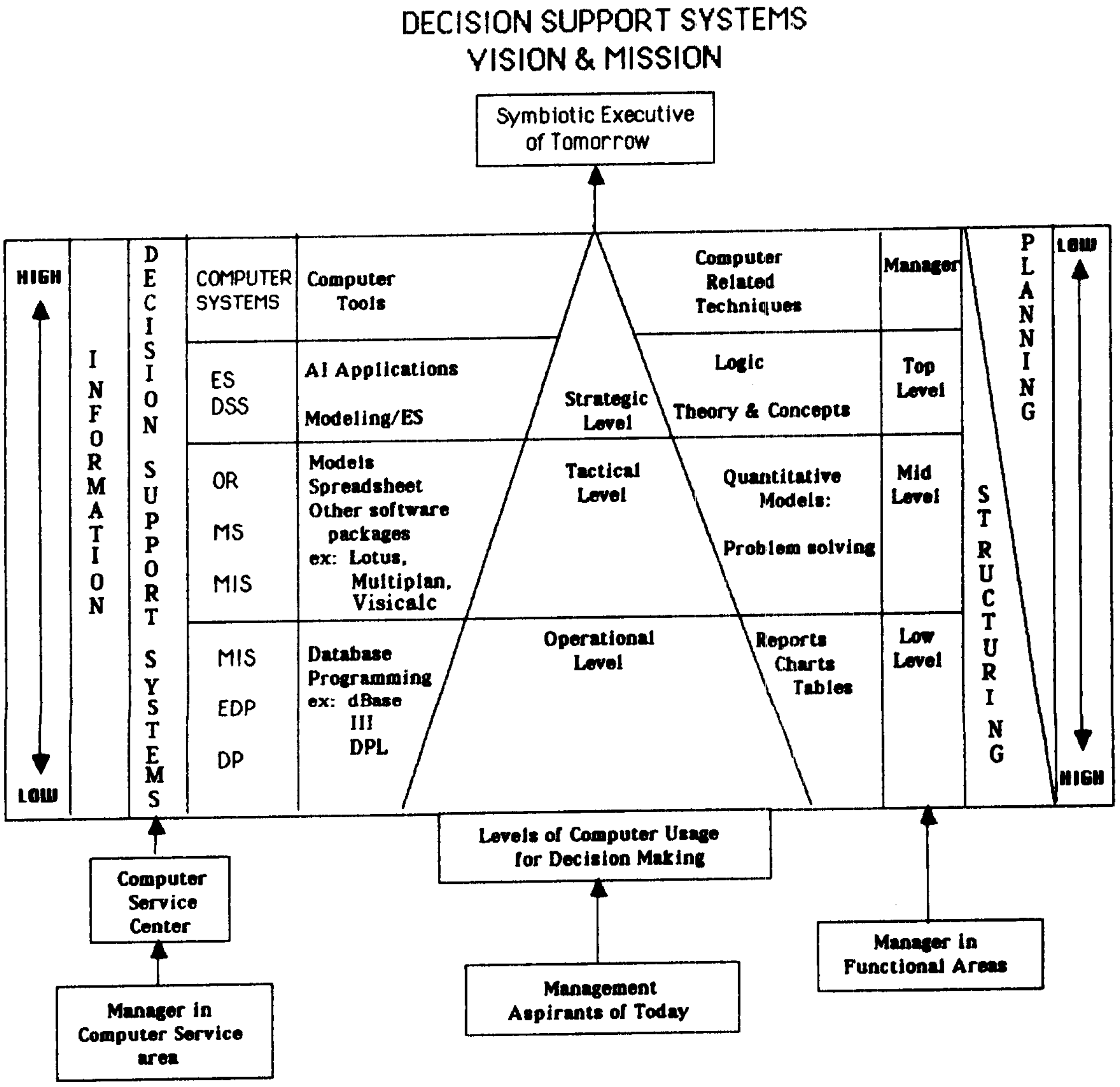


Figure 1: Athappilly's Model for the Ideal Decision Support System

Figure 1 shows the levels of computer usage for decision making (operational, tactical, and strategic) and related computer tools and techniques used at these levels. The use of these tools and techniques should begin at the academic level (i.e. the management aspirants of today) and continue up the ladder of management to the top executives (symbiotic executive of tomorrow). By encouraging the use and understanding of computer tools and techniques at the lowest levels, the present computer myths and fears held by many managers will be dispelled.

Taking a closer look at this model, it can be seen that at the lowest management level, the operational level of computer usage, there is a high level of structuring in the tasks performed. At this level, the information generated for decision making is at a low level, meaning that the manager uses very little judgment in making a decision. The decision is spelled out for him in the generated information. At the tactical level, the techniques are semi-structured, requiring the manager to use more of his management skills and decision making ability. At the strategic level, the manager may be called upon to use the information generated by the DSS in combination with quantitative decision making methods, in order to reach his decision.

At this point, the importance of the use of these tools and techniques cannot be too highly stressed. Through the use of these tools and techniques, the symbiotic executive of tomorrow will possess the skills to use and understand the ideal decision support system of the future.

CONCLUSION

As a result of this continued role of DSS giving inspiration for newer forms of information technology and information systems, businesses in the future will be going through drastic changes in terms of their information needs and decision making. This will result in a total reworking of the methods that enterprises use to conduct business and maintain their competitive advantages. Indeed, the signs of the times indicate that businesses will no longer be operating in what is known as "the information age." Rather, they will be thrust into what is being foretold as "the knowledge age," where the technologically handicapped user/managers will be branded as "misfits."

REFERENCES

Alter, Steven L. Decision Support Systems, Current Practices and Continuing Challenges. Reading, MA: Addison-Wesley Publishing Co., 1980.

Anderson, David R.; Sweeney, Dennis J.; Williams, Thomas A. An Introduction to Management Science, Quantitative Approaches to Decision Making. St. Paul, MN: West Publishing Co., 1982.

Benjamin, Robert I.; Rockart, John F.; Scott Morton, Michael S.; Wyman, John. "Information Technology: A Strategic Opportunity." Sloan Management Review. Spring 1984, pp. 3-10.

Bennett, John L. Building Decision Support Systems. Reading, MA: Addison-Wesley Publishing Co., 1983.

Bohl, Marilyn. Information Processing with BASIC. Chicago, IL: Science Research Associates, 1984.

Keen, Peter G.W. and Scott Morton, Michael S. Decision Support Systems: An Organizational Perspective. Reading, MA: Addison-Wesley Publishing Co., 1978.

Long, Larry E. Manager's Guide to Computers and Information Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

Schnake, Marilyn A. The World of Computers and Data Processing. St. Paul, MN: West Publishing Co., 1985.

Sprague, Ralph H. and Carlson, Eric D. Building Effective Decision Support Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

Thierauf, Robert J. Decision Support Systems for Effective Planning and Control: A Case Study Approach. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

Wynne, Bayard E. "A Domination Sequence--MS/OR; DSS; and the Fifth Generation." Interfaces, May/June 1984, pp. 51-58.

ARE EXPERT SYSTEMS BASED ON KNOWLEDGE INTENSE SYSTEMS?

Dr. Gerald N. Pitts and Charles Palmer
Trinity University
715 Stadium Drive
San Antonio, Texas 78284

ABSTRACT

Expert systems are knowledge driven computer programs that use heuristics to focus on key elements of a problem to produce a solution. An expert system is a problem solving program. Harmon and Kning describe problem solving as "... the process of starting in an initial state and searching through a problem space in order to identify the sequence of operations or actions that will lead to a desired goal." In an expert system the inference engine is the vehicle that negotiates the knowledge base to arrive at the goal, a solution.

INTRODUCTION

It may be said that an expert system may be considered synonomous with a knowledge based expert system. We can offer several reasons for the emphasis on knowledge in an expert system:

(1) Most difficult problems do not have traceable algorithmic solutions. These are problems that resist precise description and rigorous analysis. Examples of these problems are planning, legal reasoning, medical. Solutions depend primarily on the capacity to manipulate problem descriptions and to apply relevant pieces of knowledge selectively."(3)

(2) Human experts achieve expert performance because they are knowledgeable and if computers are to achieve the same level of performance, then they too must embody and use the same knowledge.

(3) Knowledge has intrinsic value, it is a scarce resource. The traditional transmission of knowledge has been through a long process of internship. By placing the knowledge in computers, we can greatly reduce the costs of knowledge reproduction and exploitation; also the private knowledge can be made available for public test, evaluation, and refinement.

To sum up the correlation between an expert system and knowledge, expert performance depends upon expert knowledge. "Because knowledge is the key ingredient in solving important problems, it has features characteristic of a rare element: justifying possibly expensive mining operations; requiring efficient and effective technologies to be fashioned into products; and making the means to reproduce it synthetically a dream come true." (3)

Within the field of expert systems are several subtypes of expert systems. An expert system can be generally typed as to the overall goal of the system. There have been several expert systems developed, each with its own application. The applications have been narrowly defined, even though the techniques used may have been similar or even developed using the same development shell. These types of systems can be enumerated as follows:

(1) Interpretation - These systems offer an explanation or state description that best accounts for the observed data. Examples of applications are natural language translators, image analysis, and intelligence analysis.

(2) Prediction - These systems, when given a specific situation, will infer the consequences. Much of the application of prediction systems makes use of modeling, either actual or computer based. Examples are weather forecasting, mineral mining, traffic control, and non-destructive testing.

(3) Diagnosis systems infer system malfunctions from observables. This category includes medical, electronic, mechanical, and software diagnosis, among others. Diagnosis systems typically relate observed behavioral irregularities with underlying causes, using one of two techniques. One method essentially uses a table of associations between behaviors and diagnosis. The other method combines knowledge of system design with knowledge of potential flaws in design, implementation, or components to generate potential malfunctions consistent with previous observations.

(4) Design systems develop configurations of objects that satisfy the constraints of the design problem. Such problems include circuit layout, building design, and budgeting. Design systems construct verify that these configurations conform to stated constraints. In addition many design systems attempt to minimize an objective function

that measures costs and other undesirable properties of potential design. This view of the design problem can subsume goal-seeking behavior as well, with the objective function incorporating measures of goal attainment.

(5) Planning systems develop feasible strategies in order to make use of resources and time tables. The problems addressed by these systems could include aircraft loading, building construction, and developing pert charts. An example would be the planning of a space shuttle flight.

(6) Monitoring systems are those systems that may be developed to be incorporated within a piece of machinery or even a factory or power plant. The potential for this system would lie not only in its diagnostic ability, the system would be built into the machinery. This means that the machine or plant would be supplied by its builder with an expert overseer. The monitoring system would not only be reading all the necessary dials, it would also know if a problem is developing. The monitor would then inform the human operators of the problem and possible solutions.

(7) instruction

(8) control

specific areas where expert systems may prove to be very useful natural language translators.

While it may be said that natural language translators and expert systems are separate and distinct AI fields, it can also be said tha the translators will depend upon the continued development of expert system. The thought here is that a translator will require an expert system in order to access a knowledge base for grammar, definitions, idioms, and other language related rules and data. This knowledge base will be quite large and the search can be shortened through the use of heuristics and other expert system techniques. This subject is brought up because translators are among the most exciting application of expert systems.

"Perhaps the most important economic factor of the current age is that, as a society, we are moving away from an economy based on the manufactured and dissemination of goods to an economy based on the generation and dissemination of information and knowledge."(3) Most of the information and knowledge is expressable and stored as a natural language. Therefore by developing the ability to process natural languages by computer can greatly aid in the dissemination of the information and knowledge.

"Endowing computers with an ability to communicate with humans in natural language has been a major topic of research in AI over 20 years."(3) The major goal of creating a machine which can communicate with humans in a natural language is not yet realized. There needs to be basic research, improved algorithms, improved computer architectures, and knowledge acquisition. We have seen the development restricted context natural language interfaces with today's computers. These demonstrate the feasibility of using natural languages and that considered with the desirability, will be a force in their development.

APPLICATION OF NATURAL LANGUAGE PROCESSING

The ability of computers to communicate via natural language processing can enable computers to interact with users in ordinary language, and therefore it can make computer power available to segments of the population that do not have knowledge of the formal languages usually used for accessing computers. There can be increases in productivity by providing mechanical means for manipulating and gaining knowledge as expresses in natural language text. Other useful applications of natural language processing are as follows:

(1) Machine Translation or the use of machines to convert a document written in one natural language to a document written in another language, conveying the same meaning.

(2) "Document Understanding involves reading documents by machine, and assimilating the information the documents contain into a larger framework of knowledge."(3) This sort of device may be able to make an abstract of the document, answer questions about the document, alert interested parties about the document, and the device may be able to perform some of the duties of a librarian.

(3) "Document preparation aids could perform the task of an experienced editor, detecting errors in spelling and grammar, and suggesting ways to rephrase passages of text to make them more understandable and to make them conform to patterns of high quality language usage."(3)

(4) "Document generator, a task related to document understanding, involves translating information stored in a formal language in a computer's memory into ordinary language." (3) As an example of its ability: suppose we have a formal description of a machine in the computer memory; with this description, we could generate various manuals that would be aimed at different audiences - engineers, technicians, and users with each group getting the necessary level of details.

(5) Speech understanding that gives the computer the ability to understand verbal communication as opposed to normal keyboard input. This is probably the most sought after ability in the natural language area. And accomplishing the task of developing speech understanding, will involve the greatest use of expert systems.

As has been mentioned previously, expert systems are knowledge-intensive computer programs. They use heuristics to prune the problem space and to manipulate symbolic descriptions. They can consider a number of hypotheses, make tentative recommendations, and assign weights or probabilities to the possible solutions. Within a narrowly defined domain, some expert systems can solve difficult problems better than human experts are able to do.

Traditional, conventional programming techniques have been used in creating the typical data processing systems we normally encounter. Conventional programming uses complex algorithms to process large amounts of data. The algorithms have predetermined, step-by-step procedures that reproducable results. The results are reliable so long as the correct and complete data have been entered. The data may change with each tie period, but the data are handled in the same way each time, the same calculations are made each time, and the conclusion is always the same.

"What differentiates the expert system methodology from traditional computer programming is an emphasis on the symbolic manipulation capabilites of computers, in particular the declarative representation of world knowledge, the explicit encoding of hueristics, and the exploitation of non-numeric data structures for computer simulations."(1) Expert systems have the ability to deal with uncertain and fundamental knowledge. Additionally most expert systems can explain their conclusions, offering a greater confidence in the supplied answer. This is a significant difference, it is the difference in just syppying statistical data about solution probability as opposed to an answer based upon experience that has an explained conclusion. The significance would be realized with medical diagnosis system. "Current systems can describe the "chain of reasoning" employed by the system in reaching its conclusion."(1) The "chain" makes reference to how the system went about developing the particular decision. Usually the chain refers to the questions asked and the answers received that resulted in the intermediate conclusions reached. Because expert systems are by nature interactive systems as opposed to batch processing systems, a mid-run explanation is easy.

Additionally, there is a difference between the data base in conventional programming and the knowledge base as used by the expert system. The difference lies in the use of a numerically addressed data base as contrasted with a symbolically structured knowledge base with a global working memory. The knowledge base will be discussed further in the description of the knowledge base.

As is currently defined, an expert system then is technically a knowledge based expert system, composed of a knowledge base, a search through a problem space inference engine, and user interface.

Let's define a problem space - the problem space or "map" has all the possible states and paths that can exist when searching for a correct solution. The map thus contains all the legal states of the problem and problem solving is a process of searching the map for the stated goal. As an example, suppose there exists a problem with few operators and states such that it has few legal states (a small problem space) the direct approach can be taken by drawing a map. Then a search can be made by tracing various paths on the map to find the correct path that leads from the initial state through discrete states to the solution.

The ability to map the problem space of a problem is a viable solution if there is a "well-formed problem". "In a "well-formed problem" we know the initial state, the goal state, and operators."(2) With all the elements of a well-formed problem, all the intermediate states can be systematically generated and in theory a map can be made of the problem space. A map cannot be made when a problem is not well-formed. Problems that are not well-formed can have one or more of the following attributes:

- Not explicitly stating the goal.
- Not having discrete problem states.
- Unspecified operators.
- Undounced problem space.
- Some problems may have a time constraint.

The inference engine helps provide for one of the most significant differences between expert system programming. Conventionally, all data must be present in order for the program to function. The expert system, like human consultants and experts must work with problems for which some information is missing or incorrect.

Incomplete information must be dealt with by allowing some of the rules to fail. The failure of various rules can be allowed for by the nature of the evaluation premise. If clauses are used to test for the correspondence of a rule and the supplied information. If the link between IF clauses is the logical AND the all clauses must be satisfied before the rule can success. However, should some data be missing or a datum be incorrect, then the evaluation would terminate and the rule fails. Alternately, should the IF clauses be linked by logical OR's then the absense of data would not necessarily cause the evaluation of the rule to terminate. This activity corresponds to that of a human expert who may not receive complete information, the expert would know when to ignore missing data and when to stop and get the data. The human expert can usually deal with incorrect data by allowing the preponderence of correct data to outweigh the failure of a few rules.

The construction of a problem space composed of objects and links necessitates the creation of a solution route or a chaining through the map. The choice is between backward and forward chaining. Backward chaining is the most common method used in expert systems. The method used in backward chaining is sometimes referred to a "goal directed" as opposed to "data directed" when referring to forward chaining.

Backward chaining is used when the number of possible solutions is small and all solutions are known. The process of selecting the correct solution begins with testing the solutions in turn. The rules governing the choice of a particular solution are tested with the known information or parameters become sub-goals. The sub-goals may be either new data needed fiom the user or what the program can determine for previous data or through the testing or more rules. The solution is given based upon the passing of the necessary rules in the selected route.

Forward chaining is used when the number of possible solutions is large or when a solution needs to be constructed.

There is a revolution (that has already taken place). It is not enough that computers are used as tools for record keeping, analyzing voluminous data, or as super calculator; science is in the process of creating computers that have the ability to reason. This field is commonly known as Artificial Intelligence (AI).

"AI is the attempt to understand the nature of intelligence and produce new classes of intelligent machines through programming computers to perform tasks that require reasoning and perception."(1) The goal of AI is the production of machines that act intelligently over a broad range of activities. This ability to act intelligently is more than performing certain intelligent tasks better than humans (as in hand-held calculators), it is the ability to exhibit human characteristics as well as intelligence. These human characteristics are common sense, ability to learn from experience, ability to access a situation, ability to work with incomplete data, and work independently.

Within the field of AI lies the domain of expert systems. An expert system is a computer based program that attempts to demonstrate an expert's ability in solving problems. "These programs are designed to represent and apply factual knowledge of specific areas of expertise to solve problems."(3) Areas where these programs have been applied are in medical diagnosis of infections, studying geological reports to assist in mineral exploration, configuring DEC computer systems to tailor the computer system to customer requirements, and determining the molecular structure of a new compound. There is a great potential for these programs to be developed because of the tremendous demand for expensive and rare human knowledge.

It could be said that an expert system may be great and that everyone should have one, but it is not particularly applicable for this situation. If the expert system could either answer or assist in answering a reasonable percentage of the calls for assistance, then the human expert's time could be better spent developing more expertise instead of providing the same repetitious knowledge. This new expertise could then be added to the expert systems. Another consideration in the development of expert systems lies in the communication with the expert. "Expert systems do not tire or become cranky; they do not bluff but instead tell us the limitation of their knowledge and estimate the uncertainty of their conclusions; they process and distill the experience of many experts and apply it to our problem without bias; they tell us upon our demand what assumptions they are making and what their line of reasoning is; in short, they add breadth and depth to our reasoning and decision processes."(2)

BIBLIOGRAPHY

(1) IDENTIFYING RESEARCH AREAS IN THE COMPUTER INDUSTRY TO 1955, Robert F. Cottellessa, Noyes Publications, 1984.

(2) EXPERT SYSTEMS, Artificial Intelligence in Business, Paul Harmon & David Kning, John Wiley & Sons Publishing, 1985

(3) BUILDING EXPERT SYSTEM, Hayes-Roth, Waterman, Lennat, Addison-Wesley Publishing Co., 1983.

ROBOTICS: Our Attitudes

Philip J. Sciame

In the 1970 movie "Colossus: The Forbin Project," a frustrated Dr. Forbin laments "Frankenstein should be required reading for all scientists." He had just seen his creation, an intelligent computer, announce that the human race had to be protected from itself. Therefore, the computer, in conjunction with the Soviet computer, Guardian, would become the source of all authority on the planet.

This theme goes back to the Middle Ages and the story of the Golem. This Hebrew protector was made of mud and dust, but came to life when the name of God was placed on its tongue by a rabbi. Since its mission was to protect the Jewish people from their enemies, it went on a rampage when it saw the evil and injustice within the Jewish community. (5, pp 4-5)

Fortunately, the rabbi was able to "unplug" the Golem. "Any machine intelligent enough to pose a real threat to us will also be intelligent enough to prevent us from pulling the plug." (8, p116) Colossus demonstrated this to Dr. Forbin's chagrin.

These tales and others like them have influenced our attitudes on robots, computers and androids. Like Grace Weston in Asimov's I, Robot we fear entrusting our lives to a robot since "It has no soul, and no one knows what it may be thinking." (2,p25) Moreover, perhaps we are afraid of entrusting our economic lives to these tireless steel-collar workers.

The word robot comes from the Czechoslovakian word "robota" which means worker. (20,p2) It was introduced to the rest of the world in the play "R.U.R." by Karl Capek. (5, p5) R.U.R. stood for Rossem's Universal Robots. Like the Golem and Mary Shelley's monster in Frankenstein, these creatures were not robots. They were all androids.

"An android is a creature put together from actual human parts or parts made to look human." (5,p3) A robot, on the other hand, is harder to define due to the variations in terminology and technology. In fact, the ancient Greeks may have had a better idea of what a robot is than we do today.

The god Hephaestus (a.k.a. Vulcan) forged golden handmaidens (10, p35) and tripods, three legged stools, that would do his bidding. (5, p4) The latter is mentioned by Aristotle in the first book of his Politics when he says:

> For if every instrument could accomplish its own work, obeying or anticipating the will of others, like the statues of Daedalus, or the tripods of Hephaestus, which, says the poet, 'of their own accord entered the assembly of the gods; if, in a like manner, the shuttle would weave and the plectrum touch the lyre without a hand to guide them, chief workmen would not want servants, nor masters slaves. (1, p9)

Thus, Aristotle and the Greek poets gave us our first application of robots, slaves for the modern world. It could be slavery without guilt. It could free us from boring, repetitive and dangerous jobs. (8, p117) In fact, in the early 70's, Walter Reuther, then head of the U.A.W., gave tacit approval to the use of robots on the auto industry's assembly lines. He reasoned that they were assuming jobs that were not fit for humans. (20, p34)

Although the Greeks dreamed of robots, their technology could only produce automota. This device is a machine that can not be reprogrammed and does not respond to environmental information. (8, p2) Automatae are primarily designed for amusement, but there have been some practical applications.

Heron of Alexandria's automated theaters, circa 200 B.C. (8, p2), Da Vinci's automated lion, and the mechanical rooster perched atop the Strasburg Cathedral (20, p2) are the precursors of the animated toys, music boxes and vending machines of our era. These devices may be within the quidelines of the toy industry's definition of a robot, namely, a nonliving mechanical device that can move under what appears to be its own power. (5, p2) However, due to this limited definition, they are excluded from our consideration.

"A robot is a machine with the ability to gather, study, digest and respond to information. The last two functions are done independent of human control." (5, p3) This definition only gives us part of the picture. We must merge this explanation with that given by the Robot Institute of America:

> A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools or specialized

devices through variable programmed motions for the performance of a variety of tasks. (22, p12)

If we accept only one of these, we may have an intellectually fulfilling definition, but we would fall into the trap of our heritage. We must realize that a machine does not have to appear humanoid to be a robot, and that it does not have to be the mechanized equivalent of a Swiss Army Knife. At this time, "the average American robot is more complex, expensive and interesting than its Japanese counterparts, but Japan is far ahead of the United States in making robots that are suited to factory work." (13, p120) "The Japanese have built a vast market primarily by concentrating on simple practical applications." Americans build expensive six-axis robots and then apply them to problems that cheaper Japanese robots can accomplish. (24, p194 FF)

Part of our quandry stems from the second application of robots as described by our media. The androids of Frankenstein, Moxin's Master, "R.U.R.," Plus and Minus, The Wizard of Oz, Star Wars, et al all have very human characteristics. Asimov's Robopsychologist, Dr. Susan Calvin, explained this phenomenon on the eve of her retirement when she said, "There was a time when humanity faced the universe alone and without a friend. Now he has creatures to help him; stronger creatures than himself, more faithful, more useful, and absolutely devoted to him. Mankind is no longer alone." (2, p17)

This thought is echoed by others in the literature. (8, p113) However, John McCarthy takes a different view. The man who coined the term artificial intelligence believes that "...if a machine became too human, people wouldn't like it. If you had to start to worry about the impression you were making on it - that would be bad..." (11, p33)

The one area that the authors seem to agree upon, however, is that some laws that are similar to Asimov's Three Law of Robotics would be desirable in any humanoid companions. The Three Laws of Robotics are:

(1) A robot may not injure a human being, or, through inaction, allow a human to come to harm.
(2) A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
(3) A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Handbook of Robotics
56th Edition, 2058 A.D.
(2, p11)

Asimov's view is unusual insofar as it is one of the first positive visions of robotics. The Movie Industry in the United States has been by far the most critical of any form of automation. In "Desk Set," the first movie about automation, Spencer Tracy introduced a computer that threatened Katherine Hepburn and her staff with replacement. Other movies played the same theme until in the 60's Alan Sherman recorded the song "It was Automation," a tale of unemployment due to computerization. But it was the calmly murderous H.A.L. in "2001 - A Space Odyssey" that underlined the media's judgement.

From 1966 until 1969 the classic science fiction series "Star Trek" depicted computers and androids as being a threat to humans on twelve different occasions. In ten of these episodes the machines were destroyed. In the remaining episodes, reprogramming was the answer.

One of the stories was entitled, "I, Mudd," a tale of the ultimate in slave machines. The purpose of these androids was to fulfill the wishes of their human masters. However, the humans felt trapped in this paradise. The moral: Mankind needs the obstacles of everyday life in order to develop. Willing slaves will not make us happy. (i.e. The Protestant Work Ethic)

But willing slaves are helping Japan, a country with approximately half the U.S. population (25, p554 & p594) to outproduce many of the larger industrial nations. This is especially impressive since the Japanese expect minimal help from their female population. Women in Japan work until they are married and then leave their jobs to raise children and care for their families. "At the NEC Semiconductor plant in Kyushu, it is expected that 300 young women will enter and leave every year." (4, p62)

This turnover does not send the Japanese economy into a tailspin since the plants are highly automated. The government, the industrial complex and the labor unions regard automation not as a threat, but as a positive force and a national policy.

In Japan, the robot is a "powerful symbol of the future." The Japanese have made "Cosmo Hoshimaru," an entertainment robot, the mascot of their 1985 World's Fair. (14, p83) It signifies the belief that a second economic miracle will be dependent upon microelectronics.

Since this technology does not pose the same threats as an oil or steel based economy, Japan is looking for steadily increasing GNP rates that rise with technological advancements. (13, pp120-123)

This high-tech love affair is fueled by the entertainment robots and the children's television programming. The robots range from home hobby kits to voice-activated robot pets to robot butlers. Tomy and Bandi, two of the largest Japanese toy companies, are planning to invade America with these diversions in the near future. (14, pp83-84) As for the television robots, that invasion has already begun.

In this area, you need only know a child who watches the Saturday cartoon shows to feel the effects of the wave of microtechnology. There are six (6) half-hour shows that feature robots or computers as the friends and protectors of human beings. This means that a child watching from 8 AM to 1 PM could choose one of these high-tech shows 50% of the time. On weekday mornings, from 7 AM until 9 AM, this 50% figures continues. However, on weekday afternoons, this figure increases to 67%. (Please note that these figure increase dramatically if you include Cable Television programming.)

The two most notable of these robot sagas are "Voltron" and "The Transformers." The latter program has a force of good robots defending Earth and its power sources from the evil Decepticons. The messages are clear; the good robots use teamwork and ingenuity to defeat the self-centered and cowardly evil robots.

"Voltron," on the other hand, is a combination teleoperator and human amplifier. This means that each of the modular building blocks of this giant robot is controlled by a humanoid. In "Voltron I," a group of land, sea and sky vehicles piloted by men and women of every nationality join forces to form the "Defender of the Universe." In "Voltron III," a team of five youths command a pride of five robot lions. When necessary, these lions form the warrior robot. Teamwork, friendship, honor, duty, and patriotism to your planet tend to be the main messages. As for the typical good-versus-evil theme of most adventure shows, this one adds some "Star Wars" morality. The Evil Empire is fighting a Parliamentary Government that is led by a benevolent royalty. Of course, by the end of the half hour, the good guys always win.

The most interesting depiction of a robot, however, is the character of Man-E-Faces on the animated show, "He-Man and the Masters of the Universe." The viewer is told that this character may be a good Human, an evil Monster, or a Robot that is neither good nor evil until it is programmed.

ABC Television has also begun broadcasting one minute spots to teach children computer jargon. These messages are brought to the viewer by cuddly animated creatures called the Computer Critters. They gleefully announce at the end of each installment, "We're the computer generation."

In order to find out whether these shows are influencing children, I visited a few elementary school classes. As a motivator, I brought the Hero-1 personal robot. The children were enthralled by the robot and had some definite ideas on the subject.

I found out that robots and computers were good. If someone did make bad robots, the good robots would "short circuit them." In general, robots are our friends.

I also discovered that the children tend to want robots to be their playmates, whereas, the first question from most adults is, "Can it vacuum my rugs?"

This positive attitude towards robots is encouraging in a country with a declining birth rate. Productivity can be kept at present levels or increased by using robots to replace the decline in the work force. Costs could be lowered if the average human worker is making more than $5.00 an hour including fringe benefits. This is because the average robot costs $4.00 to $4.80 per hour. The costs include installation, maintenance, depreciation, energy, and the ability of the robot to work twenty-four hours a day. (22, pp37-56)

However, what if the decline in the labor force is not imminent. How can you save or create jobs for your human workers?

In Fayetteville, Tennessee, The Tennessee Fan Company employs robots and 85 human workers in a factory that produces oscillating electric fans. It is a subsidiary of the Matsushita Electric Ind. Co. of Japan, a firm that has demonstrated that it can run similar plants with a minimum of 16 employees. The extra 69 humans imply the cost of doing business in Tennessee. (22, p4)

As for the question of worker acceptance of robots, there have been many human/robot interest stories. Among them is the story of R.E.X. a robot at General Electric that belongs to the steelworkers union. It was voted into the union.

(8, p59) In Cleveland, Clyde the Claw, an auto industry robot, received cards and flowers at a worker-organized get-well party when it broke down. (20, p35)

The real question on Artificial Intelligence is: Are we standing at the beginning of a new revolution similar to the Industrial Revolution or are we witnessing the evolution of the species? Just as the steam engine allowed human beings to be more productive, computers and robots can free them from physical and mental drudgery. As Michael Crichton put it, "It can free them from being intellectual beasts of burden, from doing repetitive, tedious, mundane tasks." (7, p189)

BIBLIOGRAPHY

1. Aristotle. Politics: Book First, New York, Grolier
2. Asimov, Isaac. I,Robot, NY, Doubleday & Co., 1941
3. Ahl, David. "Dr. Kato and His Amazing Robot Friends", Creative Computing, 10:8, August 1984, pp115-116
4. Ahl, David. "Women's Rights? Not in Japan", Creative Computing, 10:8, August 1984, pp62-64
5. Baldwin, Margaret and Pack, Gary. Robots and Robotics, NY, Franklin Watts, 1984
6. Conway, John. "Personal Robots", Computers and Electronics, 22:12, December 1984, pp60-64
7. Crichton, Michael. "Computers and Human Evolution", Creative Computing,10:7, November 1984, pp189-190
8. D'Ignazio. Fred. Working Robots, NY, Lodestar Books, E.P. Dutton, 1982
9. Elrick, George, Science Fiction Handbook, Chicago, Chicago Review Press Inc., 1978

10 Hamilton, Edith. Mythology, NY, Mentor Books, 1940

11 Hilts, Philip J. "The Dean of Artificial Intelligence", Psychology Today, 17:1, January 1983, pp28-33

12 Logsdon, Tom. The Robot Revolution, NY, Simon & Schuster, 1984

13 Mead, Christopher. "Second Japanese Miracle on the Horizon" Creative Computing, 10:8, August 1984, pp 120-123

14 Onosko, Tim. "Japan and Technology: A Nation Looks to the Future", Creative Computing, 10:8, August 1984, pp82-90

15 Shelley, Mary Frankenstein, NY Scholastic Book Services, 1967

16 Sippi, Charles. Microcomputer Dictionary, Indianapolis, Howard Sams Inc., 1981

17 Snyder, Wesley. Industrial Robots: Computer Interfacing and Control, NJ, Prentice Hall, 1985

18 Spencer, Donald. The Illustrated Computer Dictionary, Columbus, Ohio, Charles E. Merrill, 1983

19 Stonier, Tom. "Thinking About Thinking Machines", Creative Computing, 10:7, November 1984, pp252-254

20 Sullivan, George, The Rise of Robots, NY, Dodd, Mead & Company, 1971

21 Trimble, Bjo. The Star Trek Concordance, NY, Ballentine Books, 1976

22 Ullrich, Robert A. The Robotics Primer, NJ, Prentice Hall, 1983

23 Wellborne, Stanley. "Machines That Think: They're Brewing a Revolution", U.S. News & World Report, Dec. 5, 1983 pp59-62

24 ----------. "The U.S. Robot Industry Starts to Come to Life", Business Week, November 14, 1983, pp194AA-194 FF

25 ---------. The World Almanac & Book of Facts 1985, NY Newspaper Enterprises Association, Inc., 1985, pp155-594

THE COMPARISONS BETWEEN EXPERT SYSTEMS AND DECISION SUPPORT SYSTEMS

Chi-Chung (David) Yen, Department of Decision Sciences, Miami University of Ohio

ABSTRACT

Decision support systems (DSS) and expert systems (ES) have been receiving an increasing amount of attention in the field of computer-based information system. Since they are new applications, there are controversies about the nature of these systems. Some people claim that ES is just another buzz word and that ES is similar to DSS. However, other people argue that ES is more powerful than DSS, and that ES has a lot of special features to make it unique. The purpose of this paper is to first describe the nature of these two systems and then to make a detailed comparisons between these two systems.

I. INTRODUCTION

Decision support systems (DSS) and expert systems (ES) have been receiving an increasing amount of attention in the field of computer-based information system. Since they are new applications, there are controversies about the nature of these systems. Some people claim that ES is just another buzz word and that ES is similar to DSS. However, other people argue that ES is more powerful than DSS, and that ES has a lot of special features to make it unique.

The purpose of this paper is to first describe the nature of these two systems and then to make a detailed comparisons between these two systems. This paper is divided into four sections. Section I is a brief introduction section. Some important background related to these two systems are addressed in section II. Section III, which is the main body of this paper, will concentrate on the differences between ES and DSS. Finally, section IV provides conclusions and summary of this paper.

II. BACKGROUND OF DSS AND ES

A. The concept of DSS was first articulated in the early 1970's by Scott Morton under the term "management decision system." A few firms and scholars began to develop and research DSS which became characterized as an interactive computer-based system, which helps decision makers utilize data and models to solve unstructured problems. In 1976, John L. Bennett presented an enhanced description of DSS. He said:

> The name "DSS" emphasizes the role which these computer-based aids play with respect to users. DSS provides high-level operations for retrieving data, generating alternative solutions, storing and retrieving alternatives, and evaluating alternatives. The relatively unstructured nature of the problem being attacked precludes a direct decision-making system approach.

These definitions proved so restrictive that few real-life systems completely satisfy either of them. As a result, Keen and Scott Morton (1978) gave another extended definition, which is widely accepted by most DSS researchers.

> A DSS is a coherent system of computer-based technology (hardware, software, and supporting documentation) used by managers as an aid to their decision-making in the semi-structured decision tasks.

Based on the decisions above, we can see that the nature of the DSS system is basically composed of four major points:

1) Data management capability

2) Analytical capability

3) Transportability

4) Reliability, maintainability, flexibility, and capability for incorporating new technologies.

B. Background of ES

Work on expert systems (ES) has received extensive attention recently, prompting growing interest in a range of environments. The earliest description of ES has been made of the basic concept and of the rule-based system approach typically used to construct the programs.

ES investigates methods and techniques for constructing man-machine systems with specialized problem-solving expertise. This expertise consists of knowledge about the problem domain, understanding of the limitations and scope of this domain, and skill at solving some of these problems.

ES is formally defined by Feigenbaum in 1982 as follows:

> An "expert system" an intelligent computer program that uses knowledge and inference procedure to solve problems that are difficult enough to require significant human expertise for their solution.

One year later, a better and widely accepted definition about ES are presented by Branchman, Amarel, Engleman, Englmore, Feigenbaum, and Wilkins, using seven semi-independent dimensions:

> Expertise, high level rules, the avoidance of blind search and high performance.

Reasoning by symbol manipulation

Intelligence, fundamental domain principles and weak reasoning method

Difficulty or complexity

Reformalization, conversion from a description in lay terms to a form suitable for expert-rule application

Type of task

Reasoning about self in various forms, especially for explanation.

Based on this definition, several traits which characterized an ES can be categorized:

1) Symbolic nature of the task it performs

2) A capacity to refine the skills

3) An ability to rationalize and justify its behavior

4) An ability to solve important problems involving complexity and uncertainty

5) A capacity to expand its range of capabilities

6) A broad and robust intelligence.

III. THE DIFFERENCE BETWEEN ES AND DSS

ES differs from DSS in a variety of important respects. These different respects include: 1) symbolic manipulation capability, 2) inference capability, 3) language processing capability, 4) instruction message capability, 5) heuristics searching capability, 6) reasoning capability, 7) rule base techniques, 8) the technique of separating the inference engine (control part of the program) and knowledge domain, 9) different languages/tools concentrations, 10) extensive capability of "speech recognition" and "image processing," 11) recovery control capability, 12) use of common sense knowledge, 13) different knowledge representation schema, 14) different data analysis and representation technique, 15) logic programming capability, 16) different orientations to tasks, 17) different focuses on system analysis and design, 18) use of soft information, 19) embedded models and algorithms, 20) knowledge engineering technique, 21) complexity vs. simplicity, 22) nature of the system, 23) different system philosophy, and 24) reformalization capability.

1) Symbolic Manipulation Capability

One point of ES which deserves our attention is that almost every ES project of recent vintage, ranging from natural language understanding to visual perception, has employed an explicit symbolic representation of the information in its domain of concern. As a result, general languages for representing arbitrary knowledge are becoming a point of focus in this preoccupation with using symbols for facts and information for a given problem domain.

Generally speaking, symbolic manipulation is the most fundamental contribution of ES so far. Furthermore, the first important factor that distinguishes ES from high-quality, special purpose DSS systems, is its relation to artificial intelligence (AI) in general and to symbolic manipulation in particular.

2) Inference Techniques

Inference techniques are especially unique in ES due to its capability of performing regular inferences as well as symbolic inferences from certain knowledge. Currently, it is quite common to see some newer DSS have certain regular inference capability to obtain new knowledge for further problem-solving purposes. However, in these systems, the symbolic inference capability is still very weak.

The concept of inference is nothing more than obtaining new information based on assumption-building, and justification-building. However, the results from the inference process are extremely important to the reasoning process because the accuracy of the latter process is wholly dependent on the reliable data produced by the former process.

3) Language Processing Capability

The language processing capability here is referred to as the sum total of all linguistic facilities made available to the decision maker by the computer-based system.

A language system is characterized by the suntax that it furnishes to the decision maker and by the statements, commands, or expressions that it allow the user to make (Bonczek, Holsapple, and Whinston, 1980).

Language processing capability is nothing uncommon in the current ES applications (e.g. PEARL, SAVVY, ALPS). But, it is unusual to see a current DSS have this capability. The language processing capability is definitely needed for future DSS because it allows decision makers to express themselves while limiting the permissible expressions.

4) Instruction Processing Capability (IP)

Instruction processing capability refers to as the three features listed below:

a. IP incorporates a diagnosis and debugging sub-system that specifically addresses a particular subject as the system of interest.
b. IP usually constructs a hypothetical description of the particular subject's knowledge that interprets the subject's behavior.

c. Sometimes, IP diagnoses the weakness of a particular subject and identifies remedy.

In ES applications (e.g., WEST, WHY, BUGGY, GUIDON), this capability is being extensively applied. However, in the DSS design, it is hard to find any current DSSs which have this capability.

5) Heuristics Search Capability

The ES system is a product with a computer nature. Therefore, it is no surprise to see that ES systems are more powerful than DSS systems, from the viewpoint of searching capability. There are not many DSS systems equipped with strong searching capability. Even in some DSS systems which own searching capability, the searching methods are quite primitive and limited to certain forms, such as bubble sort, sequential search, etc. Those searches are problem-oriented and seldom incorporate data structure concepts, recursive function, and backtracking function. To deal with a simple decision problem, these simple search methods will not encounter difficulty. As to complex and large solution space problems, a heuristics search must be used.

Most work done in ES is now able to use heuristics searching (e.g., GPS, puzzle-solving). In the near future, parallel heuristics search will play an important role in search-oriented problems.

6) Reasoning Ability

Another important attribute which makes the ES different from DSS is their reasoning ability -- the ability to accumulating informatin until the solution is reached. Most DP professionals agree that the lack of reasoning ability is the biggest drawback of the traditional DSS.

The importance of reasoning is increasing because of complex environments and the difficulty of obtaining hard, factual data. ES can be more or less intelligent, depending on the scope of its basic principles and the quality of its general-purpose reasoning processes. The quality of reasoning is based on the accessibility of relevant facts and principles as well as on completeness of the inference procedure and efficiency of its implementation.

In general, it is important to incorporate this capability in the future DSS design. Otherwise, DSS will not be able to compete with ES in the field of computer-based information systems.

7) Use of the Rule Base Technique

Rule base is another unique characteristic of ES. It is designed to have many interactive subprograms or subroutines, each subprogram composed of one single rule or a cluster of rules which have common characteristics. The advantages of representing the rules in a base are: a) easy reference, update, delete, and add without access to other rules, b) equal opportunity for each rule as selected by the inference engine, c) it avoids rule-conflict (e.g., some situations may occur where more than one rule will apply), and 4) it facilities the sequence of searching.

In general, breaking rules and then representing them as a base are very popular in current ES design. This rule-based technique is so prominent that one may have a misconception that any system or computer program with a component in the rule-based format is an ES. This is quite misleading.

8) Separating of Inference Engine from Knowledge Domain

Separating the inference engine and knowledge domain serves to separate knowledge of how to use rules from the rules themselves; that is, dividing the ES into an inference engine and a rule/knowledge base. In this way, the knowledge in the rule base becomes more easily identifiable, more explicit, and more accessible. If these two are intermixed, domain knowledge will get spread out through the inference engine, and it becomes less clear what we ought to change to improve the system. The result is a less flexible system. In other words, separating those two components can reduce the redundancy in the problem domain. Once, the designer codes a particular domain knowledge in the inference engine, he never needs to code it again in the rule base. It is common to see that some successful DSS systems which do not separate these two parts; that is, they have some control programs that exist in both parts for the purpose of data transferring and data communication.

9) Different Languages/Tools Adopted

Another factor which makes ES differ from the DSS is the different languages/tools adopted.

Experienced builders of an ES usually develop their systems using a variety of options ranging from general programming LISP (5th generation language), Pascal and FORTRAN (3rd generation language), to general purpose programming developed specifically for knowledge engineering (e.g., ROSIE, OPS5, RLL). However, the DSS designers develop their systems by using traditional programming languages (e.g., BASIC, FORTRAN, COBOL, etc.), and software (e.g., dBASE III, 1-2-3) and seldom develop any special programming for individual needs.

10) Extensible Capabilities With Other Systems

Because of the following capabilities, ES has more extensible capabilities to other systems (e.g., speech-recognition, speech synthesis, and image processing) than the DSS.

a) Symbolic manipulation capability
b) Specific programming can be designed to fit certain specialized systems
c) Separation of inference engine and knowledge domain
d) Symbolic inference and reasoning capabilities
e) Unique knowledge representation schema (will be covered in the latter section)

11) Recovery Control Capability

The recovery control capability here means the system should be able to repeatedly interpret the current situation, predict the future, diagnose the causes of anticipated problems, formulate a remedial plan and monitor its execution to ensure success. In other words, this system has the ability of reducing and also exploiting redundancy, and it can automatically recover from certain wrong actions. Just as human beings adjust their behavior by using past experiences, this capability allows the system to generate and test these experiences learned from previous actions.

Because of the complex tasks and multiple levels of functions, there are no very successful ES examples presently. However, if this capability is being successfully developed and implemented in the future, ES will play a dominant role in the field of computer-based information system.

12) Use of Common Sense Knowledge

ES researchers have found that common-sense is the most difficult thing to model in a computer, but is is important in the reasoning process. In the real world, ES systems, like human beings, face the need to act in spite of the lack of time, facts, and knowledge. Under these situations, common-sense knowledge is needed to draw conclusions from partial information provided.

Generally speaking, common sense knowledge is very helpful to obtain the solution if it is being used appropriately and efficiently. However, it may create an expensive mistake if the wrong information is provided. Therefore, there is always a trade-off between using common-sense knowledge or not. But, it is no doubt that the use of common-sense knowledge contributes to the unique features of ES.

13) Different Knowledge Representation Schema

There are seven different representation techniques: first-order predicate logic, semantic nets, procedural-subroutines, procedural-production systems, direct or analogical representation, property lists, and frames and scripts. Because of these seven types of knowledge representation schemes, it is very easy for ES to express the symbolic constraints.

Again, symbolic constraints are very hard for DSS systems to represent. Therefore, this capability proves to be another factor which distinguishes the ES from the DSS.

14) Different Data Analysis Techniques

Because of the interference capability, the reasoning capability, and the use of common-sense knowledge, ES systems have four special technical methods to deal with the reliable data or knowledge.

a) Probability method -- Introducing a model of approximate implication, and then using numbers called "certainty ratios" that indicate the weight of the heuristics rule (e.g., MYCIN, 1976).

b) Data correction rule -- Reasoning with partial information without compromising the exactness of predicate calculus.

c) Fuzzy logic -- Discussing the law of interference for fuzzy sets (Zadeh, 1979)

d) Situational calculus -- Actions are represented by functions whose domains and ranges are situations (McCarthy and Hayes, 1969)

15) Logic Programming Capability

From those points below, we can see that logical programming capability is inherent in ES; and hence, it becomes another difference between ES and DSS.

a) Knowledge representation schema of ES -- First-order predicate logic.

b) Data analysis and representation techniques -- Fuzzy logic.

c) System Design Languages/Tools -- PROLOG (programming logics).

16) Different Orientations of Tasks
Another characteristic of ES is the generic task that the system is built to do. Different tasks may substantially change the picture of a system's architecture.

As we know, the spirit of ES design is to design a system which can deal with thousands of cases with a similar nature. In other words, the architectural implications of the task requirements must be generic. The main point here is that intended tasks are another dimension of ES because of the characteristics of DSS is its personalness (e.g., ability to deal with a unique task) and flexibility (e.g., ability to satisfy individual user's needs).

17) Different Focus on Systems Analysis and Design Activities

The ES design is highly knowledge-driven, inference-dependent, and concept-directed.

a) Knowledge-driven

One very important step in ES design is space identification. Space identification considers the knowledge of the problem domain. Furthermore, another task of factoring the original problem into a meta-problem also represents the need for another type of knowledge -- metaknowledge of the problem, which is essential to do the factoring. Also, we need to separate the inference engine (knowledge of how to use rules) with the knowledge domain (knowledge of rules). Therefore, it is no surprise at all to say that the ES design is wholly knowledge-driven.

b) Inference-dependence

Recalling back the subsections of "inference capability" and "reasoning capability" discussed above, we can definitely conclude that the ES design is completely inference-dependent.

c) Concept-directed

Reconceptualization and reformalization are two important steps in the ES design. Both steps need the activities of modifying the old concept, reformalizing the new idea, and remodeling the sequences of different concepts.

Compared to these three characteristics of ES design, DSS design does not put it's focus so heavily on the knowledge, inference, and concepts.

18) Use of "Soft" Information

"Soft information" here refers to as non-factual information such as opinions, explanations, and rumors, expressed in text form. Managers have to make decisions about the future, but hard, factual information is historical and is only useful if the past is relevant to future directions. Soft information can add both meaning and predictive value to hard data.

From the discussion above, we can see that ES already uses the soft information extensively -- not only in the inference/ reasoning process, but also in the use of common-sense knowledge. Therefore, the sources and the potential accuracy of soft information is of much value to the design of future DSS.

19) Embedded Models and Algorithms

Because the computer nature of the ES, it is common to embed models and algorithms into ES design. However, in DSS, the percentage of current systems which have embedded models and algorithms is still low. Moreover, those DSS systems equipped with embedded models/algorithms are still restricted to the field of operations research or management sciences, such as linear programming, regression line, decision tree, etc. DSS needs to incorporate more embedded models/algorithms in areas such as automated recovery routines, automated search/trace routines, and distributed processing, to improve the productivity (effective and efficient) and dependability (reliability and high performance).

20) Knowledge Engineering Design

There are three types of knowledge engineering design approaches: factoring into metaproblems, constraints and goals, and generate-and-test enumeration.

Constraints and goals techniques are common for both ES design and DSS design. However, the other two approaches are unique to ES design. Factoring into metaproblems suggests an economy of representation by separating problem-solving knowledge from knowledge about the ground domain. Without this factoring, metalevel interactions can surface as confusing interactions between ground-level tasks.

Generate-and-test techniques are actually composed of two parts: a "generator" of possible solutions and a "tester" which prunes solutions that fail to meet certain constraints. This technique is quite useful in the inference/reasoning process.

21) Complexity vs. Simplicity

The complexity and difficulty of ES design is recognized not only by the business people but also computer science people. The complexity and difficulty of ES design come from many sources, such as the following:

a) Sometimes it is very hard to clearly separate the inference engine and knowledge base.

b) It is impossible for ES design to use a highly uniform representation scheme.

c) There are still difficulties in quantifying unreliable data and time-varying data.

d) Codes a program with recursive or backtracking functions is difficult.

e) The data used in inference/reasoning process are not accurate enough to get reliable results.

However, DSS design seems to be much easier and simpler than the ES design. As a result, complexity vs. simplicity becomes another distinction between ES and DSS.

22) Nature of the System

Our perception of ES today is analogous to our perception of computers fifteen years ago. The roots of ES actually date back to 1842 when Charles Babbage first tinkered with his "machines." It is quite clear that ES cannot be divorced from the computer.

From another viewpoint, ES generally uses the reasoning method or heuristics search method to find a solution. These methods usually are associated with data structure technique (algorithm-building). To code an algorithm in a structured type certainly is consistent with a computer's nature.

On the other hand, DSS is usually user-friendly, menu-driven, and suitable to personal needs. Also, DSS is easily operated by non-DP-trained users, and hence is easily accessible. No wonder some professionals claim that DSS systems are business-oriented.

23) Different Design Philosophy

The DSS is designed to be flexible, user-friendly, and decision-oriented at any management level; while ES is designed to be expertise in dealing with specific job. In other words, the goals of an ES are three-fold:

a) To attain high performance
b) To act like a human expert to achieve certain tasks
c) To produce high-quality results in minimal time.

Therefore, the ES should be able to:

* Solve the problem.
* Explain the result.
* Learn from experience.
* Restructure the knowledge.
* Degenerate gracefully.
* Determine relevance.

This is quite different from the DSS design philosophy (e.g., data management capability, analytical capability, transportability, and maintainability).

24) Reformalization

The usual design process for an ES is to take a problem stated in some arbitary form and then convert it into a form appropriate for processing by expert rules. This form may be a collection of data with the task of finding appropriate patterns in it, then acting in accordance with the specific domain covered by the expert. There are two kinds of reformalization:

a) Simple translation from one surface form to another (e.g., MACSYMA)
b) Complete reconceptualization of a problem or condition description.

Reformalization is a very unique characteristic of ES. However, one should be cautious about overzealous reformalization, which will force problems into models that are inappropriate for the expected solutions.

IV. CONCLUSIONS

Based on the discussions above, we can make a summary table (Table 1) below to conclude and summarize this paper.

Table 1

Summarization of the Differences Between ES and DSS

	Different Features	DSS	ES
1.	Symbolic manipulation	no	yes
2.	Inference capability	limited (no symbolic inference)	yes
3.	Language processing capability	no	yes
4.	Instruction message capability	no	yes
5.	Searching capability	limited (no heuristics search)	yes
6.	Reasoning capability	no	yes
7.	Rule base technique	no	yes
8.	Separation of inference engine with knowledge domain	no	yes
9.	Languages/tools concentration	3rd and 4th generation	3rd and 5th as well as special designed
10.	Extensible with other systems (speech recognition, image processing, etc.)	hard	easy
11.	Recovery control capability	no	yes
12.	Use of common-sense knowledge	no	yes
13.	Knowledge representation schema	no	yes (7 types)
14.	Data analysis and representation	limited (hard to quantify time-varying, environmental data)	yes
15.	Logic programming capability	no	yes
16.	Orientation of tasks	individual and flexible	generic (common and general)
17.	Focus on system analysis and design	flexible, personal, and easy accessible	knowledge, inference, and concept
18.	Use of "soft information"	limited	yes
19.	Embedded models/algorithms	limited	yes
20.	Knowledge engineering technique	no	yes
21.	Complexity vs. simplicity	simplicity	complexity
22.	Nature of system design	business oriented	computer oriented
23.	System philosophy	user-friendly	expertise
24.	Reformalization	limited (some DSSs use "prototype" design technique)	yes

REFERENCES

1. AAAI, Artificial Intelligence Conference, 1980. Proceedings, William Kaufmann, Inc., 1980.

2. AAI, Artificial Intelligence Conference, 1983, Proceedings, William Kaufmann, Inc., 1983.

3. Alter, S. L. Decision Support System: Current Practice and Continuing Challenges, Reading, Mass.: Addison-Wesley, 1980.

4. ----, "A Taxonomy of Decision Support System." Sloan Management Review, Vol. 19, No. 1 (Fall 1977), pp. 39-56.

5. Bennett, J. L. Building Decision Support System, Reading, Mass.: Addison-Wesley, 1983.

6. Bonczek, R. ; Holsapple, C. W. and Whinston, A. B. "The Evolving roles of models in Decision Support Systems." Decision Sciences, Vol. 11, No. 2 (1980), pp. 337-356.

7. ----, "Future Directions for Development of Decision Support Systems." Decision Sciences, Vol. 1 (1980), pp. 616-631.

8. Buchanan, B. G. and Shortliffe, E. H. Rule-Based Expert Systems -- The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, Mass.: Addison-Wesley, 1984.

9. Feigenbaum, E. A. and McCorduck, P. The Fifth Generation, Reading, Mass.: Addison-Wesley, 1983.

10. Feigenbaum, E. A. Computers and Thought, Krieger Pub. Co., 1981.

11. Hayes-Roth, F.; Waterman, D. A. and Lenat, D. B. Building Expert Systems, Reading, Mass.: Addison-Wesley, 1983.

12. Nilsson, J. N. Principle of Artificial Intelligence, Palo Alto, California: Tioga Pub. Co., 1980.

13. ----, Problem-solving Methods in Artificial Intelligence, New York, New York: McGraw-Hill Book Co., 1971.

14. Simon, H. A. "Searching and Reasoning in Problem-solving." Artificial Intelligence, 21, 1983, pp. 7-29.

15. ----, The New Science of Management Decision, Revised Edition, Prentice-Hall, 1977.

16. Scott Morton, "Management Decision Systems: Computer Based support for Decision Making." Cambridge, Mass.: Division of Research, Harvard University, 1971.

17. Yen, Chi-Chung Design of a Decision Support System using Expert System Techniques, unpublished dissertation.

ASSURING QUALITY CONTROL OF APPLICATIONS DEVELOPED BY END USERS WITH FOURTH GENERATION LANGUAGES

by J. K. Pierson
James Madison University

ABSTRACT

As end user computing becomes firmly established in organizations, the number of applications developed by nonprofessional programmers grows. This paper first examines the phenomenon of end user developed applications, the factors leading to the growth of end user application development, and trends in end user computing. Secondly, quality control concerns are discussed. Finally, training needs for information systems professionals and end user developers to assure responsible development of applications by nonprofessional programmers are examined. Recommendations are made for implementation of appropriate quality control training for students in educational institutions and for those already in the work world.

The development of new applications by nonprogrammers is a growing phenomenon. Predictions are that by the end of the decade over 50 percent of all new applications will be developed by nonprofessional programmers called end user application developers [2]. In the past, the vast majority of new applications were developed by information systems professionals using established systems design procedures. The trend away from professionally developed systems dictates that attention be given to training current and future end users in the responsible development of applications.

This paper examines (1) the phenomenon of end user applications development, (2) concerns with end user developed applications, and, finally, (3) recommendations for training end users in development techniques to assure quality control of applications.

THE PHENOMENON OF END USER DEVELOPED APPLICATIONS

The development of computer-based applications by those not classified as data processing professionals began as a response to the inability of data processing organizations to meet the demands for new or expanded computer-based information systems. In the mid-1970s, Schneiderman [12] argued that the increasing use of computers would cause amateurs and novices to demand easy-to-use facilities to develop their own systems. That forecast has been realized, and predictions are now being made for increases of end user developed applications both in numbers and in the percentage of total new applications. Scott predicts that over 50 percent of all new applications by 1989 will be developed by end users. Factors leading to the increase of end user developed applications cited by Rivard and Huff [10] are extensive and ongoing decreases in computer hardware prices and the development of "user-friendly" software.

Not all end users develop applications on their own. End users have been classified in several different ways [7] [8] [11]. Six classifications of end users proposed by Rockart and Flannery [11] are:

Nonprogramming end users with access to computer-stored data only through a menu-driven environment or strictly-followed procedures.

Command level users who access data in the form of simple inquiries for their own purposes.

End user programmers who use both command and procedural languages. They develop their own applications for themselves that in some cases are used by others.

Functional support personnel are individuals who have become informal centers of expertise on systems design and programming within their own functional areas.

End user computing support personnel, often located in a central support organization such as an Information Center. Their duties may include assisting end users and developing application or "support" software.

DP programmers are professionals in the centralized Information Systems organizations who program in "end user" languages. They provide service to end user departments wishing to hire "contract programmers."

McLean [8] classified systems developed by end users according to their use: personal, departmental, and corporate. In a study of time-sharing environments, Rockart and Flannery [11] note that the programs developed by end user programmers were almost exclusively developed for their own use. Such applications could be classified according to McLean as "personal" systems. There is mounting anecdotal evidence, however, that some end user developed applications are being shared with colleagues [2] [3] [5]. Applications that are shared move from the "personal" category to the "departmental" or even the "corporate" categories.

Probably the most influential factor in the increase in end user computing has been the acceptance of microcomputers in the workplace. That is not to imply, however, that applications are developed by end users only on micros--many use mainframes or a mainframe-micro configuration.

Tools employed by end user developers are often referred to broadly as "fourth generation languages." There is a debate over the exact meaning of the term. A more precise breakdown of end user tools is listed by Hicks [6]:

Integrated packages that combine data management, spreadsheet, and graphics capabilities.

Database query languages

Report generators

Financial modelling languages

Graphics languages

Application generators

Very high-level programming languages

Parameterized application packages

A wide variety of packages designed for end user application development are available for both microcomputers and mainframes.

CONCERNS REGARDING END USER DEVELOPED APPLICATIONS

Nolan [9] proposed several models identifying the stages of information system growth. The stages in his six-stage model are initiation, contagion, control, integration, data administration, and maturity. Literature in the field leads to the conclusion that end user computing [in many organizations] is at the control stage [11]. Now the need is apparent to assure that end user applications are developed with sufficient quality control procedures. This section of the paper examines concerns in controlling end user developed applications.

The trend toward user developed applications has not occurred without problems. Benson [1] in his study of over 30 private and government organizations found that data processing professionals' concerns related to the increase in user developed applications included documentation, data backup, security, and the fear that end users might become de facto programmers and thereby be sidetracked from their careers. In a report of their study of user manager satisfaction, Doll and Ahmed [4] note the importance of avoiding the development of inappropriate personal computer applications.

Rockart and Flannery [11] did an in-depth study of time-sharing users and information systems staffs in seven large organizations. They list several crucial management processes that they did not see in place in the organizations: (1) a strategy for end user computing, (2) development of end user computing priorities, (3) policy recommendations for top management, and (4) control methods for managing end user computing. Recommendations to control end user computing were made in response to the perceived concern that end-user costs are rising too fast and are "out-of-control." Additional concerns were that little attention is being given to the justification of new systems and that amateurish development projects are not well managed.

Of particular concern to Rockart and Flannery [11] are operational systems developed by end users that feed into larger systems. They suggest that there should be a "control process" to identify and highlight such systems "for consideration of careful documentation, the incorporation of necessary edit and control features, and inspection by the corporation's auditors." In other words, user developed systems that feed into other systems should be subjected to the same validation processes as professionally developed systems.

Even simple user developed systems can cause problems. Creeth [3] describes a spreadsheet application containing intricate formulas that was given by a department manager to a fellow manager who had a need for a similar but not identical spreadsheet. A minor change in the spreadsheet was made but not carried through to all the formulas that should have been modified with the result that an overhead figure was understated. A mistake of this type may not be readily noticeable but may have disastrous effects. Creeth also relates that some industry experts estimate that one out of every three reports generated from spreadsheets contains errors. Because of the fact that many spreadsheets are used for decision making that do not generate printed reports, the possibility must be considered that the estimate of one of every three spreadsheets containing errors is understated.

TRAINING END USER APPLICATION DEVELOPERS

Rockart and Flannery [11] cited the critical need for several different "types" of education:

1. the need to educate information systems personnel in the capabiities and use of end-user software.

2. the need for in-depth education of end user application developers (the end user programmers, functional support personnel, end user computing support personnel, and the professional programmers described above).

3. the need for brief, "how-to," example-based education for nonprogramming end users.

4. the need to educate line management and key staff managers so they can more wisely judge the systems to be developed by their people.

5. the need to educate top line management as to the tools, techniques, and potential impacts of end user computing.

Quality control topics that should be included in training end user application developers are listed by Davis and Olson [5]:

Adequate testing of applications--Many end users either ignore testing or have confidence in simple, cursory testing.

Documentation of applications--Often, end user applications written as "throw away" code are, in fact, saved and reused and thus need documentation.

Validation of data--The importance of insuring that only valid data is used in applications is as important as insuring that the application procedures are correct.

Audit trails--All report figures should be traceable back to their sources. Additionally, any data item should be traceable forward to its impact on results of the application.

Operating control--Quality assurance procedures should be included to insure that at the time of execution of the application all data items have been entered, that all appropriate data is processed and that the results are compared to independent figures.

Backup and recovery--Users should select software development packages that adequately enforce backup and recovery.

CONCLUSIONS

End user developed applications will grow both in number and the percentage of total new applications developed. Because such development is done by nonprofessional programmers, there is great concern that appropriate quality control measures will not be taken unless the end users are aware of the measures and their importance. Recommendations for implementing effective education for end users to assure quality control of end user developed applications are:

1. Identify personnel who have responsibilities for end user computing and match each to the appropriate "type" of training.

2. Put in place rigorous training programs based on the specific needs of each group.

3. Insure that adequate assistance is available for end user developers and that refresher training courses are available when needed.

4. Additionally, it is recommended that educational institutions stress quality control procedures of end user developed applications in all computer-related courses, particularly those for majors in Computer Information Systems and Management Information Systems.

REFERENCES

[1] Benson, David H., "The Field of End User Computing: Findings and Issues," MIS Quarterly, Vol. 7, No. 4, December, 1983, 35-45.

[2] Bromley, Robert G., "Template Design and Review: How to Prevent Spreadsheet Disasters," Journal of Accountancy, Vol. 159, No. 12, December, 1985, 135-142.

[3] Creeth, Richard, "Microcomputer Spreadsheets: Their Uses and Abuses," Journal of Accountancy, Vol. 159, No. 6, June, 1985, 90-92.

[4] Doll, William J., and Mesbah U. Ahmed, "Documenting Information Systems for Management: A Key to Maintaining User Satisfaction," Information & Management, Vol. 8, No. 4, April, 1985, 221-226.

[5] Davis, Gordon B., and Margrethe H. Olson, Management Information Systems, 2nd edition, MacGraw-Hill Book Company, New York, New York, 1985, 432.

[6] Hicks, James O., Management Information Systems: A User Perspective, West Publishing Company, St. Paul, Minnesota, 387-390.

[7] Martin, J., Application Development Without Programmers, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1982, 102-106.

[8] McLean, E. R. "End Users as Applications Developers," MIS Quarterly, Vol. 3, No. 4, December, 1979, 37-46.

[9] Nolan, Richard L., "Managing the Crises in Data Processing," Harvard Business Review, March-April 1979, 115-126.

[10] Rivard, Susanne and Sid L. Huff, "User Developed Applications: Evaluation of Success from the DP Department Perspective," MIS Quarterly, Vol. 8, No. 1, March 1984, 39-49.

[11] Rockart, John F., and Lauren S. Flannery, "The Management of End User Computing," Communications of the ACM, Vol. 26, No. 10, October, 1983, 776-784.

[12] Schneiderman, B. "Experimental Testing in Programming Languages: Stylistic Considerations and Design Techniques," Proceedings of the National Computer Conference, AFIPS, Montvale, NJ, Vol 44, 1975, 653-656.

[13] Scott, George M. Principles of Management Information Systems, MacGraw-Hill Book Company, New York, New York, 1986, 220-221.

TEACHING FOURTH GENERATION LANGUAGES WITHIN THE CIS CURRICULUM

James H. Blaisdell
Computer Information Systems
Humboldt State University
Arcata, CA 95521

ABSTRACT

Fourth Generation Languages are presently being implemented by industry at an accelerated rate. They offer new modes of system development and higher productivity. In addition to the importance of exposing CIS students to the state-of-the-art tools which are being used in industry, there are pedagogical benefits to be gained by incorporating 4GLs within the CIS curriculum; there are also hurdles to overcome.

BACKGROUND - INDUSTRY'S REASONS FOR ADOPTING 4GLs

James Martin opens the "Manifesto for Computing Professors" in his book An Information Systems Manifesto with the following admonishment:

"There is a gigantic gulf between the most useful new directions of commercial data processing and what is being taught in most universities...Orient the teaching to future real-world needs."

At present, only a handful of colleges and universities deliver a course on Fourth Generation Languages (4GLs). I am aware of none that have made such a course a required part of the curriculum. Given that only about fifteen percent of the major corporations have implemented 4GLs (Schatz, 1985), this is not a surprising state. The estimates are that by 1990, ninety-five percent of the major organizations will have implemented 4GLs. CIS curriculum managers should pro-act to this change in systems development by incorporating 4GL tools within the CIS curriculum. It is clear that whether we pro-act or react, we will be teaching 4GLs in some manner in the 1990s.

Full coverage of 4GLs is beyond the scope of this paper; however, I feel it is necessary to provide some background for those who might not be familiar with 4GLs and establish some common ground for those who are familiar with various different 4GLs.

A Definition

The most certain thing to say about defining 4GLs is that there is no agreement upon a definition. I offer the following as our agreement on what is meant for the remainder of this paper:

A Fourth Generation Language (4GL) is the use of sophisticated programming extensions to database technology. These extensions include most of the following:

a user-oriented interface
an inter-active database programming environment
access to decision support packages
use of applications generators
interfaces to different storage formats
(Blaisdell, 1986)

It is most common that the underlying data base is based upon the relational model. Upon becoming the topic of a draft ANSI standard, Structured Query Language (SQL), which has been a de facto standard, has moved one step closer to becoming the standard data base interface in 4GLs.

New Modes of Systems Development

With the advent of 4GLs, two major changes are occurring in the way applications are developed: 1) small systems are being developed by the functional end-user, and 2) systems are being developed with prototyping techniques.

Many people define 4GLs as tools for end-users; indeed some of the available products are targeted specifically at end-user programming. In addition to reporting on the seven year applications development backlog, James Martin has estimated that fully seventy percent of the applications required by end-users are small (Martin, 1982). Typically, these programs are a few hundred to a couple of thousand lines of COBOL code in size, and they perform simple manipulations and produce a report. Using the non-procedural programming component of a 4GL, an end-user can meet his own requirements in a fraction of the time required by a conventional systems analysis and design effort. With the non-procedural language the programmer, whether CIS professional or end-user, specifies "what" is required rather than "how" to perform the task.

The second mode of system development which is fostered by the availability of 4GLs is the prototyping of applications. One of the things that has become clear from thirty years of using the conventional systems analysis and design life-cycle is that the user can only infrequently fully specify the performance requirements at the beginning of a project. The result is that as requirements surface or change, we either accept changes to the design and modify the project or we force the user to accept less than he learns he wants as the project unfolds. In prototyping a user and a CIS professional work together, inter-actively, with interactive tools to build a working model of the application. The working model allows the user to identify what is really needed. The prototype can be refined and become the actual application software, or it can serve as the requirements specifications for a recoded full-scale system.

Increased Productivity

In both of the new development modes, as well as in the use of 4GLs within the conventional systems analysis and design life cycle, the issue is productivity. Through each language generation, language developers have sought to provide applications developers with an increase in productivity. Typical claims for using 4GLs instead of COBOL are a ten-fold increase in productivity (Bozman, 1984). Reported performance varies depending upon the application from twice to as much as eighty times the productivity of COBOL (Martin, 1985). Considering that the implementation of structured techniques yields only about a twenty-five percent gain in productivity (Jones, 1979), it is no wonder that industry is embracing a tool that claims productivity increases of an order of magnitude.

PEDAGOGICAL REASONS FOR TEACHING FOURTH GENERATION LANGUAGES

In the Information Systems Manifesto for Computing Professors, Martin admonishes us to, "Orient the teaching to future real-world needs...Teach students languages such as FOCUS, SQL, and PROLOG, not BASIC" (Martin, 1984).

Issues of increased productivity and use by end-users routinely captures the attention when considering 4GLs. Certainly it is important that we educate our students using state-of-the-art tools. However, in addition to the productivity issue, there are other important aspects in teaching 4GLs.

Higher Level Language - Closer to the Real-world Problem

Language impacts problem solving in two important ways. First, the correct symbolic representation of a problem can aid in the solution. This can be easily demonstrated by asking you to multiply numbers where the operands are represented as Roman Numerals, for example, multiply XXIX by IX. You probably found yourself translating; it is a challenging task in the "wrong" representation (Rich, 1985). The impact language has on problem solving is more profound than this simple demonstration and in fact accounts for much of the increase in productivity and ease of use of 4GLs - the solution statement is closer to our natural representation of the real world problem.

Language Shapes the Very Way We Think

The second way that language is important in problem solving is more subtle, but perhaps even more important. According to the Whorf-Sapir Hypothesis, "the background linguistic system of each language, is not merely a reproducing instrument for voicing ideas, but rather is itself the shaper of ideas..."(Whorf, 1956). 4GLs provide the programmer with a richer texture of language from which to choose, including procedural, non-procedural, and visual programming. The student who learns to program using multiple language modes may in fact have multiple problem solving modes.

Leveraging Students' Time

A third reason for using 4GLs in those CIS courses which do not teach syntax as their primary focus (systems analysis and design, or a programming project) is that the use of 4GLs for implementation of student assignments can leverage the student's time. In the same sense that SPSS or SAS free the student from detail computational tedium and allow a student learning statistics to perform "real" statistical analysis, the use of 4GLs within the curriculum can free CIS students from the tedium of detail programming and allow them to focus on learning to solve "real" problems and develop systems.

DIFFICULTIES IN TEACHING 4GLS

Even if you find merit in the proposition of offering a 4GL course within your curriculum, there are difficulties to overcome. The key problems are: 1) placement within the curriculum, 2) software, and 3) texts.

Placement within the Curriculum

Placing the 4GL course properly within the curriculum will depend upon the following factors: 1) the goals of your current curriculum, 2) faculty interest, 3) flexibility within your existing curriculum and 4) desire to adhere to curriculum models.

The focus can range across a continuum from end-user oriented through a survey of 4GL concepts to programming in 4GLs after the model of an advanced COBOL course. Figure 1. depicts a continuum of course orientations. Appendix A represents a course outline for the CIS major focused toward actual programming in a 4GL while Appendix B represents a course oriented toward the end-user. The two end-points seem to offer more pragmatic courses for the CIS department to offer as a major and service course respectively. A course surveying 4GL concepts with focus on language requirements and constructs (similar to or an extension of the traditional survey of languages course) may emerge within Computer Science programs.

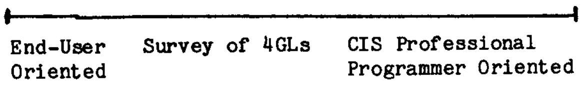


Figure 1. A Continuum of 4GL Courses

The 4GL course could simply be offered as a special topics course, or perhaps as a formally defined elective course. This option has the benefit of disturbing the current structure of the curriculum as little as possible; however, it does not maximize the benefits of integrating 4GL within the curriculum.

Another option would be to integrate 4GLs throughout the curriculum. In fact, regardless of whatever other options you might take, 4GLs should be treated throughout the curriculum. Given that you had the required software, it would be possible to embed 4GL concepts in courses such as Introduction to Programming, COBOL, Systems Analysis and Design, Software Projects, and perhaps others. The major drawback to this approach is that coverage would be uneven and non-standard, and there is no text material to support integration (for example a COBOL text with an assignment that includes embedded SQL). The Database course is absent from this list of courses where 4GL concepts might be embedded because 4GL concepts should necessarily be integrated in the Database course. However, coverage of some aspects of 4GLs within the DB course will not suffice for coverage within the curriculum because the focus of the DB course ought to be the logical design, representation and management of data rather than the manipulation of the data.

A third alternative is to formalize the course and make it a required part of the curriculum. There would be two possible scenarios for this option, 1) create an entirely new course and 2) merge 4GL concepts into an existing course. Given the intensity of most CIS programs, few curriculum managers have the luxury of simply increasing the number of units in the major. Creating a new course usually means forcing an existing course out of the curriculum. Merging 4GLs with an existing course leaves few choices within typical CIS curricula; the most likely is the Decision Support Systems course. Both of these ideas require elaboration.

About the only course that is a logical candidate for outright replacement is the one which 4GLs may functionally replace, advanced COBOL. Most employers who expect a new college graduate to become a COBOL programmer recognize that learning COBOL at the university has taught the student the basics of problem solving with the language and demonstrates the student's trainabililty. They still must teach the student their organization's way of COBOL programming. Given the move by most large organizations to 4GLs, a very fast-paced Introductory COBOL course at the university could leave the student able to perform elementary maintenance and ready for the organization to train. A graduate of a CIS program in the 1990s would be more employable with regard to an entry level business programming position if he had learned Introductory COBOL and 4GL than if he had learned Introductory and Advanced COBOL.

The difficulty in embedding 4GLs in the DSS course is that the traditional, very quantitatively oriented DSS course is already being required to accept new material in the area of qualitative decision making from the emerging discipline of expert systems. If 4GLs are merged with DSS, some component of quantitative or expert systems DSS coverage will have to be compromised.

A fourth curriculum decision which is not a mutually exclusive option (in fact it might be a way to amortize the cost of software) is to offer a 4GL service course to functional disciplines other than CIS majors. Such a course would be well positioned at the upper division or graduate level of the business program.

Software for the Course

Software for the course presents two difficulties: selection of the software package and price. There is no standard for 4GLs. In some ways this is good, and in some ways bad. It would be incorrect to prematurely standardize 4GLs and cap off innovation, yet the lack of standards complicates the decision on what 4GL to teach. In contrast to selecting software to deliver a third generation business programming language course, there is no equivalent to COBOL with regard to being the industry standard. The first decision regarding software is driven by the focus of the course, specifically whether it is oriented toward end-users or DP/CIS professionals as the applications developers. Having made the decision regarding the focus of the course, if money were no issue, the question of which package could be resolved in one of two ways: 1) teach the package with the largest market share, or 2) teach a package based on the emerging standard for the underlying relational data base, SQL. Unfortunately, these are mutually exclusive today.

The industry leader in terms of installed systems is FOCUS (Schatz, 1984); however, FOCUS is not based upon SQL. There are many 4GLs which incorporate SQL as the data base language, but SQL implementations vary with regard to extensions such as graphics, statistical packages, and other modeling tools. Furthermore, the data base language is only a portion of a 4GL, albeit an important portion.

The next problem is matching choice of software with the CIS budget. If selecting the right software were not difficult enough, many of the packages have demanding hardware requirements. Although the most popular mainframe 4GLs have been brought to a PC environment, many require an IBM PC-370. Virtually all require at least a well configured PC including a hard disk or a network file server. Many of higher education's PC labs are not so well equipped. Although some of the systems will run on minicomputers popular with CIS programs, they are resource intensive and the software license is expensive. What is needed is a constrained version of a popular 4GL like FOCUS, Oracle, or Informix-4GL which contains all of the features without all of the capacity of the production system.

Texts for the Course

At the present time, there are few textbooks to support a 4GL course (Couger, 1985). James Martin has certainly written the most on the topic. While the industry's most successful seer has treated 4GLs in at least five titles, the treatment is always oriented to the DP/CIS manager, not to the CIS student. The first Martin book to consider is Fourth Generation Languages, Volume I: Principles. It is current and extremely valuable for background and a managerial perspective. It provides guidelines for managing the development effort with 4GLs and the use of accompanying tools such as Action Diagrams. It demonstrates some code. It is, however, not a programming textbook in the vein of our common Advanced COBOL books. Volumes II and III also provide excellent reference sources. Volume II treats non-IBM packages while Volume III is devoted to IBM 4GL packages.

Although becoming dated, Applications Development without Programmers is an effective book to present the overview of end-user programming in an upper division or graduate level service course for functional end-users. It is certainly worthwhile reading for CIS majors, but it is not a text which can support delivery of 4GLs within the CIS curriculum in any more than an adjunct role. Other Martin books that deal with the topic are An Information Systems Manifesto, Action Diagrams and Software Maintenance: The Problem and its Solutions (the latter two are co-authored with C. McClure).

There are several texts which address some aspect of 4GLs and, therefore, might serve as supplementary texts. Unfortunately, none of them can serve as a stand-alone text to support a CIS course on 4GLs. These texts are listed in Appendix C.

CONCLUSION

My recommendation, given the importance of 4GLs, is that the course be taught in two modes:
1) as a stand-alone, required course in the traditional CIS curriculum and 2) as a service course aimed at the upper division Business major. If the additional units of a required 4GL course cannot be tolerated within your curriculum, the Advanced COBOL course can become an elective course. Software for the CIS course should be a production programming oriented 4GL such as INFORMIX-4GL and it should have a SQL base. The software for the service course should be an end-user oriented package such as Oracle or FOCUS. I have chosen Oracle because of its underlying SQL base. With regard to texts for the course, you will have to improvise and supplement from several sources. Depending upon your software choice, I would expect that you will find acceptable texts to support the course within the next few years.

BIBLIOGRAPHY

Blaisdell, James H., "4GL Working Papers," Humboldt State University, 1986.

Bozman, Jean, "Bank Saves With 4th-Generation DB System," Information Systems News, 1984.

Couger, J. Daniel, "Annual Bibliography of Computer-Oriented Books," Computing Newsletter for Schools of Business, January, 1985.

Frand, Jason L. and Ephraim R. Mclean, "Summary of the Second Annual UCLA Survey of Business School Computer Usage," Communications of the ACM, January, 1986.

Jones, T. Capers, "The Limits to Programming Productivity," Proceedings of the Guide and Share Application Development Symposium, 1979

Martin, James, An Information Systems Manifesto, Prentice-Hall, 1984.

Martin, James, Fourth Generation Languages: Volume I, Principles., Prentice-Hall, 1985.

Martin, James and C. McClure, Action Diagrams: Clearly Structured Program Design, Prentice-Hall, 1985.

Martin, James and C. McClure, Software Maintenance: The Problem and its Solutions, Prentice-Hall, 1983.

Rich, Robert, "Influence of Language on Its User," in The Role of Language in Problem Solving - 1, R. Jernigan, B. W. Hamill, and D. M. Weintraukb eds. North Holland, 1985.

Schatz, Willie, "Can RAMIS II Deliver," Datamation, July 15, 1985.

Whorf, Benjamin Lee, Language, Thought, and Reality, selected writings edited by John B. Carroll, MIT Press, 1956.

APPENDIX A
PROGRAMMING WITH FOURTH GENERATION LANGUAGES AN ABBREVIATED COURSE OUTLINE FOR A 4GL COURSE ORIENTED TO THE CIS MAJOR

PART I. OVERVIEW OF FOURTH GENERATION LANGUAGES

1. INTRODUCTION - THE SOFTWARE CRISIS AND LANGUAGE GENERATIONS THE FOURTH GENERATION AND THE GOAL OF INCREASED PRODUCTIVITY
2. CONCEPTS OF PROGRAMMING IN ANY LANGUAGE
3. FOURTH GENERATION PROGRAMMING - WRITING REPORTS
4. UNDERLYING THEORY - RELATIONAL SYSTEMS
5. DATABASE PROGRAMMING IN SQL - APPLICATION OF RELATIONAL THEORY

PART II. DEVELOPING SYSTEMS WITH FOURTH GENERATION LANGUAGES

6. DEVELOPMENT ENVIRONMENTS AND DEVELOPMENT TOOLS
7. END-USER PROGRAMMING
8. DEVELOPING LARGE SYSTEMS

PART III. IMPLEMENTING FOURTH GENERATION LANGUAGES

10. SURVEY OF COMMERCIAL SYSTEMS
11. MANAGING THE 4GL ENVIRONMENT
12. A GLIMPSE OF THE FUTURE

APPENDIX B
APPLICATIONS SOFTWARE WITHOUT TRADITIONAL PROGRAMMING AN ABBREVIATED COURSE OUTLINE FOR A 4GL COURSE ORIENTED TO THE BUSINESS MAJOR

1. THE SOFTWARE DEVELOPMENT CRISIS
2. SOFTWARE DEVELOPMENT WITHOUT TRADITIONAL PROGRAMMING - END-USER SOFTWARE DEVELOPMENT LANGUAGES
3. CASE STUDY OF AN INSTALLATION
4. CHANGES IN DP MANAGEMENT - NEW ROLES FOR DP PROFESSIONALS
5. "CANNED" SOFTWARE APPLICATIONS
6. THE CENTRALITY OF DATABASES - INTRODUCTION TO SQL
7. APPLICATIONS GENERATORS
8. INTRODUCTION TO ORACLE
9. SURVEY OF 4GLs
10. THE INFORMATION CENTER

APPENDIX C
TEXTS WHICH PROVIDE SUPPLEMENTARY SUPPORT FOR A 4GL COURSE

Benne, Bart, The Illustrated PC-FOCUS, Wordware Publishing, 1985.

Date, C.J., A Guide to DB2, Addison-Wesley, 1984.

Katzan, Harry, Jr., Invitation to Mapper(I), Petrocelli, 1983.

Martin, James, Application Development Without Programmers, Prentice-Hall, 1982.

Martin, James, An Information Systems Manifesto, Prentice-Hall, 1984.

Martin, James, Fourth Generation Languages: Volume I, Principles., Prentice-Hall, 1985.

Martin, James, Fourth Generation Languages: Volume II, Representative 4GLs., Prentice-Hall, 1986.

Martin, James, Fourth Generation Languages: Volume III, 4GLs from IBM., Prentice-Hall, 1986.

Martin, James and C. McClure, Action Diagrams: Clearly Structured Program Design, Prentice-Hall, 1985.

Martin, James and C. McClure, Software Maintenance: The Problem and its Solutions, Prentice-Hall, 1983.

McCracken, Daniel D., A Guide to NOMAD, Addison-Wesley, 1980.

Stonebraker, Michael, The INGRES Papers, Addison-Wesley, 1986.

Towner, Larry E., The ADS/ONLINE Cookbook, MacMillan, 1986

APPENDIX D
ANNOTATED LIST OF SELECTED 4GL SOFTWARE

The following list of software vendors of 4GLs is not intended to be all inclusive. There are literally hundreds of 4GL products. Although many colleges have access to a super-mini capable of running 4GL packages, the limitations of the quality of terminals coupled with the expense and degradation in performance of the super-mini argues for delivering the course on PCs. Since the majority of colleges of business have IBM-PCs (Frand, 1986), the list includes those popular systems which will run on a PC. Some require a hard disk or network file server.

FOCUS is end-user oriented and retails at $1295 for the PC and requires a hard disk and 512KB. IBI, the vendor offers an educational discount. It is the market leader. It is more attuned to the End-User. A major drawback in teaching prospective CIS professionals is that it is not SQL based.

Oracle is end-user oriented and retails at $995 for the PC. It requires a hard disk and 512KB. Oracle Corp. offers a 40% educational discount. They will also let you try it for thirty days on the basis of a Purchase Order.

INFORMIX-4GL is oriented to the CIS professional. It retails for $995 and RDS offers a 75% discount for educational institutions. In order to effectively teach prototyping concepts, it is necessary to purchase the ISQL package also ($795). 4GL requires a hard disk and 512KB. ISQL will run on a dual floppy PC with (some uncomfortable disk swapping). Demonstration versions of both are available.

COMPUTER-AIDED SIMULATION AND TELE-OPERATION OF A ROBOT ARM

Nishith Bhushan and Aminul Karim
Department Of Electrical Engineering
Colorado State University
Fort Collins, CO 80523.

ABSTRACT

In this paper computer-aided graphical simulation and tele-operation of a robot-arm are considered. The hardware consists of a PS300 graphics system (Evans & Sutherland), an IBM PC-AT and a SIR1 robot arm, interlinked via bi-directional communication and controller interfaces. Simulations of the robotic system are conducted on the PS300. The IBM PC-AT, equipped with a Data Acquisition and Control Adapter Board, acts in conjunction with a hardware controller as a sophisticated controller for the robot arm.

A wide variety of robot-control algorithms can be designed and implemented on the IBM PC. The performance of the robot can be observed (monitored) through a computer generated image on the PS300 display monitor. Furthermore, the existence of many devices on the PS300 enables the realization of a highly flexible method of man-machine interaction between the robot via the interactive controls of the PS300 graphics system.

An additional linkage between the PS300 and a DEC VAX-11/780 gives the possibility of implementing AI algorithms for adaptive learning of the dynamic behaviour of the physical robot. Initial results are presented, although the emphasis is on the methodology.

I. INTRODUCTION

Graphical Simulation is considerably important in the present state-of-the-art in Robotics and various CAD/CAE related applications (3). Such a simulation (graphical context everywhere in this paper) is more powerful when it is linked to an actual robotic system. The robot can be controlled from the graphics device or the graphics display can serve to monitor the work of a robot, especially in a hazardous environment from which visual information cannot be extracted through the regular means (TV cameras, etc.).

A methodology to achieve this is presented in this paper where the control theory or the kinematics of the robotic system are not the focus. Primary consideration is given to the software needs with the aim of focussing on the hardware interfacing problems in the future. For the kinematics look at (5).

There are three phases in the design of the overall system, and each leads to the next. They are :

(1) Simulation of the robot arm.

(2) Learning by motion.

(3) Tele-Operation or the learned stage.

II. GRAPHICAL SIMULATION ON THE PS300

The PS300 graphics system is based on a data-flow architecture and has 52 interactive input devices. The simulation program in PS300 graphics command language is resident on a VAX 11/780 which serves as the host computer for the PS300. The PS300's display processor takes over once a graphics file has been sent from the VAX. The display can be interactively controlled through a provision of programmable dials and function keys.

A SIR-1 robot arm is simulated on the PS300 with all its 6-degrees of freedom (dof) incorporated in the interactive nodes of the display file. The workspace of the robot arm as well as the limits of each dof are defined in the display file. For instance, lower and upper limits for rotation of an end-effector can be set to the prescribed value taken from the actual robot arm. Figure-1 shows a skeletal display tree used for simulating the SIR-1 robot arm.

The animation capability of the PS300 allows a dynamic display which makes possible concurrent demonstrations or jobs to be assigned to the actual and the simulated robot arm. A sequence of motions are attained by interactively updating the dynamic nodes of the display file for rotation and translation. This updating can be done manually or through clocked function

FIGURE : DISPLAY TREE FOR SIR-1 ROBOT-ARM

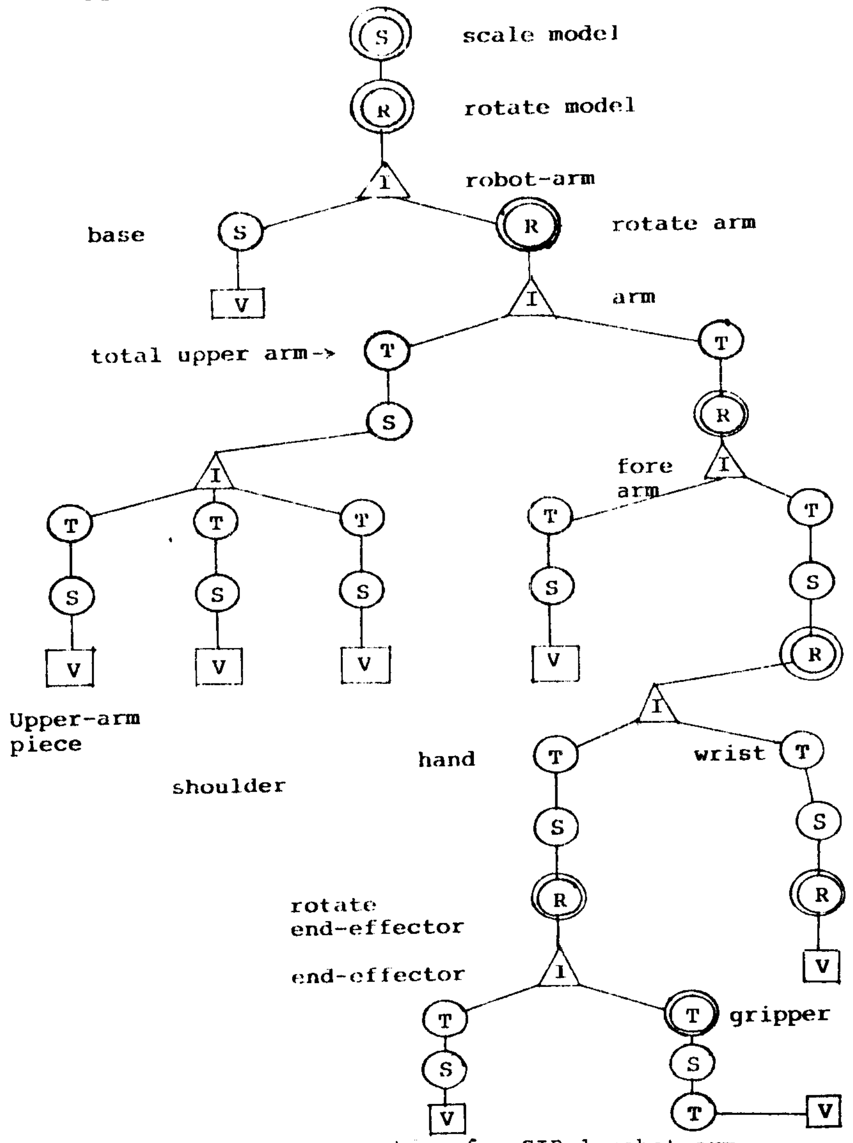


Figure-1 . Display tree for SIR-1 robot arm.

Various nodes explanations are : T : Translation; S : Scaling; R : Rotation; I : Instance of; V : Vector list;

Single circles	:	modelling transformations on data structures.
double circles	:	dynamic nodes (can be updated).
triangles	:	used for combining two data structures.

networks.

A workplace is simulated and obstructions are placed in accordance with those in the actual workplace. The simulation feels the obstructions and halts any motion as it is programmed to do so on intersection of the robot's boundaries with the simulated objects. A better method would be to provide the actual robot with sensors which send information to both the controllers : for the robot and for the simulation (2).

III. LEARNING BY MOTION

In the first stage these networks are upgraded with the spatial motion values of the actual robot. The data from the software controller (in BASIC) used to drive the robot is sent to the VAX 11/780 and is translated to spatial data. That is, the robot-control input is changed appropriately for application as the control input for the simulation. The translation program on the VAX attains an isomorphic mapping between the controls for the robot and for the display so that the displayed image in terms of workplace and dof limits. The mapping is done by isomorphic blocks for each dof and an internal isomorphism for singular motion mappings.

The mapping (in LISP) provides data that is fed to the simulation program for updating the dynamic nodes. At present, real time simulation is not conceivable as the new data values are stored in a file for later updating (owing to the nature of UNIX based systems). The nodes are updated with one value each time A continuous stream of data needs to be available to the PS300 display file for real time processing.

IV. TELE-OPERATION.

In essence this is a learned state. The learning incorporated from the mapping is used to operate the robot arm from the PS300 display dials. The dial outputs can add dynamic control to not only the simulation but also the actual robot arm. These outputs are sent to the host computer for translation from spatial data to signals for the robot controller. The software controller on the IBM PC is replaced by the graphics controller. The schematics are shown in figure-2.

The accuracy of the forward and reverse isomorphism that relates two data types determines the accuracy in the transfer of controls from the numeric/ alpha keyboard to the PS300 dials. The bi-directional communication interface between the robot arm and the simulation allows the use of various learning cycles to define tasks for the robot arm. It is more natural to control the robot through tele-operation as it provides for the dexterity of human control.

V. CONCLUSIONS.

Tele-operation is mostly used for operation in environments hostile to man and TV is the most common viewing medium (4). Graphical simulation is required in cases where visual data extraction from the environment is not possible. Our methodology fits in those situations.

Although only a robot arnm is used, extensions can be made to incorporate a full robot. Instead of a manual tele-operator the process can be automated whereby the human factor is placed in the control loop, which is often desirable. And going in path of other researchers in this area (1), a major step would be to incorporate intelligence in the automated tele-operator so it can adaptively learn from its is used environment and make the neccessary adjustments - quite like the manner in which humans adaptively change behaviour and tasks based on changes in their environment (4).

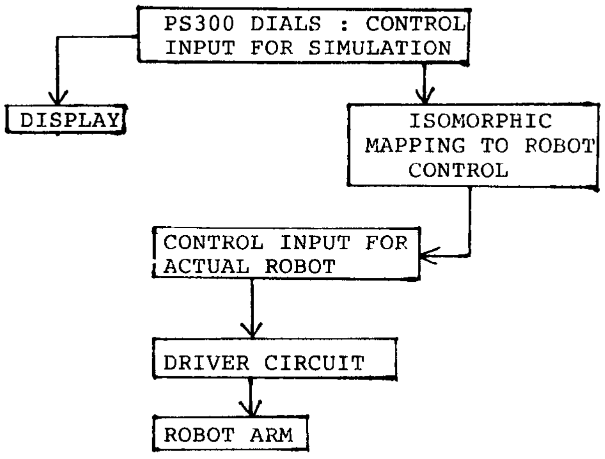

PS300 DIALS : CONTROL INPUT FOR SIMULATION
DISPLAY
ISOMORPHIC MAPPING TO ROBOT CONTROL
CONTROL INPUT FOR ACTUAL ROBOT
DRIVER CIRCUIT
ROBOT ARM

Figure - 2 : Tele-Operation

REFERENCES

1. Badler, Norman I., B.L.Webber, J.U. orein, and Jonathan Korein : " TEMPUS : A System for the Design and Simulation of Mobile Agents in a Workstation & Task Environment," 1983 Trends and Applications : Automating Intelligent behaviour - Applications and Frontiers. 1983, pp 263-269.

2. Bajcsy, Ruzena : " Integrating Vision and Touch for Robotics Applications," ibid. (1), pp 193-7.

3. Cohen, A. and J.Erikson : " Future Uses of Machine Intelligence for the Space Station and Implications for the US Economy," IEEE J. of Robotics and Automation, vol. RA-1, No. 3, Sept. 1985, pp 117-123.

4. Johnsen, E.G. and W.R. Corliss, Human Factor in Teleoperator Design and Operation, Wiley, New york, 1971.

ASSESSING THE FOURTH-GENERATION LANGUAGE AS A PRODUCTIVITY TOOL: AN EXAMPLE OF USE FOR THE BUDGETING SYSTEM FOR THE IDAHO LEGISLATIVE OFFICE

Wita Wojtkowski, Boise State University, Boise, Idaho 83725, (208)385-1372
W. Gregory Wojtkowski, Boise State University, Boise, Idaho 83725, (208)385-1227

ABSTRACT

Use of the Fourth Generation Language for the HP/3000 is presented, productivity gain and improvement of design technique through its use is discussed.

INTRODUCTION

It appears that there are two major problems with the way information systems are developed: the process of the development is too long and right systems are never developed first time around.

For the last two decades information systems researchers and practitioners have been trying to develop tools which will help to alleviate those problems. The tools include programming techniques based on structured methodology, code generators, query languages, and, in recent years, prototyping and Fourth Generation Languages (4GLs) [2] [3].

The 4GL originated with the business sector in 1968 but the introduction process was slow [4]. During the last five years increased interest in this class of languages can be observed. Researchers in academia are showing greater interest moreover, there is some pressure to introduce them into the curricula of information science and computer science departments of educational institutions [1]. With this increased interest we will see improvements and better acceptance of the 4GLs in the business sector as well as in academia.

The object of this paper is to present an example of system development with 4GL. We point out productivity gains and some of the pitfalls.

PRINCIPLES UNDERLYING THE IMPLEMENTATION OF SYSTEMS USING 4GL

The typical users of computing in the corporate environment usually understand which mathematical manipulations should be performed by the computer program and they also know where they want the results to appear in the output. The problem they face is to determine the proper sequencing of their logic. Similar problems are faced by students in programming classes. They can learn the syntax for the various types of statements, but they often have great difficulty in learning the proper logical sequencing especially if the problem they are solving is for a real-life application.

The 4GLs offer the tool for effective and efficient implementation of real-life computer systems. Users are able to concentrate on solving their particular problem and not to worry about the logic requirements of the specific language [4]. Very important too, systems developed with 4GLs are easy to maintain [5].

Good 4GL will have the dictionary system which will support different data base management systems (DBMS) and file types (in some 4GLs the DBMS is the integral part of the language). It will also have an extensive screen design and reporting facility. It should allow to call upon subroutines written in some other procedural languages . In [4] on page 36, there is an extensive list identifying properties of languages that can be considered as 4GL type.

With the 4GL which allows for fast creation and modification of systems prototypes one should be able to substantially shorten the system analysis and design stage. Moreover, prototyping is the most appropriate technique for the design of systems with the use of 4GL [5]. The users play more active roles in the system development utilizing 4GLs than is possible with traditional development methods. In effect, users become system designers and evaluators. Certainly, some of the features of an application will have been initiated by the builder rather than the user. For example, technical features, performance and integration of the system with the existing databases and processes should be the builder's concern, not the user's.

THE APPLICATION

In order for the computerized budgeting system in any environment to be effective it must offer features usable by personnel with varying levels of computer expertise and skill. The system we are presenting here will be highly integrated and on-line.

1. the budgeting data base which can handle the lowest level of detail and will provide for maximum output flexibility;

2. reporting system that supports the actual work performed by the budget analysts;
3. electronic preparation of agency budget requests;
4. revenue and expenditure projection models;
5. control and tracking of the allocation of appropriations; and
6. integrated voting system for the House and Senate of the Idaho Legislature.

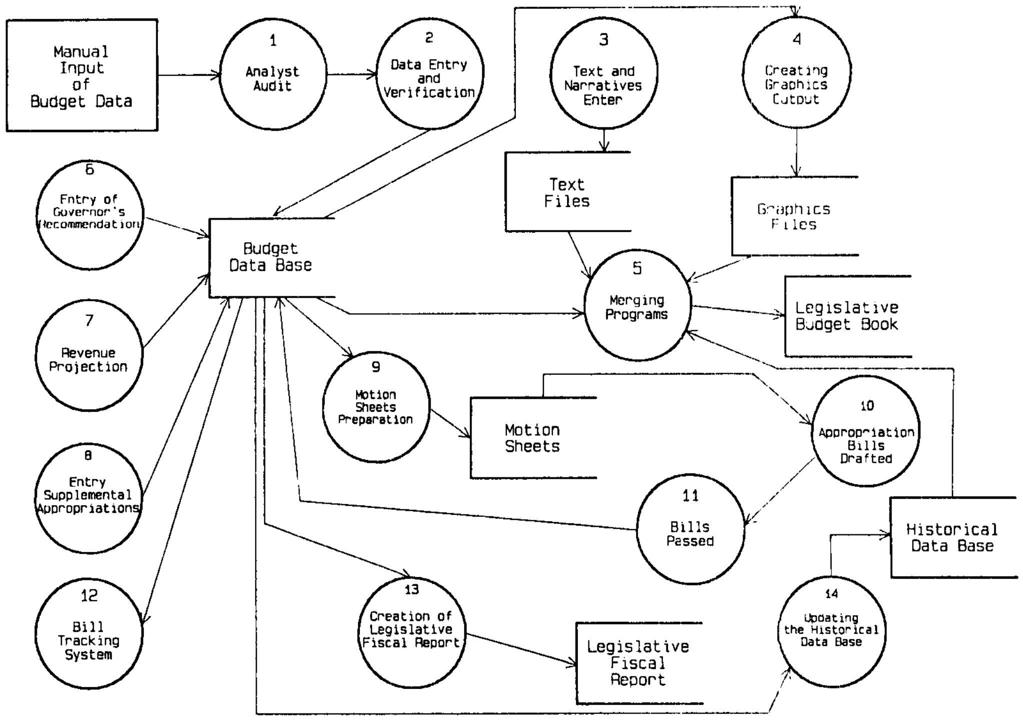
The design for this system is modular so the individual functions are being implemented gradually over time.

The general data flow diagram of the computerized budgeting system for the Idaho Legislative Budget office is shown on Figure 1.

THE SYSTEM DEVELOPMENT TIME--REAL PRODUCTIVITY GAIN

Fourth generation systems are typified by high-level, user-driven system generators. The user is provided with a means of entering parameters, and then this information is used in the automated construction of a new system. This results in substantial change to the system development life-cycle. With an iterative development approach, that is prototyping, much less attention is given to prior requirements definition and investigation phases of the project efforts. Compressed development time is achieved. This shortened development time is a true productivity gain.

Development of the Idaho Legislative Budgeting System is done by the students in



Manual Input of Budget Data
1 Analyst Audit
2 Data Entry and Verification
3 Text and Narratives Enter
4 Creating Graphics Cutout
Text Files
Graphics Files
6 Entry of Governor's Recommendation
Budget Data Base
5 Merging Programs
Legislative Budget Book
7 Revenue Projection
9 Motion Sheets Preparation
Motion Sheets
10 Appropriation Bills Drafted
8 Entry Supplemental Appropriations
11 Bills Passed
Historical Data Base
12 Bill Tracking System
13 Creation of Legislative Fiscal Report
Legislative Fiscal Report
14 Updating the Historical Data Base

Fig 1. Budgeting System Data Flow

the CIS430, Software Design course. The system is programmed in the 4GL for HP/3000 called POWERHOUSE. During the first five weeks of the semester students are introduced to this specific 4GL. (They do not need to be introduced to the design of the budgeting system since they originally designed it in their Systems Analysis and Design course in the previous semester.)

The next six weeks of the semester are devoted to coding and the rest of the four weeks to testing and documenting. Classes are divided into teams of four students each. Each week students select a project leader

The design and implementation of the system is done by students in the CIS program at the Boise State University School of Business. Such a venture is possible because of the application of the user-oriented design strategy and it is also due to a decision to develop the system in the 4GL.

It is hoped that the successful development of the budgeting system for the the Legislative Branch will provide a sound foundation for the future development of the automated Statewide system.

and the work is divided among the members of the team. Teams are delivering written code to the instructor for approval and acceptance. If the code is not working properly students are responsible for changes until the code is acceptable.

Successful development of the budgeting system by the inexperienced programmers (students) and in such a short period of time is possible because of high productivity achieved through use of 4GL. Moreover, this 4GL is easy to teach and to learn. Improvement of the design technique is done through prototyping. System failure does not have the same connotation as it does in the traditional models. In fact, we find a failure desirable for troubleshooting and for learning how users cope and react.

CONCLUSIONS

With 4GLs which offer productivity gains it is possible to build, with students in CIS, quite complex real-life systems. More importantly, this gives students the experience of solving problems that cannot be found in any book.

Use of 4GL offers an extremely important opportunity for concentrating on teaching problem-solving techniques, not on teaching syntaxes of languages only. However, problems arise when one is attempting to carry second and third generation languages "mentality" into fourth generation application. Often the decision to use a specific tool for the ENTIRE application is made, when perhaps some portions of the system would work better with an alternative software language or tool. Thus, available tools are often used for the wrong purpose [6]. 4GL chosen to bring about productivity gain in the applications development phase is then expected to out-perform traditional languages. This is impossible.

More research should be done concerning comparison of performance of 4GLs and 3GLs. Through this some guidelines for the most effective use could be established. Perhaps, more academic researchers should be working on improvements to 4GLs as they are presently working to improve 3GLs.

The authors are convinced that the next few years will bring about wide application of 4GL's and their unavoidable introduction into the CIS and the computer science curricula. Through this paper we present an example of such an introduction.

REFERENCES

1. ACM 85 Panel Session, "The Role of Fourth Generation Languages in Computer Science Education," Denver, Colorado, 1985.
2. King, W.R., "Alternative Designs in Information System Development," MIS Quarterly, Vol. 6, No. 4, December, 1982, pp. 31-42.
3. Lucas, H.C., Jr., "The Analysis, Design, and Implementation of Information Systems," second edition, New York, McGraw-Hill, 1981.
4. Martin, J., "Fourth-Generation Languages," Vol. 1, (Principles), Prentice-Hall, Inc., 1985.
5. Naumann, J.D., and Jenkins, M.A., "Prototyping: The New Paradigm for Systems Development," MIS Quarterly, Vol. 6, September, 1982, pp. 28-44.
6. Young, T.R., "Superior Prototyping," Datamation, May 15, 1984, pp. 152-158.

THE CASE FOR TECHNICAL WRITING

Ellen Mahon, CDP
Iona College

The Technical Writing course, when tailored to the CIS/MIS major or minor, can serve as a natural "lab" in Systems Analysis since the documents studied correspond to phases in the system development life cycle. Assigned to project teams, students learn to listen to one another in preparing a class project. They also draw upon the group experience to write individual reports and memos. By complementing software engineering with such a writing course, key system development concepts are reinforced in a practicum which teaches the communications skills critical for the information worker.

Having been a DP Manager before returning to college teaching, I know only too well the "We'll document it later" syndrome. Therefore I heartily welcome the increasing emphasis on including a Technical Writing course in the academic curriculum for CIS majors. Though my institution, like many others, offers this course through the English Department, the reality in business is that the job of documentation most often falls to the systems analyst. Without arguing the curriculum development question, however, I will show in this paper how using a case study approach in Technical Writing reinforces the Systems Analysis course while developing analytical and "people" skills as well as the ability to write an unambiguous document. Hopefully I will also persuade at least one CIS professor that an English teacher can be trusted.

As a discipline younger even than CIS, "Technical Writing" communicates little about course content. Even when subtitled "Computer Documentation Skills," or otherwise restricted to computer technology, a course in Technical Writing may cover merely the preparation of manuals or it may extensively overlap the Business Writing course in its attention to letters, memos and reports. Textbooks on writing specifically for the computer industry are precious few, so further enlightenment cannot be sought from that quarter. How, then, to structure a meaningful course?

"Know your audience," counsels every book about writing. I see my audience as the prospective systems analyst, the student majoring in computer science who has learned (well, we hope) the importance of logic and the predictability of a machine that follows rules, but who is yet innocent of the technological illiteracy and powerful sway of "Users." Among the skills this student needs are certainly the skills of communication: the ability to listen--to hear the problems of others because others, not he, will ultimately live with the system--and the ability to speak in a way that others will understand. What better way to throw this budding MIS professional into the swim than by simulating a project development team?

So I structure the Technical Writing course according to the system development life cycle. Professor F.L. Bauer of the University of Munich summed up this familiar cycle in a truth-telling metaphor at the 1979 Software Engineering Conference in Albany: the egg, the caterpillar, the cocoon, the butterfly. Since there are always students in the class who have already taken Systems Analysis, eliciting the more conventional nomenclature comes easily. Documents are studied in chronological order, fitted to the appropriate development phase: the proposal, feasibility study and functional spec during the analysis cycle followed by design specs and ultimately manuals. Along the way memos, minutes and status reports are worked in as a record of "project team meetings."

It has been my good fortune to teach Technical Writing at night in an environment of working adults mixed with full-time students from the day division. Consequently, the projects or case studies have come from real life, adding a welcome immediacy and practicality. The course is open to CIS majors and minors and "with permission of the instructor" so an occasional computer neophyte adds the requisite technical naivete--and prevents the group from committing jargon.

To see just how the project team approach works to structure the course, let us focus on a particular class: Spring, 1986. At the first class meeting, students filled out a profile sheet so I could assess and match up their backgrounds and interests. This group was predominantly concerned with manuals and procedures, though the previous class leaned heavily toward project proposals. Of the thirteen, nine were CIS majors or minors (two non-computer students subsequently dropped out), eight had already taken Systems Analysis, seven had no conception of project team experience, and five were actively engaged in documentation at their job. Three of these five actual documenters sounded genuinely burdened, so I proposed that they function as project leaders for the class.

Early writing assignments in the course concentrate on fundamentals and the difficulty of defining such taken-for-granted terms as the verbs "boot" or "format"; meantime class lectures establish a common frame of reference. During this start-up period, each of the three prospective project leaders presented a system overview and then explained the writing task at hand. A telecommunications project more concerned with installation diagrams than written words was eliminated on this round, and we settled on two viable systems: a mainframe hospital admissions system and a microcomputer-based order processing/invoicing system. Class members gave me their project choice in writing, but I reserved the right to assign groups, and so insure a mix of experience on each team.

By the third week of class, therefore, project teams were formed and all students had a specific case to relate to as each document was studied. Technical Writing builds on Systems Analysis. In the Technical Writing course, it should be noted, the emphasis is on the purpose of each document and what it should contain to achieve that purpose: the proposal is a bid to solve a problem and it must persuade; the feasibility study clearly lays out alternative solutions and their comparative costs in order to recommend the best choice in the given circumstances; a functional spec serves as a contract between user and developer so it must record agreed upon functions of the proposed system. But no one can write what he does not understand, least of all a "technical" writer. So the simulation of project development affords specific content while providing practice in communicating.

With the skeletal structure in place, we can think about the assignments which flesh out the course. Because good writing requires constant practice, a two-page paper is due every week. In these short papers, students assimilate content: what belongs in one kind of document as opposed to another. Each student must also submit an individual project, allowing scope to amplify one of the documents surveyed. Finally the group project approximates the length of a real life spec or manual and is put together by many though it must seem the work of one.

It works out that about every fourth class hour is given over to project team meetings, though as the semester progresses students realize the need to call meetings on their own time. The first two-page paper that relates to the project falls due after studying the proposal. Together the students work out a formulation of the problem their system is supposed to solve; they also identify the parameters of the solution. Then each separately writes up the problem statement/proposed solution. Before handing in these embryonic proposals, they exchange them in the group to evaluate writing and also to clear up misunderstandings.

For the proposal, then, two project team meetings were held--one long, one short. At each, a recorder was assigned to take minutes and write them up for the group. For a change of pace I had the students search out actual examples of feasibility studies and analyze the different styles brought in; however, this assignment, too, could have been project-related. Like the proposal, the functional spec involved a problem-solving meeting in preparation. This time the task was harder, because the spec is a much longer, more complex document. But the team members now knew both each other and the project better; they worked together to distill the essence of "the contract" for the short weekly assignment, relegating details to appendices and charts.

Interesting things began to happen as each project team "jelled." The mainframe group, working on hospital admissions, sailed along quite smoothly; they had a strong leader, and they had defined the system problem clearly. It should be confessed, also, that each member of this group had gotten the project s(he) chose. But the micro group kept straying off the topic into technology! Sound like real life? You bet. To deal with their difficulties this group was forced to exchange phone numbers early on and to try to resolve project ambiguities in one on one conferences.

By mid-semester, team meetings began to focus on the actual group project. Design specs were presented and analyzed in the lectures, but written only in parts except for the programmer who had chosen a program spec as his individual project; in a class that was not so manual-oriented the emphasis would have been different. The balance between individual and group work kept frustration to a minimum, and no student was "left out" through ignorance of system development.

Ultimately, each group had to report to the whole class on its experience: strengths, obstacles, insights, final product. I will discuss elsewhere my findings and how I put them to use in a further iteration of this teaching approach! However, the collective reflection on the project team experience seems as necessary to real learning as the process itself. And only from the perspective of the "completed" system can a review of its major documents, their purpose and content, be really understood.

At this point a computer science teacher may well be wondering just what the Technical Writing course offers that the Systems Analysis course does not. I have refrained from dwelling on "English" concerns because that is not the purpose of this paper. Nevertheless, a small example may prove helpful. When starting to teach the functional spec, I begin by distributing a sample proposal to review that document. I then present a hierarchy chart for the same proposed system. I ask the students what the chart is and what it tells; and gradually we work to the realization that the hierarchy chart does identify functions but it says nothing about who or when--responsibility is ignored. We then look at a flowchart and consider how this abstract representation also has the virtue of organizing our understanding of the system while leaving us unclear about important details. From this point it is an easy step to the place of unambiguous writing in the total system development picture.

What benefits result? (Admittedly I have business and not pure science primarily in mind.) Not only will the CI--that stands for information--S graduate know structured programming, compilers and operating systems, file and database considerations, software engineering and project management; he or she will also know how to write this knowledge down for the lay person to understand. He or she will have confidence about where to say what and why.

At the outset I promised not to argue the curriculum development question. However it cannot be denied that Systems Analysis is a logical prerequisite for a course in Technical Writing. Further, if it is a pedagogical truism that we do not understand what we cannot explain (why else do teachers put essay questions on tests?), and if it is also true that writing forces us to think things through, then it would seem that Technical Writing is the logical conclusion to any course in Systems Analysis.

EMPHASIZING GENERAL EDUCATION SKILLS WITHIN THE DATA PROCESSING CURRICULUM

Dr. Ruth D. Armstrong and Dr. Gary R. Armstrong
Shippensburg University of Pennsylvania
Shippensburg, PA 17257

ABSTRACT

Today's technical revolution has made data processing faculty extremely cognizant of the need for frequent curriculum review. As we make changes to the technical content of our courses, there is a tendency to lose sight of the importance of general education skills and knowledge which are necessary for life-long learning. This paper will discuss methods and procedures by which faculty can emphasize general education skills within the data processing curriculum.

EMPHASIZING GENERAL EDUCATION SKILLS WITHIN THE DATA PROCESSING CURRICULUM

Today's technological revolution has made data processing faculty extremely cognizant of the need for frequent curriculum review. Although a data processing curriculum must stress concepts and knowledge which are equipment independent and which generalize to various applications, rapidly changing technology ultimately affects these concepts and systems as new procedures become available. As a result, faculty in this discipline frequently evaluate the data processing curriculum to assess its relevance to the business world, to discard obsolete topics which do not contribute to the basic foundation of the discipline, and to incorporate new topics which need to be added to this expanding knowledge base.

The importance of a broad background of general education is reflected by the curriculum standards of the American Assembly of Collegiate Schools of Business. However, as data processing faculty review their curriculum, they may have a tendency to become so enthusiastic about the data processing courses that they lose sight of the importance of a broad general education and a strong background in business. Frequently, however, faculty are reminded of the need for this solid background by recommendations of advisory committees and comments made by business recruiters. Sometimes the reminder is provided by the data processing graduates.

In a survey of Shippensburg University Business Information Systems department graduates employed in computer-related positions in business, the respondents were asked to evaluate the importance of 67 topics related to data processing. There was some agreement by a majority on the importance of one or two of these data processing-related topics. However, the strong response to the importance of the eight general knowledge and skills included in the survey was overwhelming. Seven of the eight general topics were rated as extremely important or highly important by over 90 percent of the respondents; the eighth item--leadership--was so rated by 79 percent of the respondents. These eight general education included on the survey are presented in Table 1 below.

TABLE 1
GENERAL EDUCATION SKILLS AND KNOWLEDGE CONCEPTS

Ability to Work Independently
Effective Oral Communications
Follow-through on Projects
Knowledge of Basic Business Operations and Applications
Time Management
Effective Written Communication
Reading Comprehension
Leadership

This paper will present ways to emphasize general education in data processing courses. Since most data processing curriculums include introductory and advanced COBOL programming courses, systems analysis and design, and a project-type course, ways of incorporating general knowledge and skills concepts into these four courses will be used to exemplify this integrative approach.

COBOL PROGRAMMING COURSES

As students prepare program specifications and documentation within a program, the COBOL professor can emphasize the importance of clear, concise, but complete communications. Students can be required to submit items such as programming specifications prior to completing the program. The professor can review these documents and evaluate them for acceptable English usage as well as for content. The papers can be returned to the student with comments and suggestions for necessary revisions, and students can be required to make necessary changes before submitting them with the program for a grade.

To reduce the burden of grading papers, the professor can evaluate the program specifications and internal documentation and make comments for revisions or improvements, but refrain from giving a grade for the work. After several assignments have been completed in this manner, the students can complete an in-class activity as a quiz grade. One such activity might be to give the students a problem and have them write the program specifications; another might be to give them a program without internal documentation and have them write the necessary comments. Not only can this reduce the necessity of pondering a grade for so many out-of-class assignments, but it ensures that graded work is independently prepared by each student.

When the programs are submitted, the professor can check the documentation within the program. The documentation within the programs can be checked randomly within each student's program rather than by reading everything. Although this will consume precious faculty time, less time will be required to check the program specifications as students improve. Also, after several programs, the professor will begin to identify the students who need to have their work reviewed more carefully.

A side benefit of stressing, emphasizing, and requiring good documentation is that students will have a set of completed programs they can take to a job interview. These programs can demonstrate to prospective employers the person's abilities.

To develop time managment skills and the responsibility for following through on projects, deadlines need to be set, and students need to be held to those deadlines. Some professors do not accept work after the deadline unless there is a valid medical excuse; others impose a penalty on late work which increases with the length of time the work is overdue. Directions for class organization such as class policies, submission of materials, etc., need to be given once and not repeated. Students will learn to read directions and/or heed oral directions only if they are required to do so.

In advanced programming courses, students need to be given the practice of solving some minor problems by referring to the COBOL reference manual for the computer system at the college/university. For example, the basic parts of the STRING and UNSTRING commands can be presented in class. However, assigned programs can require the use of optional features which the students are responsible for "digging out" for themselves, thus learning how to locate information and interpret it from the type of reference materials that will be available on the job. Also, this will help to develop a student's ability to work independently and become more self-reliant.

Throughout both programming courses, assignments should be realistic examples of information requested by a business organization. As students analyze the problem and design a solution, they will be aware of the importance of understanding the appropriate principles of accounting, management, marketing, business statistics, etc.

In advanced COBOL, students can be held responsible for understanding the business application for which the program is being written. Thus, if students do not understand the business concept and practice, they will need to research it rather than expecting the professor to explain it. Communication skills can be strengthened during program walkthroughs. One person in the group can be identified as the leader. That individual will be responsible for conducting the walkthrough and for submitting a one- or two-page summary of the activity. As an additional aspect, the team members can evaluate the assigned leader, providing feedback to that individual for improvement in leadership and communication skills.

To assist in teaching students the importance of preparing readable, easy-to-follow logic for maintenance programmers, students can be given requirements for planning and coding programs.

Occasionally the hierarchy chart for the top modules of a program can be given to the students. The class members can be divided into several groups. Each group is to write the program as a team. The group can assign each member certain modules to design and code; however, they must work together and coordinate their efforts. Once again, a leader can be designated and held responsible for conducting project development sessions as well as for seeing that the program is completed on time. The other members of the team can each be responsible for submitting a written summary for one of the project development sessions. This will require students to utilize communication skills, both oral and written, leadership skills, follow-through on projects, adherence to directions, and time management.

Although these activities may sound time consuming for the professor, it need not be the case. Every class activity and written document does not need to be graded; peer evaluation and professor's comments, oral as well as written, can provide students with feedback for skill refinement. Only selected activities and written documents need to be assigned a grade.

SYSTEMS DESIGN AND DEVELOPMENT COURSES

The day-to-day responsibilities of the systems analyst require the use of a variety of general education skills. Communicating with end users, developing system proposals, and coordinating systems design projects are but a few of the activities where technical expertise are supplemented with communications skills. As an integral part of the data processing curriculum, the systems analysis and design course offers numerous opportunities for emphasizing general education skills.

Case studies included in the textbook or provided as a supplement are often used to apply the tools and techniques covered in chapters and lectures. These can provide the instructor with opportunities to promote better writing skills as case studies are completed. Grading emphasis on grammar and sentence structure will encourage students to develop the habit of proofreading their work.

Students are often reluctant to make changes to their written assignments due to the tedious process of retyping the entire document. Using the microcomputer and word processing software enables students to focus their energies on creative thinking and writing.

A capstone course that is often included in the data processing curriculum is an applied software development project-type course. Students are assigned an application that requires them to apply systems analysis and design techniques as well as programming skills. With the increasing integration of microcomputers within the small business community, numerous opportunities are available for students to assist these organizations in developing microcomputer-based solutions. Their problems must be defined; data gathered through interviews; alternative solutions must be determined; and their cost and benefits stated. Possible software solutions include custom programming, packaged software, or alterations made to packaged software. Procedures must be documented; run manuals produced; and considerations given to end-user training. These activities provide the student with a realistic environment in which to exercise a number of general education skills.

The class can be broken into small groups, each responsible for developing a microcomputer-based solution for the proposed application. By keeping the class focused on the same application, the instructor can encourage competition among the groups. This also allows the instructor to better coordinate data-gathering interview sessions with members of the business organization.

Problems common to most small businesses, such as sales analysis, accounting, and inventory control, will require the students to apply their business knowledge and skills acquired within their curriculum in a realistic environment. Textbook case studies lack the variety of experiences that only working directly with end users can provide. Memos with questions that are too technical or that are vague, misinterpreted questions and answers, insufficient detail in reports, and indecisions by the end users are but a few of the outcomes that will force students to quickly make adjustments as they work with the end users in the systems study.

The data processing profession is often viewed by students as an occupation characterized by individuals working behind closed doors writing programs or running the computer--the thought of making oral presentations is something only management has to do. As the student groups complete each stage of the study (e.g., preliminary investigation, detailed investigation, systems design, and systems development) their oral presentations can be videotaped. This will give the students an opportunity to critically view their presentation and to strengthen areas that need attention.

The overall role of the instructor in this course is that of ensuring that each team is proceeding in the proper direction. The instructor should set completion dates for the various stages of the system development and coordinate meeting times involving the teams and the end users. With the instructor taking a rather passive role in the course, students have an increased opportunity to exhibit leadership skills. These group projects require a coordinated effort by each team member to ensure a successful completion of the system. The management of one's time is essential if responsibilities are to be fulfilled and deadlines met.

SUMMARY

Competition in today's job market among graduates with data processing and computer science expertise is increasing as the number of available positions within corporations slowly decrease. Companies are becoming more selective as the pool of applicants becomes larger. Graduates with sound general education skills will have the competitive edge. Therefore, it is important for data processing faculty to reevaluate their programs so as not only to provide up-to-date technical knowledge but also to prepare graduates with the general education skills necessary for life-long employment.