A REALISTIC, TWO-COURSE SEQUENCE IN LARGE SCALE SOFTWARE ENGINEERING

Richard E. Bolz

Lawrence G. Jones

Department of Computer Science
United States Air Force Academy

ABSTRACT

We discuss a two semester, senior level sequence of courses in large scale software development. The courses are keyed upon the element of realism by having an actual user supply an actual project. In the first course students develop a functional specification of user needs. In the second course students design a system from the specifications and implement at least a portion of the project. We discuss the significant benefits of having a real project and point to some drawbacks. We conclude by discussing possible applicability of our courses to other schools.

1. INTRODUCTION

For a number of years computer science educators have recognized the need for courses in large scale software engineering. Unfortunately, studies indicate that satisfaction of that need usually falls short [Wass78], [Thay81]. Even where courses exist there are problems in conveying the concepts adequately. Many of the problems revolve around the central issue of realism of projects.

The problem of realism is tied to constraints of the academic environment. We are usually forced into using "toy" problems to be able to fit them into a semester or quarter. When students try to apply large system tools to small scale projects, they usually feel they are killing a fly with a sledgehammer. Worse yet, intellectually manageable problems allow students to make poor use of the tools and still reach their project goals. Thus, they are not really prepared to use the tools when it counts.

Another problem is that today's undergraduates are tomorrow's software project managers. There is little or no university training in software engineering project management and "toy" problems contribute little toward project management experience [Thay80]. The criticality of this problem is closer to home at the United States Air Force Academy where many of our graduates are charged with monitoring major software projects within a year of graduation.

Against this background the Computer Science Department at the Academy conducted an intensive curriculum review leading to a new curriculum in computer science. The new curriculum includes a two course "capstone" sequence in large scale software engineering. In this paper we discuss those courses. First, we discuss the general approach to the courses and state the objectives. We then get to the heart of the courses, the realistic term projects. Finally, we point to some ways the Academy situation might be translated to other universities.

2. THE SOFTWARE ENGINEERING COURSES

2.1 Introduction

The two semester software engineering courses are entitled Systems Analysis and Design I & II. The first

course concentrates on user requirements analysis. The second course concentrates on design and implementation. The sequence is restricted to senior computer science majors. By the time students start the first course, they have had courses in architecture, languages, algorithms and data structures. The background in small scale software concepts is essential for them to relate to the large scale software engineering concepts.

2.2 Course Objectives

The objectives of the courses are for students to:

1. Understand the need for a total life-cycle approach to system development.
2. Understand how to produce structured user requirements specifications.
3. Understand how to produce a good quality design from the specifications.
4. Explore implementation problems and how to deal with them.
5. Be exposed to practical experiences from computer professionals.
6. Gain practical, hands-on experience with a real-life, large application.
7. Learn how to work together to manage a team effort.

The most significant aspect of the course that ties the objectives together is the term project. The project will be discussed in detail later.

2.3 Course Mechanics

In this section we hit the highlights of the courses. Both courses are organized in similar fashion. The first half of each course presents necessary tools. The last half of each consists of work on the project.

Analysis and Design I begins with an overview of the system development life cycle and the tools to be used. Class is held in a seminar room and students lead the seminars on designated topics for the first 12 lessons using Yourdon's book as a text [Your79a]. The instructor then presents lessons covering the tools for requirements analysis using Gane and Sarsen [Gane77]. The seminar aspects involve the students in the material and allow the instructor and guest participants to interject real-life experiences that relate to the material. The rest of the semester is devoted to the term project with a few lectures scattered throughout.

Analysis and Design II follows the same general seminar approach but concentrates on design tools for the first half of the semester and the continuation of the project in the last half. While we survey other design techniques, the principal methodology is Yourdon and Constantine's Structured Design [Your79b], as presented in the student text, Page-Jones [Pag80]. Thus, transaction analysis and transform analysis are the principal means of generating the structured design from the set of functional specifications.

As mentioned, the term project is the heart of the course. It is the glue that ties the courses together and makes them work.

2.4 The Term Project

Our term project cannot be called a toy project. It is real. It has real users with real needs and they want real products.

Since we have two semesters to work with, the project can be of significant scope and complexity. We take advantage of the natural break between requirements analysis and system design. The students produce a functional requirements specification as the major product for the first course. The second course takes the specification, produces a design and implements at least a portion of the project. The amount of implementation is determined strictly by the complexity of the project.

This is the third year we have offered the courses and our projects have been:

1. Automate the Academy library including circulation, acquisition and cataloging.
2. Automate the summer program scheduling for the Commandant of Cadets. The Commandants staff must schedule all 4400 cadets for one of several dozen activities (and one vacation period) in three summer periods. Each activity has special requirements.

3. Automate the Cadet Clinic. Functional areas include primary patient care, laboratory, physical exams and standards, and environmental health.

The library project was obviously too large to complete. The summer scheduling project was also very large and only partially implemented. We are currently working on the clinic project. At this writing it appears that it is feasible to implement a substantial portion of the project. The shortcoming of not fully completing the project is addressed in the next section.

We have been fortunate in finding users who are willing to interact with the students. This teaches the students how to converse with laymen. It also gives them experience with the real-life frustrations when the user leaves out some information or changes his mind! The users understand at the outset that the students will probably not totally complete the project, but that they will get a good start on it.

A very important phenomenon takes place due to the realism. The students begin to assume a sense of ownership about the project. They seem to feel a greater sense of responsibility because if they don't do a complete job, they don't just let down an instructor. They would let down a user who really cares what kind of job they do. This allows the instructor to fade out of the picture and let the students manage the project. Evaluation of the effort is simplified because the instructor can see if the user is satisfied. Thus far our users have been quite pleased with the results.

Since the projects are large, teamwork is essential. We typically have 3 or 4 sections (15 students each) of the course and each section has its own functional area as a project. The students divide the workload by teams within a section and must conduct frequent walkthroughs to insure section continuity. They are also forced to interact with other sections to insure consistent interfaces with common areas. Thus they gain experience with teamwork, interpersonal relations and project management.

Since students follow a project through most of the life cycle, they suffer for mistakes they make in earlier phases. Also we make no special provisions to maintain the same students in the same sections across semester. Therefore, this year, a student only has a .25 probability of deriving a design based on specifications he helped produce. Early in the second semester there are cries of, "Who wrote this lousy spec?!!".

We are quite pleased with the way the project brings home the need for the development tools and provides real experience for the students. Feedback from our graduates indicates that the courses were very valuable to them,providing real-world experience which greatly eased their transition from student to computer professional.

3. Problems and Shortcomings

Now for the bad news. We are still constrained by limited time and cannot cover all aspects of the life cycle. Since the projects are picked by the instructor, the students don't get experience with performing a feasibility study. Since the students don't necessarily complete the implementation, they obviously don't get program maintenance and modification experience. Finally, again due to time constraints, they don't get much exposure to system sizing and cost/benefit studies. It might be possible to include these elements if the instructor were the user. He could have tighter control over the scope and direction of the project. However, we believe the benefits of the realistic project outweigh the possible benefits of the added topics.

We realize that many serious design and specification errors don't manifest themselves until implementation. To help address this problem, we require implementation of at least a stubbed version of the project. This does hold them somewhat accountable for mistakes in design and allows us to have time to put emphasis on the more important earlier phases of the life cycle.

Perhaps the biggest potential problem is in our dependence on an outside supply of users with real projects. Our situation is rather like that of a barnacle. If food (a project) doesn't happen to float by, we are in trouble. On the positive side of the analogy, the sea is rich with food and the Academy seems rich with projects. Since we have a support structure that runs like a small community, there are many people who would like computer help. If all else fails, we can fall back on an instructor generated problem and hope for better luck next year.

4. Summary and Conclusions

The key aspect that makes the course work so well is the realism. Project realism is possible due to the availability of real users and the two

semester length. Realism of experience is brought to the classroom via the instructor's contribution to the seminars. Realism of project management experience is brought about by the teamwork and accountability to the user. What remains to address is whether the Academy situation is transferable to other institutions.

A problem noted by Thayer [Thay80] is that many institutions feel they lack the necessary experience to run such a course. Since the Academy Computer Science faculty are all Air Force officers with operational experience, this is not as much of a problem for us. At other institutions it may be possible to fill the experience gap with sabbaticals to industry or have adjunct faculty from local industry help get the program on its feet.

Another major obstacle may be the availability of real projects. While the Academy has a very rich environment for such projects, the more typical university may also have many possibilities. The library, student health center, and athletic department are possibilities. You just need to find someone who could use some help and doesn't mind interacting with students.

While there may be obstacles to setting up such a program, we recommend it as a valuable sequence for any computer science program. The students come away with a real sense of accomplishment and our graduates continue to report the value of the experience.

REFERENCES

[Gane77] Gane, C. and Sarson, T. Structured Systems Analysis: Tools and Techniques. Improved System Technologies, New York, 1977.

[Paig80] Paige-Jones, M., The Practical Guide to Structured Systems Design. Yourdon Press, New York, 1980.

[Thay80] Thayer, R., Pyster, A., and Wood, R. "The Challenge of Software Engineering Project Management," Computer 13, 8 (August 1980), pp 51-59.

[Thay81] Thayer, R., Pyster, A., and Wood, R. "Major Issues in Software Engineering Project Management," IEEE Transactions on Software Engineering 7, 4 (July 1981), pp 333-342.

[Wass78] Wasserman, A. and Freeman, P. Software Engineerig Education: Needs and Objectives. Springer-Verlag, New, 1976.

[Your79a] Yourdon, E. Managing the Structured Techniques. Prentice Hall, Englewood Cliffs, NJ, 1979.

[Your79b] Yourdon, E. and Constantine, L. Structured Design. Prentice Hall, Englewood Cliffs, NJ, 1979.