Lessons Learned in Teaching a Software Projects Course

Dale Hanchey
dhanchey@bison.okbu.edu
Computer Science Department, Oklahoma Baptist University
Shawnee, Oklahoma 74804, USA

Abstract

Today, almost all computer job ads include an experience requirement. The phrase "2 to 5 years experience" is typical. Why is this? Well, employers want more than employees who know **how** to do the work. Employers want people who can be given a task and who can handle it with little or no supervision. In other words, they want employees who know **what** to do. New college graduates may know the theory and possess the right skills, but will they know what to do when assigned a typically unstructured problem. A software project course can help students "put it all together". This paper outlines how a software projects course has been done at one university and summarizes lessons learned from sixteen years of experience in teaching that course.

Keywords: Projects course, software development, systems analysis and design, systems development life cycle

1. INTRODUCTION

In order to give students experience with a complete software project, our university implemented a software projects course in the 1986-1987 academic year. The course is part of a two-course series with the first course being a pre-requisite for the second. The first course, systems analysis and design, is taught in the fall semester. In that course, students learn how to do a project. The second course, a projects course, is taught the following spring semester. In it, students actually complete a software development project. The faculty member teaching the two courses has "real world" experience as a project leader of both large and small projects.

This paper describes both courses. It identifies things to do and things not to do. The intention is to provide "how to" instructions for anyone who wants to teach a successful projects course.

2. ONE COURSE OR TWO?

Many schools attempt to do a software project as part of a traditional systems and analysis and design course. This does not work well because there is not enough time. Too many compromises must be made. Such compromises include team size, project size, and last project phase completed. Unfortunately, a choice usually is made between completing a very small project and not completing a larger project. Neither choice is really satisfactory. There is simply not sufficient time in a single course to address both systems analysis and design and a complete, meaningful software project. For this reason, we chose to require two courses. This means that some other worthy theory or skills course cannot be required. It is a question of priority.

3. THE ORIGINAL PLAN

The original plan was to teach a traditional systems analysis and design course in the fall. In that course, students would learn the traditional system development life cycle (SDLC) and how to use it to perform systems development. That course would be primarily lecture with traditional SDLC assignments. Included in the assignments would be such things as data definitions, data flow diagrams, form designs, report designs and screen designs.

The following spring, students would take a totally self-contained projects course in which real (not contrived academic) projects would be completed. Project sponsors would be solicited from the university community in the fall. Based on surveys, students would be divided into teams of two to five with one of those acting as project leader. Potential projects would be presented at the beginning of the spring semester. The course instructor would ensure that approved projects were of the proper size (neither too large nor too small). Teams would choose their project from among those presented and approved by the instructor. The project sponsor would also serve as the primary user. Projects would be completed with formal presentations made during the last week of the spring semester.

4. ADJUSTING THE PLAN

When the original plan was implemented, most of it worked well. However, it became clear that there was not sufficient calendar time to complete projects of the desired size. The problem was not so much the number of student hours available or expended. The problem was with user time. Meetings with users were often delayed. Also, users required time to make necessary decisions. This problem was addressed by moving project selection back to the middle of the previous fall semester. Student teams are required to produce a requirements determination (RD) document and present it during the last week of the fall semester.

A second problem was team size. Team sizes of two to five students were tried. Experience has shown that two is too few and five is too many. Two students per team do not provide enough man-hours. Five students per team create logistical problems in scheduling. Three or four students per team provide the best compromise between these man-hour and scheduling constraints. If attrition were expected, a team size of four would be preferable.

5. PROJECT COURSE FORMAT

Our project course is a standard three-hour course. It is scheduled during a standard meeting time. All members of a team are required to be in the same section. The class is normally scheduled as a two-day-a-week class. Except for presentations, the class usually meets as a group for only about five minutes. Teams are then released to work on their own or meet with users.

A six-phase SDLC is used. The phases are as follows:

Phase 1:	Requirements Determination (RD)
Phase 2:	External Design (ED)
Phase 3:	Internal Design (ID)
Phase 4:	Programming and Testing (P & T)
Phase 5:	Training and Installation (T & I)
Phase 6:	Maintenance

Phase 1 (RD) is completed in the fall semester. Phases 2 through 5 are completed in the spring semester. Limited time is available for system maintenance. However, it is not uncommon for students to be involved in the maintenance of their systems long after the projects course is over.

Students are required to address all major components of an MIS. Students are tempted to address software only. However, they must also address issues of data, hardware, personnel and procedures. In particular, dealing with users is emphasized.

Each student team is required to document each project phase. Outlines for most of these documents appear in the appendices. The project team and the primary user determine the format of the user document.

Teams do formal business presentations at the end of phases 1, 2, 3 and 5. The purpose of these presentations is to inform the instructor and the primary user of the project status. Other faculty and students are also invited to the final presentations.

6. PROJECT GRADING

Most assignments in the projects are graded on a team basis. That is, all students on a team receive the same grade. The instructor grades documents. Presentations and software are graded by the instructor and by the members of the other teams in the class.

There is some individual grading in the projects course. There are peer evaluations within each team. There is also a final exam taken by the individual. The peer evaluations by fellow team members provide a strong incentive for students to participate in their projects.

7. PROJECT SOFTWARE TOOLS

Student teams determine the software tools for their projects. Some choose tools that they already know. Others choose tools that they want to learn. Either way, it is a good learning experience. Most teams choose wisely.

Many different software tools have been used on our student projects. The most commonly used tools include *Visual C++*, *C++ Builder* and *Visual Basic*. Many different data base managers have been used. Recently, students have been using *Access* and *Firebird*.

In recent years, more projects have been done to develop web sites. Common tools used on these projects have been *Dreamweaver*, *Flash*, HTML, Java, PHP and XML.

Some projects have used proprietary software for specialized equipment such as ID card scanners. Both magnetic stripe and bar code scanning have been used on projects.

It is not required that all of the code for projects be written by team members. In fact, students need to learn to research to find legally available code. For example, web chat software is available as a free download. Teams do not and should not write such code.

8. STUDENT BENEFITS

Students benefit from a projects course in several ways. First, they gain experience working as part of a team in doing a real, completed project. Students gain experience in designing, developing, documenting and presenting. Much is learned even if the project is not a total success. Some of the students gain project leader experience.

Second, students become intimately familiar with the SDLC. Working on a complete project is a much better way to learn the SDLC than by reading about it. Students who have participated in the projects class know what they must do in systems development.

Finally, the projects course experience makes graduates much more confident in their ability to contribute immediately. Some are willing to be project leaders on their very first job. It is common for job interviews to focus on the project course experience.

9. LESSONS LEARNED

Finding potential projects has not been difficult. Projects are solicited via email and by personal contacts. Typically, there are two to three times as many projects as there are teams to do the projects. This does present a small public relations problem. Potential project sponsors should be informed in advance that there is no guarantee that a project will be selected by one of the teams. All project sponsors should be thanked for their participation. Projects that are not selected the first time that they are presented may be needed and selected by a team in a future class.

Project sponsors should be informed in advance that they must commit approximately three hours per week for the duration of the project. This time will be used to meet with the project team and make needed decisions.

Projects have revealed some common student traits. Most computer students do not like to write. In general, computer students need more practice in presenting. Finally, some students perform well as an individual but have trouble working in a group.

It is important that the projects course instructor knows how to do a project. The instructor's primary job is to keep the projects focused and progressing. Scope creep can be a major problem for some projects. Some projects can stall when the team encounters a problem. Prodding a project leader is sometimes required. However, the instructor should not usurp the power of the project leader.

It may be necessary to kill a project. This is also common in the real world. Team members should then be distributed to other teams.

A team size of either three or fours students works best.

Class size should be limited to ensure that all teams in a section present during a single week. For that reason, it is recommended that no more than eight teams be in a single section.

A software projects course serves as an excellent "capstone" course for a computer degree program. It is effective in ensuring that students integrate their learned knowledge and skills. It provides a good "real world" experience with real users. It also provides experience working on a team.

Offering a successful projects course requires that the instructor take some risks. There is an inherent lack of control. Every project will not be a complete success. Keep in mind that the primary objective is for students to learn. Much can be learned from a failed project. Another concern is the variety of software used. Remember, it is not necessary that the instructor know every tool that every team chooses to use.

Requiring a separate projects course does necessitate giving up another required course. There is definitely a trade-off. Based on our experience, it is worthwhile.

10. BIBLIOGRAPHY

DeMarco, T. 1979. *Structured Analysis and System Specification*. Upper Saddle River, NJ: Prentice Hall.

Hoffer, Jeffrey A., Joey F. George, and Joseph S. Valacich. 2001. *Modern Systems Analysis and Design*. Upper Saddle River, NJ: Prentice Hall.

IBM. 1982. Business Systems Planning. Pp. 237-314 in *Advanced System Development/Feasibility Techniques*, ed. J. D. Couger, M. A. Colter, and R. W. Knapp. New York: Wiley.

Kendall, Kenneth E. and Julie E. Kendall. 2001. *Systems Analysis and Desig*n. Upper Saddle River, NJ: Prentice Hall

McFadden, F. R., J. A. Hoffer, and M. B. Prescott. 1999. *Modern Database Management*, 5th ed. Reading MA: Addison Wesley Longman.

Valacich, Joseph S., Joey F. George, and Jeffrey A. Hoffer. 2001. *Essentials of Systems Analysis and Design*. Upper Saddle River, NJ: Prentice Hall.

Yourdon, E. 1989. *Managing the Structured Techniques*, 4th ed. Upper Saddle River, NJ: Prentice Hall.

Yourdon, E., and L. L. Constantine. 1979. *Structured Design*. Upper Saddle River, NJ: Prentice Hall.

APPENDIX A

SAMPLE SOFTWARE PROJECT SCHEDULE

Week	Topic
1-3	External Design
4	External Design Presentations
5-7	Internal Design
8	SEMESTER BREAK
9	Internal Design Presentations
10-13	Coding & Testing
14	Implementation
15	Project Presentations
16	Final Exam

APPENDIX B

SAMPLE SOFTWARE PROJECT GRADING

Team Points
2 Project phases (ED, ID) @ 100 points = 200
- presentation (40%)
- documentation (60%)

Overall Project = 200
- presentation (20%)
- user document (20%)
- software (60%)

Individual Points
Comprehensive Final Exam = 200
Peer (team member) evaluation = 100

Total Points = 700

Grading Scale:

A	90%	630-700
B	80%	560-629
C	70%	490-559
D	60%	420-489
F	<60%	0-419

APPENDIX C

REQUIREMENTS DETERMINATION DOCUMENT OUTLINE

TITLE PAGE
ACKNOWLEDGEMENTS
TABLE OF CONTENTS
I. PROJECT SCOPE
    A. CURRENT ENVIRONMENT
        i. DESCRIPTION
        ii. PROBLEMS
    B. SYSTEM REQUIREMENTS
    C. NEW SYSTEM
        i. DESCRIPTION
        ii. SOLUTIONS
II. FUNCTIONS
    A. FUNCTION LIST
    B. FUNCTION DIAGRAMS
III. INPUTS
IV. OUTPUTS
V. COST/BENEFIT STATEMENT
VI. IMPLEMENTATION PLAN
VII. SUPPORTING DOCUMENTS
VIII. REQUIREMENTS DETERMINATION CONTROL SHEET

APPENDIX D

EXTERNAL DESIGN DOCUMENT OUTLINE

TITLE PAGE
ACKNOWLEDGEMENTS
TABLE OF CONTENTS
I. INTRODUCTION
II. LAYOUTS
    A. LISTS
    B. REPORTS
    C. SCREENS
    D. FORMS
III. DATA ELEMENTS
    A. LISTS
    B. DATA DEFINITION WORKSHEETS
    C. EXISTING DATA BASES
IV. APPLICATION STRUCTURE
    A. LISTS
    B. HIERARCHY DIAGRAM
    C. IPO CHARTS
V. SUPPORT REQUIREMENTS
VI. EXISTING APPLICATIONS SYSTEMS EVALUATION
VII. IMPLEMENTATION CONSIDERATIONS
VIII. EXTERNAL DESIGN CONTROL SHEET

APPENDIX E

INTERNAL DESIGN DOCUMENT

The document that is produced during the Internal Design Phase of a project should include as a minimum the following sections:

TITLE PAGE
TABLE OF CONTENTS
ACKNOWLEDGEMENTS
I. INTRODUCTION
II. DATA ORGANIZATION
    A. DATA BASES
    B. NON-DATA BASE FILES
III. PROGRAM SPECIFICATIONS
IV. SUPPORT REQUIREMENTS
V. IMPLEMENTATION SCHEDULE
VI. INTERNAL DESIGN CONTROL SHEET

The document may also include the following items if the environment dictates a need for them:

PROGRAM TEST MATERIALS
DEMONSTRATION MATERIALS
SYSTEM RUN MATERIALS
EXTERNAL DESIGN MODIFICATIONS
(ADDITIONS, CHANGES, DELETIONS)