A PROJECT-ORIENTED COURSE FOR SOFTWARE SYSTEMS DEVELOPMENT

Harbans L. Sathi
Department of Mathematics/Computer Science
SUNY College at Brockport, New York

Introduction

At SUNY College at Brockport, we offer a Computer Science program that prepares students for a wide variety of postgraduate and career goals. The emphasis is on a sound scientific education equipping students with sufficient ability to cope with unforeseen circumstances. At the same time, we are also aware that for most of them, scientific life is not going to be the real life. Most of them start as programmers, entrusted with the task of maintaining already developed systems or with developing new systems in team environments. This requires preparing students for what has come to be known as "Software Engineering." But can engineering be taught? Engineering requires experience and experience is acquired and not learned from a textbook or in a classroom. What should be done then? We should combine theory with practice. At SUNY, we have made a modest effort and introduced a course entitled "Software Systems Development" (also called CSC 426). The course has been modelled along the lines of course CS14 as recommended in the ACM CURRICULUM '78 [1]. Some recommendations in the course CIS-7 of the DPMA Model Curriculum [3] have also been included. In Section 1, we describe the course organization together with its objectives and course contents. It also lists the textbooks used and describes administration of the projects and grading criterion. In the next section, we enumerate the various problems encountered.

1. Course Organization

Software Systems Development is a one-semester course that is concerned with the design and development of large-scale systems. It introduces the topic through a series of lectures, reading assignments, exercises and one medium-sized project. The project is to be analyzed, designed and implemented in team environments. Briefly, the course objectives are:

(i) To understand the various phases of large-scale systems development activity.

(ii) To apply structured techniques to these various phases, viz., system analysis & problem specification, system design, implementation, testing, verification and maintenance.

(iii) To apply technical, managerial, communications and interpersonal skills to a realistic, medium-sized project set in a team environment.

1.1 Prerequisites

This course has been offered for the first time at SUNY College at Brockport during the last spring semester. (Prior to this, I have taught this course at Central Michigan University, Mt. Pleasant, Michigan.) The students drawn to this course are undergraduates in their junior or senior year, and some part-time students who are employed in some industry or business or government organization. Some of them may be in some computer-related fields. They have already taken an introductory programming course in PASCAL (CSC 203) and a course on data structures (CSC 205). They should have taken another programming course in PL/I or COBOL (CSC 215 or CSC 214) to be able to manipulate ISAM and direct files. They should be able to organize and write reports in clear and concise English. Besides, they should have good skills to communicate with others.

1.2 Topics Covered

The course introduces the software systems development process through a series of lectures, reading assignments and exercises. The course, being just a one-semester course, demands a lot of time on the part of the student. The following topics are included:

(i) Systems Development Life Cycle. Various phases of the system's life cycle. Functional specification, systems analysis, design, implementation, testing and verification, acceptance, maintenance.

(ii) Systems Analysis. Tools and techniques, definition of user requirements, study of the current system, modelling the new system. Documentation. Use of data flow diagrams.

(iii) Structured Design. Various design methodologies. (e.g. Functional Decomposition. Data Flow Design, Data Structure Design, etc.). Organization of modular systems, modular coupling and cohesion, transform analysis, and transactional analysis.

(iv) System Implementation. Reinforcement of structured programming concepts. Programming style.

(v) Testing and Verification. Testing tools and techniques. Derivation of test plan & test data. System testing and integration testing. Measures of software reliability.

(vi) Maintenance. Definition and dimensions of maintenance, maintenance modelling, ripple effect analysis and maintenance tools.

(vii) Project Organization and Management. Basic software management concepts. Team organization, Chief Programmer Team, Quality Assurance, Structured walkthroughs, Team/Group dynamics and conflict management.

(viii) Communications. Conducting user interviews, writing system and program documentation, Presentation of various project reports.

1.3 Textbooks

There is no one book that covers all the topics enlisted above. This makes the choice of a textbook a very difficult task. However, I require the students to acquire "Structured Analysis" by Victor Weinberg [9]. Other recommended texts are Myres [5], Trausworthe [8], Jackson [6], Yourdon and Constantine [10]. Reading assignments are given from these recommended texts and other recently published material on the subject [2] & [4]. One assignment is to present a review of one recently published paper in a journal. A summary of the paper is to be circulated to each student in the class at least one day before the presentation.

1.4 Team Projects

Team project is an essential feature of this course. The purpose is to expose the students to the real world environments, and to give them a "hands-on" experience with the development of a medium-sized system. They should be able to blend formal academic methods with the knowledge of field practices.

The class is divided into groups of five each. This lets students face the issues of project management. Students are free to form groups of their choice. But they should join a group so that they may all program in COBOL or PL/I so that they may not have programming problems later.

The selection of a project is a very difficult task. One has to be very careful that this may be a medium-sized one and not a "toy." This should reflect the real world situation. Its size should be such that students can complete it during the short span of the semester at their disposal. So far, I have been assigning a project myself. But, in the future, I intend to provide a list of suggested topics and also permit students to come up with an acceptable suggestion. (Last Spring, the students developed a text editor.)

Each team chooses a project leader and a secretary (recorder). In the first meeting, they set up milestones, the timings of their weekly meetings, etc. Subsequently, the roles are rotated to give equal chance to every member of the team.

Projects are to be submitted in phases. Milestones are earmarked and schedules for their submission are provided to the students at the time of project allocation. Complete documents are to be submitted on due dates. There are some oral presentations, including a final demonstration of the system. This gives them an experience in development of their communication skills. Besides participating in a team project, a student participates in a walkthrough of some other team. Such walkthroughs are held at the end of each milestone. These are a very important part of a project.

1.5 Grading Criterion

Grading is based on the following scheme:

(i)	Exams:	one mid-term	15%
		one final	20%

(ii) Individual performance: 15%
(oral presentation, written assignments & reports)

(iii) Project: 35%

(iv) Performance in team: 15%

Evaluating the student project is a difficult task. Each team receives one grade for all members. Each student is given a separate grade for his/her individual performance in the team. This is done in consultation with the student's peers. Each time a team leader is finished with his/her turn as a leader, he/she submits a report about the performance of each team member. All the reports collected about a team member are used to determine the performance.

Problems Encountered

(i) Lack of suitable textbook. No one single textbook is available that covers all the topics one would like to cover. And then, there is inconsistency of material. As observed by Spicer [6], 'Mix-and-Match' is not a possibility.

(ii) Selection of Projects. For the project to be worthwhile, it should be a real-world problem and not a "toy" to play with. Students should be able to apply the concepts they learn for a large-scale project to the project assigned to them for this course. The product they deliver should be marketable. At Central Michigan University, I assigned a project that set up an information system a Placement Office at a university could use. At SUNY at Brockport, students developed a text editor. Both these projects were assigned by me. To assign projects that a business or industrial organization may use requires interaction with such an organization. Efforts are being made to set up such interaction.

(iii) Incompatibility of Team Members. Some of the students who registered for the course are part-time students. They work full-time at some other organization and are not available to put in as much time as the course demands or as the team leader may expect them to do. The course is offered in Spring and one has to cope with the vagaries of weather. Commuting students face this problem. Personal jealousies among some team members is another problem to be reckoned with.

(iv) Size of the Course. There is so much material to be covered in one semester. Then the student is expected to do a lot of reading besides the project. A typical suggestion has been to offer this as a 2-semester sequence.

(v) Lack of Users Availability. Finding users who can interact with students is a big problem. At Central, I was lucky to find a colleague who was willing to act as one. This afforded a good opportunity to the students to interact with him in finalizing the functional specifications of the project.

3. Conclusions

This course has developed over the last three semesters. From the comments that I have received from my students, I can say that the course has been found to be quite beneficial. Several students have used the various techniques they learned during this course in their professional jobs they have taken upon their graduation.

References

[1] ACM Curriculum Committee on Computer Science, Curriculum '78 Recommendations for the Undergraduate Program in Computer Science, Comm. ACM 22,3 (March 1979), 147-166.

[2] Bergland, Glenn D. & Ronald D. Gordon, "Tutorial: Software Design Strategies." IEEE Computer Society, NY 1981.

[3] Adams, David R. & Thomas H. Athey (ed), "DPMA Model Curriculum for Undergraduate Computer Information Systems Education," Data Processing Management Association, IL 1981.

[4] Miller, Edward & William E. Howden, "Tutorial: Software Testing & Validation Techniques," IEEE Computer Society, NY 1981.

[5] Myres, Glenford J., "Software Reliability," John Wiley & Sons, NY 1976.

[6] Jackson, Michael, "Principles of Program Design," Academic Press, FL 1975.

[7] Spicer, Joseph C., et al, "A Spiral Approach to Software Engineering Project Management Education," ACM SIGSOFT Software Engineering Notes, Vol. 8, No. 3 (July 1983) Pg 30-38.

[8] Trausworthe, Robert C., "Standardized Development of Computer Software," Part I & II, Prentice Hall, Englewood, NJ 1977.

[9] Weinberg, Victor. "Structured Analysis," Prentice Hall, Englewood, NJ 1980.

[10] Yourdon, Edward & Larry L. Constantine, "Structured Design," Prentice Hall, Englewood, NJ 1979.