Software Engineering Design:
A Laboratory in Building Team Management Skills

Susan E. Conry and Douglas J. MacIntosh
Electrical and Computer Engineering Department
Clarkson University
Potsdam, NY 13699-5720

Introduction

> *"Engineering is problem recognition, formulation, and solution. In the next 20 years, engineers and engineering students will be required to use new tools and apply ever-increasing knowledge in expanding engineering disciplines, all while considering societal repercussions and constraints within a complex landscape of old and new ideas. They will be working with diverse teams of engineers and nonengineers to formulate solutions to yet unknown problems. They will increasingly need to address large-scale systems problems."* [1]

Engineering in the 21st century requires teamwork. The problems are large and open ended, requiring expertise across a range of disciplines. Because solving these problems demands competent engineers, engineering education has been focused on educating the individual – enabling each student to acquire the fundamental knowledge and skills associated with his or her chosen discipline of study. Students are evaluated using metrics that assess the knowledge and skills they have acquired as individuals. Yet engineering students must learn the skills they need to work effectively on teams.

Software engineering design provides an ideal framework for teaching students the skill set that they need to function well on interdisciplinary teams. The problems involved are multifaceted and large. Solving them requires contributions from groups with differing functional responsibilities, yet the product cannot be delivered in the absence of coordination and cooperation. This paper describes a two course sequence in software engineering design that focuses attention on design and teamwork. Specifically, we discuss the nature of these courses and give an indication as to some of the strategies that are used to guide students as they learn to work effectively in large teams.

Teamwork Skills

Engineering curricula are usually highly structured, building in each student the foundation of technical expertise in mathematics, science, and engineering that enables graduates of engineering programs to solve real-world problems. In the 21st century, though, the problems are large and complex. Their solution requires contributions from a wide variety of disciplines and from individuals operating as a problem solving team. Engineering educators must find

ways of helping students to develop the skills they will need if they are to function well on problem solving teams.

A team is a group of individuals who work together to accomplish a common goal. A good team is one in which each of the team members understands the common goals and the direction the team is taking. It is one in which team members understand the roles they play and the ways in which they can make contributions. Team members understand that even though problem solution is a team effort, each individual on the team must work together with his or her colleagues and must also work independently to complete his or her assigned tasks. It is critically important that clear lines of communication be maintained so that frustration and confusion can be avoided whenever possible.

In learning how to work effectively on a team of this kind, students must develop a number of attributes and skills. In particular, engineering students need to learn:

- how to work within the dynamic of a large (more than 6) group of people with diverse backgrounds
- that a personal commitment to contributing one's own share is critical to team success
- that each team member must stay focused on the common goal and on his or her own assigned tasks
- that once a collaborative team decision has been made and a plan for the team formulated, everyone on the team must support the outcomes
- that each individual's thoughts and ideas must be respected and valued as the collaborative effort develops
- that it takes "give and take" to arrive at the best solutions and that confusion and uncertainty are clarified through questioning and discussion
- that differences and conflicts occur, that they must be acknowledged, and the best way to deal with them is constructively and openly.

Engineering curricula today are packed with technical content. Increasingly, there is so much for students to learn about their chosen discipline that there is little space in the curriculum for coursework that deals directly with these kinds of issues. It is hard to find space in the typical engineering curriculum for a course in teamwork skills. The theme of this paper is that strategies for guiding students in their acquisition of teamwork skills can be integrated in their design experiences.

Curricular Structure

The two course design sequence described in this paper forms the culminating set of design experiences in a software engineering program. Prior to the first of these courses, students have completed a two course sequence (SE1 and SE2) in C++, object oriented programming and fundamentals of software engineering, a course (SE3) that treats generic programming, software components, and programming for reuse, and a course (SE4) involving user interface design, event driven programming and programming for visual environments.

In SE1 and SE2, students learn C++ and object oriented programming. They are also introduced to the software life cycle and problems that arise at differing stages of this cycle. The assignments and projects they must complete vary in size and scope. Students in SE1 do their assignments individually and the problems are small. In SE2, students do on the order of eight or so small assignments individually. In addition, they complete three or so larger projects in which they may be allowed to work in groups of two. In SE3 and SE4, students continue to build their individual skills, but projects are typically completed by small two or three person "teams".

In the courses that are prerequisite to the design sequence, students gain experience with an increasingly broad range of activities in the software life cycle. One can view the software life cycle as having several stages, though iteration through these stages is common. (Design is, after all, an iterative process.) Figure 1 depicts the software life cycle stages and the focus of student experiences in coursework encountered prior to the design sequence.

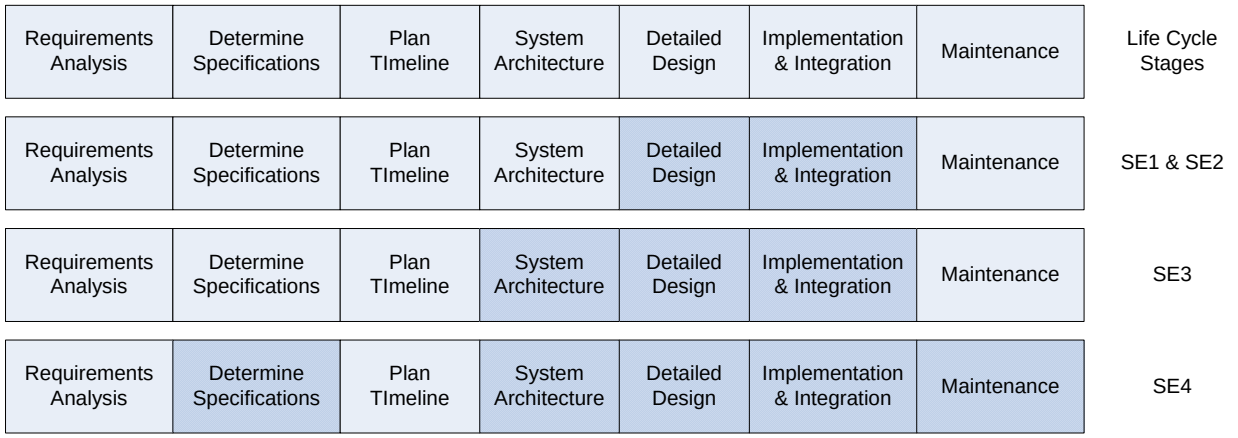

Figure 1: Focus on Activities in SE1 – SE4

As is evident from Figure 1, students gain experience with issues that arise and problems that must be solved as system specifications are determined, system architecture defined, modular functionality and inter-module interface specification defined, modules implemented and integrated with one another, and maintenance. They have not typically been responsible for requirements analysis and the subsequent system specification based on this requirements analysis and they have not typically been responsible for task assignment and the timeline planning required to deliver software on time. They have also not been asked to work in a team of size larger than two (and it is hard to call a "team" of two a real team).

Teamwork in the Design Sequence

Normally, when two or three students work together, they have chosen to work with one another and they believe they work well together. When the team size grows, as it must in any realistic setting, team composition becomes much more heterogeneous. It also becomes much more

difficult for groups of students to work together effectively to solve a problem. The number of different inter-personal interaction patterns is determined by the number of distinct groupings of individuals. Each distinct subgroup of individuals may form its own "personality" and interact with other individuals and subgroups differently. Thus, if the group is a group of three, there are six distinct sets of "communication/interaction patterns" (three pairwise patterns - person A with B, A with C, and B with C; and each individual with the remaining subgroup – person A with group BC, person B with group AC and person C with group AB). The number of potential interactions grows very quickly – a four person team will potentially involve on the order of 30 different communication/interaction patterns. Managing these effectively is critical if a team is to be productive.

In the first course in the design sequence (DES1), students work in large teams (8 to 15 students per team). This course is one that is required of all software engineering students and also of students pursuing a minor in software engineering. Students from multiple majors are involved in the course. Although most of the students are software engineering, computer engineering, and computer science students, students majoring in business and other engineering disciplines also take the course. The only restriction on enrollment is associated with students having acquired the appropriate prerequisite knowledge.

The course is structured around solution of a large problem that typically has a network component, a server component, a database component, and one or more client components. (As an example, one year the project involved construction of an internet chat room, instant messaging, and game modules.) As a large group, guided by the instructor, the class analyzes system requirements implied by the concept document that has been provided. The large team then constructs a specification document. Because their first effort at deriving a specification document usually results in a product that needs to be further refined, this is done in two or three iterations. Once a specification document has been produced, the group engages in planning so that the effort can be completed successfully. Since the entire course is conducted over a standard 15 week semester, the time line that is determined must be realistic and progress must be monitored carefully. The class learns how to do this under the guidance of their instructor.

Guiding students as they learn to analyze requirements as a group to determine the system specification is a delicate task. Very often, student groups find it difficult to stay on task. Instead of staying focused on the high level system structure, they tend to gravitate to the level of thinking about "how are we going to implement this". The instructor lets the group meetings proceed for a time, intervening when appropriate to bring the discussion back to the high level. He points out to the students that activity at this stage needs to focus on high-level module interactions rather than how to realize these interactions. The student teams also have a tendency to make systems overly complex. They are guided to consider minimal coupling between modules in a design mode that fosters simplicity of overall system structure. In this set of activities, the students learn by leadership example: their instructor, serving as team leader, brings them back to task as appropriate.

Once a plan is determined, the class engages in detailed high level design activities. Some of the tasks involved are distributed to smaller groups of students (sub-teams), and the resulting design document includes use case analysis, definition of high level modules and the interfaces among

them, sequence diagrams, transition diagrams, inter-module message formats, and detail down to the level of names for major functions, their inputs and their outputs. The design at this level of detail does not involve any implementation of code. The process includes iteration on the design so as to ensure that all of the interface issues, in particular, have been clearly and unambiguously specified. Once that is done well, the implementation and module integration phase is completed in short order.

It is in this process of determining a specification document, producing a viable system architecture with module structure and functional definition, allocating tasks to subteams, and implementing the overall system that teamwork skills are developed. Students first learn to communicate their ideas constructively. It is often the case that different individuals have differing internal perceptions. These diverse internal perceptions must be built into a collaborative, team-derived specification of the system and its functionality. There are a number of major barriers that students learn to navigate:

- *communication difficulties* – The instructor manages the group meetings so as to emphasize that everyone has the responsibility of contributing ideas and everyone has the responsibility of listening to and constructively considering the ideas of others
- *ego* – Students often have a "proprietary interest" in their own ideas, and this sense of ownership and association with the individual's sense of personal worth is a barrier. The instructor manages the group interactions with an attitude that conveys the message to each student that his or her idea is important to convey even if it is not the one that is eventually adopted. Ideas that are not adopted but whose consideration led the group to a better set of alternatives are highlighted as examples that illustrate the importance of constructive and open "egoless" communication.
- *working style* – By nature, it appears that individuals who work on software tend to develop something of a solitary working style. Many of the students seem to be "type A" individuals. Some work late at night and others find that their best working hours are in the morning. Working out the inter-personal issues that arise is done (when needed) with the aid of the instructor, using a philosophy and an approach that emphasizes the importance of helping each individual to make his or her best contributions. Every effort is made to direct the students away from an approach that attributes blame to any individual and towards a working philosophy that tries to elicit the best contribution from each individual.
- *differences in perspective* – Students in different disciplines come to a team with different perspectives and approaches to problem solving. Some will want to start a design at the low level, solving the important "nitty gritty" issues first. Others will prefer to start at the top, with development of a high level concept that then is refined as part of an evolutionary process. The instructor usually lets the students proceed for a time, allowing them to work out these differences on their own. He is, however, prepared to intervene and guide them in a specific direction when it becomes evident that they are not likely to succeed if they continue to proceed along lines that are potentially unproductive.

In DES1, the instructor serves as team leader for this large team, teaching them by engaging them in assigning individual and sub-team responsibilities, guiding them as they work out interpersonal and team disagreements in ways that bring out the best contributions from each

individual, and aiding them as they schedule milestones and delivery of the product. An example of what is needed in this aspect of the teamwork laboratory can be found in a scenario that is frequently observed. A subteam of three students working on implementation of a major system module was having considerable difficulty with team dynamics. Progress on their module was suffering because all three of the students (all very capable students) were taking responsibility for writing the code. Each of them felt that his way of doing things was the best, and the subteam was at a standstill. The instructor called a meeting of the subgroup and set a constructive tone at the outset. The discussion focused on identifying the particular strengths each member of the subteam brought to the table. The outcome of the meeting was a reallocation of effort that took advantage of each individual's strengths. The students learned to optimize the team activity by making sure that task assignments play to each individual's strengths. They may perhaps have learned better because they experienced the frustrations that arose from mis-aligned assignments first. At the end of this course, each student has learned how to work as a member of an effective team. Furthermore, they have had their first set of experiences that has involved activity across the entire spectrum of issues encountered in the software life cycle.

Software engineering is concerned with more than just effective participation in development of large software products. It is also concerned with managing teams in software development and managing the software development process itself. In DES1, the instructor provides a model that shows students what is needed to manage a software development team and engages the students in learning how to constructively manage the interpersonal and technical challenges that arise in team-based problem solution.

The second course, DES2, is one in which the students put what they have learned about working effectively on teams and managing the software development process into practice. The students are presented with a customer "wish list" and it is their responsibility to organize themselves as a team and work effectively to produce appropriate results. In consultation with their clients, the students determine exactly what functionality they will produce, plan and schedule their activities so that they attain their objectives on time, participate in design reviews, and keep themselves on track. The team is responsible for designating its own leadership (with some guidance) and is held accountable for maintaining its schedule. The work is done in two phases, with a mid term retrospective. This provides students with an opportunity for recognizing lessons learned and making mid-course corrections.

In the first week of DES2, the students are given a customer "wish list" that outlines what the product should do. One year, this wish list was generated by a group in an industrial organization and the desired product was a document handling system that should work across multiple platforms, be accessible by multiple arms of the organization, have appropriate security interlocks, and provide automatic "prodding" to ensure smooth progress of documents through internal approval channels. Another year, the wish list was generated by an occupational therapy group which articulated a need for cognitive rehabilitation software that is age-appropriate (since the age groups most often affected by head injuries are from late teen through late middle age and the existing software is adapted from educational software used by small children).

Students in DES2 work much more independently than those in DES1. The role of the instructor is that of a higher level manager – the students must manage their own efforts. A model similar to that outlined in the Team Software Process[2] is used as the students organize their own efforts. After a discussion of the roles team members play and responsibilities associated with each of them, the students first organize themselves and take responsibility for designating team leadership and allocating roles and responsibilities to individuals. Working with the client, the team analyzes the system requirements that are implied by the "wish list". In some cases, the underlying technologies are identified by the customer and in others they are not. The students must frequently interact with the customer as the needs are clarified. By the end of the second week of the term, the system requirements are identified and significant progress has been made in formulation of the specification document. This document specifies clearly what the functional specifications are, the characteristics of the user interface, the underlying technologies that are to be used. After the specification document is completed, a high level (system architecture level) design is formulated and a test plan that forms the basis for acceptance testing is generated. The high level design is usually completed about one quarter of the way through the semester. Throughout this process, the instructor's role has been somewhat "hands off". The instructor insists on continuing evidence of steady progress in the development of the specification document and high level design, but managing the process is the responsibility of the student team.

Once the specification document and high level design are completed, the team must plan its activities so that the product is produced in an appropriate time frame. The team creates a spreadsheet that details the high level design at a level that includes tasks required to implement the functionality of various components that have been identified in the system architecture. The spreadsheet also includes tasks that are concerned with learning about new technologies that are required for system development. Specific elements of the basic functionality are designated as a forming "basic functionality" product. The tasks associated with implementing this basic set of functionality are to be scheduled so that they are implemented shortly after mid-term. The idea is to get rapid deployment of a set of fundamental functionality then refine the product to incorporate additional functionality in a second phase of development.

Using the spreadsheet of tasks to produce functionality, the team attaches estimates of the time and effort involved for each of the identified tasks required to implement the basic functionality and the team produces a timeline of activities required to produce this basic functionality. Each student indicates a time allocation commitment that is consistent with what he or she is realistically willing to provide and also indicates areas in which he or she believes it is most appropriate to provide contributions. Subteams are formed to tackle tasks requiring more than one individual's effort and tasks are allocated to individuals on these subteams as well. As an example, the cognitive rehabilitation software required development of a back end database that would track each patient's activity with the system, development of user interfaces for the professional user and for the patient, development of a multiple activity modules (each of which was designed to exercise a particular cognitive function and allow increasingly complex interactions by the user), and a web interface so that users could access the system from home and from the clinic. Subteams were formed to address tasks associated with each of these system components, and responsibility for timely completion of these tasks was assigned to individuals on these subteams. As the timeline is developed, each individual in the class is assigned a set of

tasks to complete, and each of these tasks has a "due date" associated with it. It is the individual's responsibility to provide his or her deliverables on time. Some of the tasks are integration tasks that may be done by subgroups of students in the class. In order for the development to proceed smoothly, the students who are responsible for inter-related tasks must communicate with one another and must work cooperatively. It is the job of the team leadership to facilitate this process, and it is in doing this that the class puts what has been learned in DES1 into practice. The instructor has the job of monitoring the class activities, stepping in whenever absolutely needed to facilitate, and resolving any problems that the students seem unable to resolve without aid.

A key aspect of DES2 is the phase retrospective that is held just after midterm, at the conclusion of the rapid development of basic functionality. At this point, the team looks back over the events that have transpired and asks itself some fundamental questions. Among these are questions about team working effectiveness, problems that have been encountered, strategies for improving productivity, individual contributions and how they can be improved, and planning for implementation of the remaining functionality. A form (shown in Figure 2) adapted from the PEER form of the Team Software Process[2] is completed by each student. Although the comments regarding individual students and who made which suggestions are kept confidential, the results are used to guide the team retrospective discussion. Sometimes the team reallocates responsibilities among team members. Estimates of the amount of effort required on some set of tasks are frequently low, so the effort may be scaled back. More manpower is frequently added to tasks that were harder than anticipated. Differing subteam composition may be arranged, based on lessons learned about degrees of individual expertise and compatibility of working styles. The balance of the semester is spent completing deployment of the system that was specified and producing system documentation.

Lessons Learned

In this paper we have described how strategies for guiding students in their acquisition of teamwork skills can be integrated in their design experiences. We have noted that students must develop a number of attributes and skills in order to work effectively on the multidisciplinary teams that are involved in solving modern engineering design problems. The two course design sequence in software engineering serves as a laboratory in building team management skills. As part of this design sequence, students are engaged in a range of activities across the spectrum of phases found in the software life cycle. As they carry out the tasks in these design activities, they also learn:

- how to work within the dynamic of a large (more than 6) group of people with diverse backgrounds (in DES1 with a group of 8 to 15 other students guided by an experienced instructor, and in DES2 as a team led by students comprised of the whole class with subteams that are student-formulated and led by student colleagues)
- that a personal commitment to contributing one's own share is critical to team success (as they meet – and fail to meet – personal commitments recorded on timelines agreed upon by the team in DES2)

- that each team member must stay focused on the common goal and on his or her own assigned tasks (in DES1 as the group collectively engages in design and implementation activities and in DES2 as the team guides itself through the process)

1. Rate the overall ***team*** in each of the following categories (give a number from 1 (inadequate) to 5 (superior) in each category:
    a. Team spirit
    b. Overall effectiveness
    c. Rewarding experience
    d. Team productivity
    e. Process quality
    f. Product quality

2. Rate ***each individual*** for overall contribution (again giving a number from 1 (inadequate) to 5 (superior) for each individual).
    a. Student 1 _____
    b. Student 2 _____
    c. Student 3 _____
    d. Student 4 _____
    e. Student 5 _____

3. Rate ***each individual*** for helpfulness and support (again giving a number from 1 (inadequate) to 5 (superior) for each individual).
    a. Student 1 _____
    b. Student 2 _____
    c. Student 3 _____
    d. Student 4 _____
    e. Student 5 _____

4. Please give your best estimate of the fraction of the overall effort that was done by:
    a. Student 1 _____
    b. Student 2 _____
    c. Student 3 _____
    d. Student 4 _____
    e. Student 5 _____

5. Any constructive suggestions that you wish to make regarding ways in which better teamwork can be fostered would be very much appreciated.

Figure 2: Peer and Team Evaluation Form

- that once a collaborative team decision has been made and a plan for the team formulated, everyone on the team must support the outcomes (in DES2 this becomes particularly evident in the midterm retrospective)

- that each individual's thoughts and ideas must be respected and valued as the collaborative effort develops (in DES1 as the instructor engages the group in brainstorming and as the students direct themselves in requirements analysis and system specification in DES2)
- that it takes "give and take" to arrive at the best solutions and that confusion and uncertainty are clarified through questioning and discussion (in both DES1 and DES2 as specifications are clarified and in DES2 as interface specifications and implementation strategies are determined by the student-led group)
- that differences and conflicts occur, that they must be acknowledged, and the best way to deal with them is constructively and openly (throughout all of the group interactions – both instructor-facilitated and student-led in both courses).

When students are asked, at the end of DES2, what they would have done differently, the most frequent answer is one that centers on the importance of staying on task, communicating with one another better, allocating back-up responsibility for tasks, asking for help sooner, and not procrastinating. That these are the most frequent responses is probably indicative of the learning that has occurred. There is, after all, no better teacher than experience.

References

1. National Academy of Engineering, *The Engineer of 2020: Visions of Engineering in the New Century*, (2004), p. 42.
2. Humphrey, Watts S., *Introduction to the Team Software Process*, Addison-Wesley, 2000.

Biographical Information

SUSAN E. CONRY
Dr. Conry is a faculty member in the Department of Electrical and Computer Engineering at Clarkson University. She obtained her B.A., M.S., and Ph.D. degrees at Rice University. Her interests include engineering education, multiagent systems, and parallel and distributed systems. Dr. Conry teaches in a variety of areas in computer engineering and software engineering.

DOUGLAS J. MACINTOSH
Dr. MacIntosh is a faculty member in the Department of Electrical and Computer Engineering at Clarkson University. He obtained his B.A., M.S., and Ph.D. degrees at Clarkson University and has spent a decade engaged in software engineering in the industrial sector. He serves as Director of the ABET accredited Software Engineering program at Clarkson University.