A Framework for Interaction in Software Development Training

Rose F. Gamble & Leigh A. Davis
University of Tulsa, Tulsa, OK, USA

gamble@utulsa.edu davisl@utulsa.edu

Executive Summary

Complex application development should be an integral part of a capstone software engineering course for undergraduates. In the same respect, graduate students in computer science should have practical knowledge of project management. In this paper, we introduce FIST (a **F**ramework for **I**nteraction in **S**oftware development **T**raining), which allows students to experience real-world project structures and goals. FIST was devised to establish a corporate-style interaction between the instructor, managers, software project teams, and customers.

There are multiple courses involved in FIST. The first course is a combined senior-level/graduate course in software design and specification. In this fall semester course, instruction is given on the Unified Process and the inception phase of the project is begun. In the spring semester, senior computer science students take the capstone software engineering projects course, while graduate students enroll in project management. FIST structures these two courses in an innovative manner. Namely, graduate students perform all the duties of middle and upper management for the senior projects.

The main objectives of the framework are to increase productivity, create a realistic business atmosphere, and gain needed experience in development and the application of studied management techniques. These objectives are accomplished through multiple mechanisms. First, we assign explicit roles to all participants and structure the projects, lectures, and assignments across the Unified Process. The classes meet separately so that focus on the particular course instruction can be performed. However, external team meetings are required, as well as multiple customer meetings and reviews throughout the semester. Students evaluate themselves and their team during the semester to catch technical and personnel problems early. Management graduate students use learned techniques to handle problems, as well as develop and review artifacts, such as schedules, budgets, and design and test documents. They keep journals regarding their management and the impacts of their decision on the projects. Senior development teams report to management via progress reports and technical documents. The teams maintain a certain level autonomy within the project goals set by the instructor and customer.

FIST works very well. The development teams have produced higher quality and increasingly complex software products that include virtual reality systems, games, and PDA applications. There is better communication among team members fostered by the general respect the undergraduates have for the graduate managers and their part in the project. Artifact documentation is more in-depth and much improved as FIST allows for more iteration before the instructor and customer view it. There is continual feedback throughout the process that contributes to adapting it to particular projects, team structures, and technical issues.

Keywords: Software engineering curriculum,

project management, capstone project course, industrial experience, multi-course interaction.

Introduction

"What do you mean we have to document everything?" - a project team to their management

Recent articles on software engineering education stress the importance of real-world experience (McCauley & Jackson, 1999). It is agreed that this experience can be gained from participating in projects that are complex with potentially marketable results (Nachbar, 1998). A customer, that is not the instructor, should be involved in the development process (Polack-Wahl, 1999). Unfortunately, time constraints often make the realization of meaningful projects difficult.

Within a software projects course, an established technique is to maintain the instructor as direct manager over all of the projects. In this model, four issues make relevant projects difficult to use (Bryant, 1999): (1) instructor overload, (2) the balance of theory with practice, (3) evaluation of student performance, and (4) keeping students motivated. Sacrificing depth and quality in code and documentation can provide a more manageable workload and help to address these four issues. However, the undergraduates then lack the experience (and the frustration) that is part of industrial-scale development.

Different frameworks have been devised for software engineering education – both at the graduate and undergraduate levels. In many case, the frameworks are tailored to the curriculum and the degree of focus on software engineering, e.g., the number of software engineering courses and whether a degree is software engineering is offered. Cowling presents a hierarchical curriculum framework based on process, products, and people (Cowling, 1994). The framework serves as a guideline for establishing software engineering courses specific to the computer science curriculum. While he suggests there be a balance between the three parts, he recognizes that products are the main focus, and thus, that branch of the hierarchy requires more depth and attention.

Laboratory concepts have been extended to software engineering (Sebern, 2002). One approach is to use a laboratory to have traditional project-based training, while giving students "staff" assignments, like Quality Assurance Group or Configuration Management Group, to foster cross team sharing of information and experience. Similar to lab experiment reports, students provide feedback through "process improvement proposals" and "advice to future students" as project hand-ins. This provides a mechanism for self-reflection and expression.

Graduate students in software project management classes are often taught using case studies (Kemerer, 1997). Students assess the managerial problems and detail potential project directions in written briefs. Sample projects for architectural descriptions, configuration management, and development tool usage may also be included. Missing from this curriculum is actual project scheduling and assessment, managing of personnel, and communicating with a realm of stakeholders. One clever approach to partially overcome these problems uses assignments that focus on a particular software project in which the "managers" formulate project artifacts (schedules, test plans, etc.) given an imaginary team that performs the development (Murphy, 1999).

Programs for Master's in Software Engineering have greatly expanded over the past decade. Most encompass multiple courses that are focused on obtaining industrial foundation for problems, teaching methods, and projects. Some go beyond the typical academic bounds by having industry partners contribute to master's theses by providing students with relevant problems and participating on the committee that judges the outcome of the thesis (Wohlin & Regnell, 1999). This provides a genuine problem to accompany classroom topics.

An interesting concept called the "Virtual Enterprise" strengthens a project-based approach for the Master's in Software Engineering degree (Herwig, 1997). The concept requires seven semesters with at least

one software engineering course, along with one semester in which an industrial internship is required. Students are taught foundations of software engineering early in their academic career and then given a small, probationary project on which to test their skills. Errors are allowed (and expected) in order for the students to be able to tackle a larger, more complex project. Project management is one of the courses taught during the 8 semesters, but it is not associated with any particular project.

In this paper, we discuss a framework, FIST, that provides project interaction between an undergraduate software projects course and a graduate course in software project management. FIST can be tailored to different course sizes, different projects, and different customer expectations. We outline the goals and structure of FIST for each class, discuss the most notable benefits of the interaction, and provide some examples of implemented projects.

Framework Goals

"Do customers know less than they let on?" – a project team member after a customer meeting

FIST was devised to establish a corporate-style interaction between the instructor, managers, software project teams, and customers. All stakeholders share in the responsibility of delivering a satisfactorily completed project with sufficient complexity and depth. This requires predefined and well-understood roles and responsibilities. Good communication among the project team and the customers, as well as process and tool understanding are paramount.

FIST strives for scalability in several areas.

- Project development scenario scalability to industrial software engineering

- Project management environment scalability to real-word experiences

- Framework scalability to variable class issues

To attain project scalability, development teams need to work with real customers and users. The software developed should include multiple components, some of which may be commercial-off-the-shelf (COTS) or open source products. Management students should bear the responsibility of a project, including scheduling and task assignments of personnel. Moreover, they need to have a certain degree of autonomy, similar to that found in industry at a middle management level. Finally, as a framework, FIST should accommodate variable class sizes, different project domains, and diverse academic expectations.

To achieve the desired scalability, one objective is to reduce the instructor's direct involvement in the projects, yet maintain a layer of support. This leads to valuable interplay among the teams, the middle managers, and their customers, in which the managers can catch problems early and relay expectations up and down the "corporate" hierarchy. As part of this support system, solid project completion and delivery along with improved documentation quality become goals.

Another objective within FIST is to maintain some autonomy across courses. Not all students will perform at a high level, and thus, a failed project might not be the fault of the graduate student managers. On the other hand, not all graduate students are good managers, and thus, a team sometimes has to complete their project without the expected support.

Realizing FIST

"How do I get past the blank looks from my project team?" - a manager after a meeting on UML modeling

In the Department of Mathematical and Computer Sciences at The University of Tulsa, software engineering is an integral part of the curriculum from undergraduate instruction to Ph.D. research. Thus,

theoretical and practical training in all aspects of software development is a must. In addition, TU is a growing institution whose curriculum must accommodate an increasing population of industry-minded students without sacrificing the quality of the education.

Team Formation and Instruction

Based on resumes along with project and position preferences, 4-5 students comprise an undergraduate project team. Each team contains a project lead, a senior programmer, and programmer analysts. Depending on the number of customers, some teams have unique projects while others have duplicate projects. For the first half of the semester, lectures on the Unified Process and the Rational™ Software Developers Suite are given while teams meet at least once a week to begin design. The second half of the semester allows the students concentrated development time, with no regular class meetings scheduled.

The graduate project management class is organized differently. Almost immediately, managers are chosen to direct a single project team. Depending on the relative class sizes, a hierarchy is formed in which middle management reports to directors, also to students from the class. Directors report to the instructor. In the case of a small management class, only one level of middle management is used.
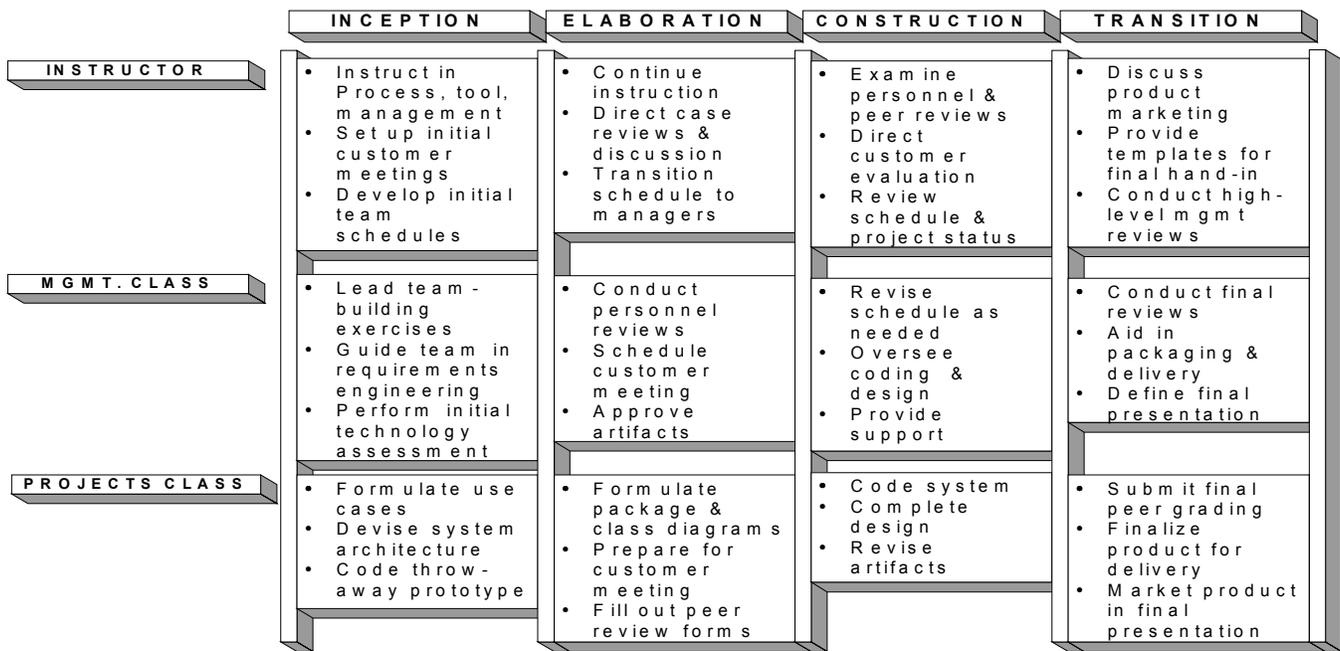
	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
INSTRUCTOR	• Instruct in Process, tool, management • Set up initial customer meetings • Develop initial team schedules	• Continue instruction • Direct case reviews & discussion • Transition schedule to managers	• Examine personnel & peer reviews • Direct customer evaluation • Review schedule & project status	• Discuss product marketing • Provide templates for final hand-in • Conduct high-level mgmt reviews
MGMT. CLASS	• Lead team-building exercises • Guide team in requirements engineering • Perform initial technology assessment	• Conduct personnel reviews • Schedule customer meeting • Approve artifacts	• Revise schedule as needed • Oversee coding & design • Provide support	• Conduct final reviews • Aid in packaging & delivery • Define final presentation
PROJECTS CLASS	• Formulate use cases • Devise system architecture • Code throw-away prototype	• Formulate package & class diagrams • Prepare for customer meeting • Fill out peer review forms	• Code system • Complete design • Revise artifacts	• Submit final peer grading • Finalize product for delivery • Market product in final presentation

Figure 1: Stratified Responsibilities

Students are usually paired in a management team. The pairing is based on their assets and preferences. This structure allows for a mix of communication strengths and technical expertise, while accepting the general inexperience of the graduate students in management. The purpose of the pairing is to make management easier. For example, if one manager is uncomfortable in addressing a particular problem, it is hoped that the other manager can do it. However, they are duly instructed to provide a single voice or unified front to the team, such that all decisions require the collaboration of both managers. Moreover, sharing responsibilities within the management team lessens the heavy time commitment.

The Unified Process is being used in software engineering courses in various ways because it has many characteristics that make is amenable to an undergraduate software engineering course. Robilliard, et al (2001) scaled down the process for a "short-course" approach that would fit in a yearlong software engineering project course. The goal was to maintain a balance between learning the process and applying it. Using a sequence of probationary projects leading up to a large industrial project, (Halling et al, 2002)

also incorporates the fundamental concepts of the Unified Process – again across a yearlong course. This course combines lectures and workshop tasks that introduce and apply the four phases of the Unified Process: inception, elaboration, construction, and transition (Figure 1).

Using the Unified Process phases, we highlight some of the distinct responsibilities among the instructor, management, and project teams in Figure 1. The purpose of the diagram is not only to show division of labor, but also to indicate how well the Unified Process structures and integrates with FIST. Because the Unified Process is iterative and incremental, there is some overlap present between phases (Cantor, 1998). For example, the instructor is handling the schedule throughout the Inception and Construction phases while transitioning it to the managers. Managers conduct reviews over multiple phases, as well as oversee packaging. The team, meanwhile, has the responsibility of documenting the design and coding the product throughout all of the phases.

Communication

All project team members submit weekly progress reports to the project lead. These reports move up the hierarchy, allowing all levels of management to be aware of both present achievements and future work. Each report comments on the state of the project, current risks, action items, along with concerns and experiences of the team and management.

Weekly meetings are scheduled between the project teams and their management. This reinforces the need for communication as well as adapting the roles of the teams and management to the state of the project. The instructor meets regularly with the highest level of management to review project status. However, this does not restrict students from coming directly to the instructor at any time.

Graduate students are responsible for communicating customer expectations, instructor expectations, and the initial schedule to their project teams. Moreover, middle managers must acquire deeper knowledge of the software development tool and the Unified Modeling Language, so that they can tutor the project teams. This responsibility allows the graduate students some expertise over a project area while they acclimate to their management roles.

Customer Relations

Customers are an extremely valuable part of software development training. Customers are often from local industries: some are alumni, eager to participate. Meetings are usually scheduled three to five times during the semester. The customer initially presents the project vision. After this presentation, all stakeholders meet to scope the requirements to realistically fit the time and effort constraints of the semester. Subsequent meetings involve team presentations to the customer. Formal agendas are published, and the instructor and customer evaluate each presentation. The project culminates in a marketing presentation where the teams "sell" their product to faculty, staff, students, and all participating customers.

The instructor retains all logistical duties of customer interaction – meeting scheduling, product evaluation, and product delivery. This frees the customer from being bombarded with unnecessary inquiries, while giving the students time and distance (especially from a temperamental customer) to resolve issues, and to understand requirements and parameters.

Assessment

Student assessment comes in many forms. Project materials, documentation, presentations, and case studies make up a large portion of the assessment. The customer grades the team based on the ongoing involvement and the final artifact transitioned. There are also other individual factors that constitute a student's grade.

Peer Reviews

Personnel reviews are conducted twice during the semester. For these reviews, students fill out a general self-assessment form detailing their responsibilities and how well they performed at them. The forms are submitted to the next level of management in the hierarchy. All levels of management meet individually with students under them (e.g. project teams or other managers) and perform the review. All documentation from the reviews is given to the instructor, who evaluates the quality of the review.

This type of assessment can be difficult for students on both ends of the review, especially if the student is viewed (by his/herself or by the reviewer) as not performing adequately. However, review is an integral aspect to any position. Therefore, the process requires the students to qualify their effort and productivity, while reviewers are forced to be objective, examining results, not personality.

Students also evaluate their managers and other team members at least twice during the semester. Qualities for assessment include motivation, trustworthiness, creativity, project knowledge, commitment, and professionalism. These assessments especially empower the undergraduate students to fully detail the quality of people they work with. The evaluations provide the instructor insight into team and management dynamics, particularly when intervention is necessary to lessen team problems.

Case Write-ups

Improving management skills often requires objective self-assessment (Drucker, 1999). One way incorporated into FIST is that each individual graduate student writes a business case study. The case is written in the third person, describing his or her project as it progressed through the semester. Students highlight key decisions and their impact on the project. They note both successes and failures (large and small). Moreover, they discuss personality, competence, technology, and communication issues that influenced the outcome of the project. Progress reports and personal journals add depth to the cases.

This self-reflection often provides graduate students with much needed insight about how to be a better manager. It allows them to judge decisions made by the upper management (including the instructor), as well as decisions they made during project development that may have impacted their performance. It also helps them put the project in perspective, comparing their styles to those described in the reviewed cases.

Instructor Evaluation

Documentation, product demonstration, project presentation, and customer satisfaction are evaluated for the team grade. Project teams are provided with detailed templates on which to organize their artifacts for evaluation. Usually, a documentation draft is required a few weeks prior to finals. This eases some of the stress around the end of the semester when the marketing presentation and product transition are primary concerns. Moreover, it gives the team a chance to correct errors and add missing information. A team member can achieve a grade lower (by using peer grading as an assessment tool) than the team's grade, but never higher. This solidifies the concept that in software development, it is the results that are important, not necessarily the effort one puts in.

Graduate students are graded on their various assignments within a traditional class setting, along with their attempts at applying management techniques. The project grade has less weight than for the undergraduate teams, because some teams perform well, despite poor management and some teams are impossible to manage well.

Experiences and Anecdotes

"When I needed personnel advice, I went to one manager and when I needed coding advice I went to another." - a project lead about her management team

FIST has been used multiple semesters with varying class sizes, project complexities, and customers. A single instructor is responsible for implementing FIST. This allows for continuity of the approach, as well as experience in solidifying the framework concepts. FIST has worked very well to achieve the objectives of realistic experience without overburdening the students, customer, and instructor. We can accomplish in FIST whole projects that required multiple years of the senior software engineering course. In addition, the project management class actually manages.

One might think that undergraduates would resent being "managed" by graduate students – many with little experience. In contrast, the undergraduate students are generally positive about working with graduate student managers. While the instructor maintains a definitive presence in the infrastructure, the students feel a sense of autonomy and responsibility for their projects. This translates to deeper accomplishments and better artifacts. Undergraduates have noted a great sense of pride in what they deliver through the management chain. It is believed that this is due to peer relations, rather than the typical instructor-student relationship.

Undergraduates do experience some difficulties when managers are irresponsible, non-communicative, or do not understand requirements from either the instructor or the customer. Fortunately, this usually occurs with only one manager of the graduate team. Though the more competent manager may carry a heavier load, the project teams can still move forward. When the management class is large enough, directors are available to work with problem managers and either set him/her on a better path or support the project themselves.

Graduate students are themselves under pressure for their projects to succeed. Achieving "buy-in" for their decisions is a difficult task for some. They are put in a situation where they need to earn respect from the group in order to influence the direction of the projects. In many cases, this translates into learning how to motivate the undergraduates. Many techniques are tried. The results of their application are written in a journal for objective consideration.

For example, some project team members are absent for long periods of time. Some members of the team clash or are viewed as having insufficient experience. In these circumstances, managers have to present the problem and a resolution plan to the instructor. Then this plan is acted upon and the results documented. Through this approach, students gain valuable experience in scheduling, architecture, and personnel management. Instead of learning by cases only, they learn by doing.

One software project was in association with a local rehabilitation facility that provided access to a virtual reality helmet. In turn, the university contributed the software to program the helmet for stroke patients with "left-hemisphere neglect." The students met with doctors, therapists, and patients to form requirements. Two teams were involved with the project: one for the development of the VR world and another to track and analyze patient performance. The project challenged the students to interface between different software paradigms and to learn a new development tool for the helmet.

In this particular project, management caught a major error during the construction phase coding. Worse, it was discovered that the senior programmer on the team knew about the problem and had hidden it. To save the project, management worked side-by-side with the team to fix the problem so that its VR world operated properly and then, to update all of the documentation. The rehabilitation doctors thought the project was such a success, that they hired one of the team members to help with clinical trials.

Another project involved developing an e-commerce prototype to sell telecommunication services, rather than products. An off-the-shelf e-commerce package had to be integrated with an application server for dynamic updating of existing and new services. This project had high potential for direct industrial application. However, one team member, perceived to be the best programmer, refused to follow the process, listen to management directions, or cooperate with the team. The student was repeatedly absent from team meetings, only to appear for coding the night before a customer review. Each review was substandard, lowering morale. To make matters worse, one of the managers had many of the same opinions as the errant programmer. Thus, there was a rift within management, as well as the team.

The director over the managers applied pressure, was motivational, and suggested the team take steps to fire the employee and take over his assignments. The team and their managers refused. Sadly, the customers were not pleased with the outcome of the project. However, the self and team evaluations, as well as the management case studies revealed the students had learned a valuable lesson, that over-dependence in a person's abilities is detrimental.

Most recently, a team developed a unique, customizable computer game. The idea originated with a professor's child. Parents of similar-aged children were used as customers. This project had some technology difficulties but the team was so respectful of each other and so motivated that they worked hard to overcome them. The manager commented on how he learned that a good team should not be interfered with, but kept on schedule and made aware of expectations. The product developed by this team was such a success that students and faculty requested copies for their children to play at home.

Discussion of the Approach

"I didn't know that managing people required the writing and reading of so many emails." - a manager at the end of the semester

Though a semester affords limited time, the projects using FIST achieve far more complexity and depth than in those semesters in which only undergraduates participated. One reason for this is that the structure of the framework allows for faster and more detailed project feedback between the managers and teams. The instructor is no longer the bottleneck for approval and progress of all the teams. Another reason is that team members and managers take more responsibility for the project outcome.

The intertwining approach is a way to practice communication and negotiation techniques at all levels. Serious students will always have to work with less motivated people. But, within FIST, manager support makes the communication of roles and responsibilities clearer, including where those responsibilities are not being met. Negotiation can also take place with less intimidation than a student might feel interacting with the instructor. Class discussions about management allow graduate students to help each other assess their actions and try new strategies to improve team production. Weekly status reports highlight difficulties where instructor intervention is needed.

Documentation is of a higher quality than was seen without FIST in place. This increase has some basis in peer review. All documents must be assessed up the management hierarchy, instituting changes as they proceeds to the instructor. Thus, there is more iteration over documentation prior to the customer and instructor viewing it. The project teams have consistently shown more pride in the professional quality of their work. For example, one student commented that because the team managers were also students, they did not want them wasting their time with poor quality documents.

Students have given very positive feedback from participating in FIST. Consistently it is highly ranked in the survey sent to seniors as the most important course that CS students take. Graduate student communication skills, both oral and written, greatly improve during the semester. The case write-ups are both insightful and entertaining. They find that even if the project is unsuccessful, they have a worth-

while experience and have learned a lot about their management style. For international students, this translates into increased confidence in the job market.

Customers, as well as faculty that attend the final presentations, are pleasantly surprised at the products that are produced and the professionalism of the students. However, the success of FIST is most noticeable in that undergraduate students no longer fear and dread taking the senior software projects class in which many difficult hours were needed to complete a project that was far less than they envisioned. Now students look forward to coming into the class and developing software that demonstrates their capabilities as future software engineers.

Conclusion

"I have learned that email is not a very good communication medium. While it is fast and convenient, it is too easy to misunderstand the tone or spirit of a message – especially in the face of a deadline." – a director after a poor showing at a customer review

In this paper, we discuss a novel approach to software engineering and training education called FIST. An undergraduate, capstone, software engineering class is enhanced by the inclusion of middle and upper management from a graduate project management class. The framework for interaction is now a stable part of the CS curriculum at a major university where it continues to be tested and updated.

Future instances of the framework will allow more manager input in team make-up and individual responsibilities, the use of extreme programming (Beck, 1999), and experiments with lay-offs and team reorganization. We will incorporate more evaluation mechanisms into FIST that allow for different processes and process improvement. Moreover, we plan to assess student performance with FIST as compared to other CS courses, to see if the interaction correlates with better grades.

References

Beck, K. (1999). *Extreme Programming Explained*. Addison-Wesley.

Bryant, R. (1999). Software engineering for seniors – overcoming the administrative fears. *SIGCSE Bulletin*. March, pp 83 - 86.

Cantor, M. (1998). *Object-Oriented Project Management with UML.,* John Wiley & Sons.

Cowling, A. (1994). A framework for developing the software engineering curriculum. *International Workshop on Software Engineering Education*, Sorrento, Italy, pp. 111 - 118.

Drucker, P. (1999). *Management Challenges for the 21st Century*. HarperCollins.

Halling, M., Zuser, W., Köhle, M., & Biffl, S. (2002). Teaching the Unified Process to undergraduate students. *15th Conference on Software Engineering Education and Training,* February, pp. 148 - 159.

Herwig, M. (1997). Teaching Software Engineering by Means of a "Virtual Enterprise." *10th Conference on Software Engineering Education and Training,* Virginia Beach, VA, pp. 176 - 184. .

Kemerer, C. (1997) *Software Project Management*, Irwin McGraw-Hill.

McCauley, R. & Jackson, U. (1999). Teaching software engineering early. *SIGCSE Bulletin*, June, pp. 86 - 91.

Murphy, M. (1999). Teaching software project management: A response-interaction approach. *12th Conference on Software Engineering Education and Training*, New Orleans, LA, pp. 26 - 33.

Nachbar, D. (1998). Bringing real-world software development into the classroom. *SIGCSE Bulletin*. March, pp. 171 - 175.

Polack-Wahl, J.A., (1999). Incorporating the client's role in a software engineering course. *SIGCSE Bulletin*, March, pp. 73 - 77.

Robillard, P.; Kruchten, P.; & d'Astous, P. (2001). Yoopeedoo (UPEDU): a process for teaching software process. *14th Conference on Software Engineering Education and Training*, Charlotte, NC, pp. pp. 18 - 26.

Sebern, M. (2002)., The software development laboratory: Incorporating industrial practice in an academic environment., *15th Conference on Software Engineering Education and Training,* Covington, KY, pp. 118 - 127.

Wohlin, C. & Regnell, B. (1999). Achieving industrial relevance in software engineering education. *12th Conference on Software Engineering Education and Training*, New Orleans, LA, pp. 16 - 25.

Biographies

Rose F. Gamble received her B.S. in Mathematics and Computer Science at Westminster College, New Wilmington PA. She received an M.S. and D.Sc in Computer Science at Washington University, St. Louis MO. She is currently an Associate Professor of Computer Science at the University of Tulsa, where she directs the software engineering curriculum for the Department of Mathematical and Computer Sciences. She is the Director of the Software Engineering and Architecture team (http://www.seat.utulsa.edu). Her current research interests are in software architecture and component integration.

Leigh A. Davis attended New York University where she received a Bachelor of Fine Arts in Film Production. Being a great lover of film, but not a financially successful filmmaker, she sought a challenging and creative field of study for her master's education, namely computer science. She is currently a PhD candidate at The University of Tulsa in Computer Science. Her current research interests include software architecture characteristics, dynamic composability, and secure component integration.