A Dual Approach to Software Engineering Project Courses

K.L. Nance
Department of Mathematical Sciences
University of Alaska
Fairbanks, AK 99775-1110

P.J. Knoke
Department of Mathematical Sciences
University of Alaska
Fairbanks, AK 99775-1110

Abstract

In the effort to produce knowledgeable software engineers who have some practical experience, the University of Alaska, Fairbanks (UAF) has adopted a dual approach to software engineering education. The approach includes two courses each of which addresses specific problems generally associated with software engineering project courses and both of which meld together to introduce and reinforce software engineering principles. Upon successful completion of both courses, students have reached the "evaluation" level of the appropriate educational taxonomy and have the educational background and experience necessary to become successful software engineers in both team and individual settings.

1 Introduction

Student projects are the hands-on components of Computer Science/Software Engineering courses and are very valuable in aiding the learning process. In some Computer Science courses these student projects take the form of assigned "toy" projects. These are often programming projects involving little or no requirements analysis, documentation, or testing because the requirements are often decreed by the instructor and documentation and testing are considered to be of secondary importance. Such projects do little toward reaching the goal of an internalized student software development process or personal software process [4] which leads to high quality software in large software developments.

It is a challenge to try to use the mechanism of student projects while also requiring the use of valid software engineering concepts and methods in the projects. The latter are best learned and applied in the context of real and large-scale software developments. However, few universities can provide a real and large-scale software development environment, especially one mapped into a 15-week semester context. In an effort to overcome these traditional barriers, UAF has adopted a dual approach to software engineering project courses which enables students to enroll in two distinct software engineering project courses. In the first course the students participate in a "toy" or "peanut project" and in the second course they participate in a real-world project with a real client other than the instructor.

2 Course Descriptions

For the purpose of the treatise, the courses will be referred to as Senior Project 1 and 2 (SP1 and SP2). The two courses are offered annually, with the first course, entitled Software Engineering (SP1) offered in the fall semesters and the second course, entitled Senior Project and Professional Practice (SP2) offered in the spring semesters. SP2 is a graduation requirement. SP1 is currently an elective, but one which most students choose to take. Both courses have the same prerequisites, with the added restriction of SP2 that the student must be a senior. Most students take the courses in sequence. The department is constantly evaluating the Computer Science degree requirements, and the possibility of making both courses required is currently being discussed.

3 SP1 - Software Engineering

The first course, SP1, is entitled Software Engineering. Topics covered in the course include software requirements, design, implementation, test and documentation. The course utilizes the "peanut project" approach to software engineering education. It consists of the required use of full-strength software engineering methods for individual student-defined small projects. The projects are called "peanut projects" because the use of these powerful software engineering methods for small projects is an obvious overkill, analogous to cracking a peanut with a sledge hammer.

3.1 Implementation

Peanut projects serve as the hands-on component of SP1. There is some benefit to the use of this name. It highlights the intent of the approach and its purpose (deliberate overkill for educational reasons). Understanding this, students tend to accept it with good grace. The approach has had good results in the UAF educational environment. Formal written documentation for the peanut projects requires the that project notebook include the following documentation: project description, project plan, software requirements, software design, software implementation, test plan, test results, and project summary. Students are able experience all phases of the software development life cycle and to examine their personal strengths and weaknesses. This course with its embedded peanut projects, has been offered four times. The number of students completing the first four offerings was 5, 6, 19, and 18.

The text used in all offerings has been Pressman's "Software Engineering: Practitioners Approach,(3rd Ed)"[7]. Selected readings from Pressman are supplemented by various current handouts from such sources as IEEE Software, SEI Bridge, FASE and SPIN reports (regarding the latter, in a recent semester two traveling students found an opportunity to attend a Silicon Valley SPIN debate on software metrics, and they reported on this in class upon their return to Fairbanks). In addition to the peanut project, each student is required to complete 12 homework assignments and take three exams during the course of the semester. Selected students also give class presentations on their projects at various project stages.

3.2 Example Projects

The peanut projects tend to be quite diverse because of the students' different interests and backgrounds. One semester resulted in the following projects being chosen for implementation:

1. Rendering And Displaying Splines
2. Home Cook Book
3. Movie Player For PC
4. Baseball Card Database
5. 3-D Object Modeling Program
6. Parallel Processing For A Scheduling Application (T3D)
7. Software For SAR Earth Station
8. LBLGAS (Atmosphere Model For Geophysical Institute)
9. Alaska SAR Facility User Services Tool
10. Materials Management System
11. Mandelbrot Set Animation Program (SGI)
12. Home Library Database
13. Alaska SAR Facility Valid Dataset Tool
14. Keno Game Simulator
15. Database Management System
16. Aegis (Classroom Management Tool)

In this set, the projects range in size from about 300 to several thousand lines of code, and in type from games (Project 14) to re-engineering (Project 8). Design methods include classical structured programming, object-oriented, and event-driven models, and several students managed considerable reuse. Some of the implementations used tools like Visual Basic and Microsoft Access, while others used Fortran (e.g., for the T3D project), C, and C++. One project used some Assembly Language (Project 3). Implementation platforms range from PCs to the Cray Y-MP, the Cray T3D, and the SGI Onyx. Some of the students are undergraduate Computer Science majors with no software engineering experience, while others are employed at such local places as the University of Alaska's Geophysical Institute, the Alaska SAR Facility, the University of Alaska Computer Network, and the Arctic Region Supercomputing Center. Some of the employed students have up to 14 years' experience as practicing software engineers.

Each project requires an associated project notebook. Each project notebook must include the following materials:

1. Project Description - A Few Paragraphs, Format Per Student's Choice
2. Project Plan - Format Per "Software Project Plan", [7]
3. Software Requirements - Format Per "Software Requirements Specification Outline", [7]
4. Software Design - Format Per "Design Specification Outline", [7]
5. Software Implementation - Specified To Include Source Code With Comments
6. Test Plan - Format Per "Test Specification Outline", [7]
7. Test Results - Format Per Students' Choice
8. Lessons Learned - Format Per Students' Choice

Students are required to turn in draft chapters of the project notebook on a scheduled basis over the 15 week course period, and an attempt is made to synchronize lecture subjects with the project notebook chapters coming due. For the final project notebook, students are allowed to revise and polish their draft chapters.

3.3 Benefits

As with any of the many approaches to presenting software engineering project courses, there are problems and benefits. Identified benefits associated with the peanut project approach include the following:

- Students tend to be interested in the project itself because it's something they have chosen and want to do.

- Because of the approach, some students tend to be turned on to the software engineering subject matter.

- Some students learn the software engineering methods and concepts quite easily particularly if they have had any real world programming experience)

- A number of the peanut projects turn out to be operationally useful (e.g., those projects that develop small software tools)

- Some students improve their presentation skills because of having to present their project results in class

- Students learn from each other, especially during the presentations. The mix of student backgrounds is also helpful in this regard.

- The approach could lead to a kind of internalized "Personal Software Process" [4]

- The approach automatically points out some problems in the state-of-the-art of software engineering today.

A major benefit of the approach is that it automatically points out some problems in the state-of-the-art of software engineering today as well as the state-of-the-practice. The reality of the peanut projects presents students with the opportunity to apply text book learning to a real situation. They quickly find that idiosyncrasies in individual projects frequently require a new interpretation or adaptation of standard practice. Several groups noticed that the principles of a "good" software design as presented in the text do not apply well to object-oriented design or event-driven design, and adapted the concepts accordingly.

3.4 Problems and Pitfalls

Problems and potential problems identified over the offerings of the course include the following:

- It is difficult to synchronize software engineering material with project phases, and the attempt leads to skipping or skimming some important concepts.

- Some of the project notebook outlines decreed as standard ideals for use aren't well supported by discussions in their host text chapters.

- As the class size increases, permitting all students to give in-class presentations significantly impacts the amount of material which can be covered in the course.

- As the class size increases, the instructor is required to keep track of the details associated with many diverse projects

Some of the problems introduced in this course continue in SP2 while some are solved automatically by the nature of the second course. The synchronization issue is resolved to a degree by the wealth of prior knowledge which students have prior to enrolling in SP2. The class size is not as much of an issue in SP2 as the group project approach is utilized rather than the individual project. The standard ideals issue continues to be a problem especially when considered with respect to the varying nature of the projects themselves.

4 SP2 - Senior Project and Professional Practice

The second course, SP2, is entitled Senior Project and Professional Practice. This course requires that software engineering principles be applied by a software engineering project team to a real-world project . Projects are solicited from the university and community prior to the semester in which the course is offered. Students choose their own software engineering project teams and each team selects a project from the pool of projects which have received the instructor's preliminary approval.

4.1 Implementation

The original stated goals of this course are to provide realistic project experiences for the students, develop their software engineering skills, address computer ethics issues, and improve the students' written and oral communication skills. [5] In addition to the stated goals, a myriad of beneficial skills are gained by the students. Examples of important additional skills are the ability to cope with group dynamics, the ability to deal with unexpected situations, and the ability to pursue a stated goal while dealing with schedule and cost constraints.

4.2 Example Projects

Examples of projects implemented over the past two years include the following:

1. UAF Public Relations - Match Student Home Addresses With Local Newspapers

2. UAF Museum - Integrate Database For Four Collections, Do Web Page

3. UAF Alumni - Develop Mac-Based Alumni Database System

4. Bureau Of Land Management - Develop Cabin Reservations System

5. North Star Volunteer Fire Department - Track, Record And Print Fire Personnel Training Records

6. Eielson High School - Develop Data Base System For High School Curriculum Requirements
7. UAF Center For Health and Counseling - Develop Student Appointments Scheduler
8. Alaska Student Rocket Project - Develop System For Post-Flight Analysis Of Research Rocket Telemetered Data

Each group was required to submit a detailed project notebook including the following materials (average page lengths are indicated in parentheses):

- Proposal (12)
- Project Plan (9)
- Requirements - User's Manual (27)
- Design (16)
- Source Code/Implementation Details (87)
- Test Plan/Results (11)
- Customer Acceptance Document (3)
- Project History/Evaluation (3)

4.3 Benefits

A number of benefits of this course have been identified by the students and instructors. Benefits for the students are listed below with benefits for UAF and the community following. The major benefits complementing the educational experience include the following:

- Students are pleased to have a project which they can include on their resumés.
- Students are provided with excellent exposure to real-world software project management issues including schedule and cost constraints.
- Students are exposed to requirements engineering and associated matters such as determination, analysis, and "requirements creep"
- Students are exposed to risk management concepts as applied to software development.
- students learn to work as a team toward a goal, and are faced with development under realistic stress conditions.
- Students are exposed to full "waterfall model" with the exception of the software maintenance phase.

The students are not the only ones benefitting from this educational experience. In addition, since real-world projects are sought, there is a win-win situation for UAF and the community. Benefits of this nature include the following:

- Students assist community by doing needed small software development projects.
- Community assists students by providing them with a real-world software development environment.
- Helps fulfill the "service" component fo the university mission.
- The image of the university has improved in the community.

4.4 Problems and Pitfalls

Like any approach, this one has its drawbacks as well as its benefits. So far, in 15 years of use, the benefits have far outweighed the drawbacks. Nevertheless, drawbacks and potential drawbacks do exist. Some of the major identified problems and potential problems are enumerated and discussed below.

1. The desired software may not be developed to a satisfactory working model.
2. Some faculty may not be comfortable teaching the course because control is limited.
3. If too many project fail, the supply of projects may diminish.
4. There may be adverse legal implications of a failed project.
5. The approach leads to a constant need for new computer systems and software.

Item 1, above, necessitates the view of potentially "passing a failed project." Ideally, this circumstance will not arise particularly if proper screening is done of the projects at the beginning of the semester. This will minimize the potential of failure. It will also minimize the impact of a potential failure if the customer is advised of the possibility at the beginning. Item 2 is more a personality issue. Unexpected, and often undesirable, events occur with almost every project. They are the rule rather than the exception. Instructors willing to undertake such projects will find the benefits to the students generally outweigh the discomfort as the lack of control.

Items 3, 4, and 5 are itemized as potential problems. They have not occurred in a manner that requires they be dealt with, but accurate risk assessment necessitates their inclusion. Item 4 anticipates the fact that a customer, even though nothing is paid for the development, could potentially sue the university for damages. Item 5 is a potential problem which is dealt with every semester. It is classified as a potential problem, because it has not caused project failure, but rather causes project elimination. In this era of ever-declining university budgets, UAF is unable to acquire the software and hardware necessary to facilitate some projects. Currently, this can eliminate an otherwise excellent project from consideration if the customer is unable to provide the students with the necessary development environment.

5 Educational Implications

The duality of the course structure allows students to achieve higher levels of the Software Engineering Institute's (SEI) taxonomy of educational objectives for software engineers. The SEI taxonomy was adapted from Bloom's taxonomy [1] and includes the following levels: knowledge, comprehension, application, analysis, synthesis, and evaluation. The first level, knowledge, is reached upon the memorization of facts with no deep understanding. [6] This level is easily achieved in the first course through required information processing upon which students are quizzed. Comprehension is the second level and is demonstrated by the ability of a student to put a concept into his own words. [6] Capability at this level is demonstrated in the first course through the use of examinations, and through the second course in the group interaction. Group dynamics necessitate the ability to reword a concept repeatedly and in many different ways in order to effectively communicate a concept to another individual. The heterogeneous population from which the groups are chosen in the second course necessitates that students achieve "comprehension" prior to developing the group project.

The third level of the taxonomy is application. Students achieve this level when they are able to apply specific concepts to a specific situation. The fact that both courses require that students be involved in a specific project obviously also necessitates that they be able to apply specific concepts to such projects. The fourth level is analysis. This level includes the ability to identify structures and relationships. This level is demonstrated in both courses most specifically by successful completion of the requirements analysis phase of the project. The fifth level is synthesis and can best be shown by the ability to combine elements to produce a pattern or structure not previously presented. [6] This level is demonstrated by the production of a project plan which fulfills user requirements and provides a means of constructing a required artifact. The final level is evaluation. This can be demonstrated in the ability of a student to recognize potential process improvements. [6] This is thoroughly demonstrated in the final phase of the second course in which the projects are evaluated in retrospect and process improvements are suggested.

6 Summary

What are the the major advantages and disadvantages of presenting the courses in this manner, i.e. two-courses with two-projects as opposed to one course with one project? At UAF, eight major advantages have been identified. The first is that this approach allows the students to first learn on a small-scale, self-defined project and then apply what they have learned to a larger real-world problem. This minimizes some of the risks associated with immersing students in real-world projects before they have the tools and experience necessary to solve the problem.

The second advantage is that this approach provides the students with a sufficient depth and breadth of knowledge in the first course to effectively concentrate their efforts in the second course on coping with group dynamics, solving a real-world problem, honing their skills, and determining their personal strengths and weakness which better enables them to choose a good group and a good role within the group.

The third advantage and one upon which current emphasis is focused is that the students do have the opportunity to have an active role in a real-world project. This provides the students with a marketable skill which they can include on their resumes.

The fourth advantage is that students are afforded the opportunity to work with and learn about the individuals and businesses in our community. The students learn not only about their projects, but also about other students' projects through the use of team and individual presentations.

Another advantage is that this approach allows the University to provide community service by solving real-world problems at little or no cost to the user. This facilitates the sixth advantage which allows local to learn about software engineering. This has served both as a recruitment tool and provided community businesses with good models of software engineering practice.

The seventh advantage is that this approach requires all individuals to experience all phases of the software engineering process. Since the first course requires the individual to handle all phases of the "peanut project," the student are better able to choose the phase(s) of the process to which they are the most suited.

The final advantage which was identified is that this approach provides students with confidence in their ability to solve software engineering projects both in a group and individual setting, whether dealing with a real-world problem or a hypothetical situation which has more flexibility.

Of course, this approach is not without its disadvantages. Most of the disadvantages are associated with each specific course and were addressed under problems and pitfalls in the previous sections. An identified disadvantage which reflects on the combination of courses as currently taught includes the fact that some students take the courses out-of-order. Allowing some students to do this is necessary to accommodate students. The disadvantage of having occasional students take the classes out-of-order is currently outweighed by the wish to accommodate students and allow them graduation flexibility.

The dual approach has been tested and works well at UAF. The peanut project provides the students with a sound foundation of software engineering principles, and the tools and techniques to effectively apply these principles on a project. This foundation provides the depth and breadth necessary for the student to function effectively as a member of a software engineering project team in the application of a real project. The solid foundation enables the students to cope with group dynamics, client demands, and other real-world experiences encountered in the second course. The final result is confident and competent software engineers.

References

[1] B. S. Bloom. "Taxonomy of Educational Objectives: Book 1, Cognitive Domain." Longman, Inc., New York, 1956.

[2] G. A. Ford and J. E. Tomayko. "Education and Curricula in Software Engineering." *Encyclopedia of Software Engineering*, John Wiley and Sons., Inc. 1994

[3] T. Forester and P. Morrison. "Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computing," The MIT Press, 1990.

[4] W. S. Humphrey. "A Discipline for Software Engineering." Addison-Wesley Publishing Company. 1995.

[5] P.J. Knoke. "Medium Size Project Model: Variations on a Theme." in *Lecture Notes in Computer Science - Software Engineering Education.* J.E. Tomayko, Editor. Springer-Verlag. 1991.

[6] K. L. Nance and M. Strohmaier. "Incorporating Reuse Into Software Engineering Project Courses." *Proceedings of the IEEE Conference on Software Engineering Education.* 1994.

[7] R.S. Pressman. "Software Engineering: A Practitioner's Approach, 3rd Edition." McGraw-Hill. 1992.