122nd ASEE Annual Conference & Exposition
June 14 - 17, 2015
Seattle, WA
Paper ID #13557

Introducing Software Specifications to an Undergraduate Software Engineering Program

Dr. Anna Koufakou, Florida Gulf Coast University

Dr. Koufakou is an Assistant Professor in Software Engineering in the U.A. Whitaker College of Engineering at Florida Gulf Coast University. Dr. Koufakou received a B.Sc. in Computer Informatics at the Athens University of Economics and Business in Athens, Greece, and a M.Sc. and a Ph.D. in Computer Engineering at the University of Central Florida. Her research interests include mining of large datasets, outlier detection, and frequent itemset mining. Educational areas of interest are promoting student engagement via techniques such as hybrid teaching, flipped classroom and problem-based learning.

Introducing Software Specifications to an Undergraduate Software Engineering Program

Introduction

The complexities of developing clear and well-defined specifications and their important role in the success of a software project are widely recognized. This recently led to increased attention in corresponding courses in the Software Engineering curriculum. One of the challenges for such courses is that related topics are typically perceived by students especially at the undergraduate level as uninteresting and irrelevant, while it is difficult to bring the "real-world" experience to the classroom.

This paper summarizes the author's experiences in developing and teaching for the first time a Software Specifications course to the newly established Software Engineering (SE) degree program at Florida Gulf Coast University (FGCU) in Spring 2014. The SE program belongs in an Engineering College which emphasizes undergraduate education (there is no graduate degree offered). The Software Specifications course is a required course for all SE students at FGCU. It includes topics such as Eliciting, Writing, and Testing Requirements, and Requirement Engineering Tools, as well as Unified Modeling Language (UML), Agile Methodologies, and SE Ethics. The SE student population consists mostly of traditional college students typically from the surrounding regions. As the school does not offer a graduate degree in any related majors, teaching assistants are also undergraduate students. In this setting, it is especially important to capture the interest of the students and engage them in the class, as well as offer students practical knowledge and tools that they can apply in their projects and potentially in their future career.

With this in mind, the course employed a variety of techniques including hybrid (blended) course delivery, and client interaction for student group projects. The first implementation of this course achieved positive student feedback and performance in the class. Results and lessons learned are also discussed in the paper.

Literature Review

Despite the importance of Requirements Engineering (RE) in Software Engineering, RE is not emphasized in computing education. In fact, most computer science and software engineering programs do not include RE courses and tend to cover this area using a few class periods[1,2]. Additionally, topics and careers related to RE, and subsequently a related course, are perceived as uninteresting and not relevant to future career prospects[3, 4]. This is described very well by (Lethbridge et al.)[5] as follows:

> "*Anyone who has tried to teach topics such as ethics, quality, process, configuration management, maintenance and requirements will recognize the glassy-eyed appearance in the eyes of some (or most) students. These are critical topics for industrial practice, yet it is a particular challenge to motivate students to feel passionate in these areas, and hence learn what they need to know*".

Given the limited resources in many undergraduate programs, it becomes a challenge to develop a course that would provide students the skills needed to successfully conduct RE activities.

A variety of methods and ideas are presented in the literature regarding teaching RE and development of RE-related courses. Callele and Makaroff [6] introduce the teaching of RE to students of an introductory programming course (an "unsuspecting audience"). They also argue that students are not able to effectively simulate stakeholders due to their lack of domain knowledge. Suri[4] describes two offerings of an RE course, where the students work towards producing a Software Requirements Specification (SRS) document. There has also been effort to introduce role-playing in order to involve students actively in requirements elicitation[7, 8]. Mohan and Chenoweth[9] present a three-tier learning system where students produce use cases, supplemental specifications, and a user interface, among other deliverables. More recently, Kilcay-Ergin and Laplante[10] describe an asynchronous online RE course geared towards graduate students. These students are described as professionals who are mature and responsible to "self-direct their learning according to their individual learning style and pace" [10].

Course Description

The Software Specifications course (CEN 3073) presented in this paper is a 3-credit hour undergraduate course without a laboratory component. The CEN 3073 course is taught in the second semester of the junior year, after an 'SE Fundamentals' course in the previous semester, and before 'Software Architecture & Design' and 'Software Testing' courses in the following semesters. Students arrive at this course with intermediate knowledge of programming and experience developing a group software project (from the SE Fundamentals course in the previous semester). The course objectives are listed below:

- Learn techniques, processes, and challenges involved in requirements elicitation, analysis, validation and management;
- Be able to work in a group environment to perform activities and provide documentation related to all phases of software development, focusing on requirements analysis, while also carrying out design, implementation, and testing;
- Be able to construct UML diagrams (e.g. Sequence Diagram, State Diagram);
- Conduct an oral presentation and demonstration of a software project developed by a group.

Table 1 shows a general outline of course topics. For example, "Writing Requirements" includes Formal/Informal/Semi-Formal techniques, SRS standards and recommended practices, and Use Cases, while "Testing Requirements" includes Requirements Validation/Verification (V&V) techniques, NASA Requirements Testing, and deriving test cases from requirements/use cases.

Table 1. List of Course Topics

#	**Topic**	**Timeline**
1	Introduction to Requirements Engineering (RE)	Week 1
2	Preparing for Requirement Elicitation: Stakeholders, Project Scheduling	Week 2
3	Eliciting Requirements	Week 3
4	Unified Modeling Language (UML)	Week 4-5
5	Writing Requirements	Week 6
6	Testing Requirements	Week 7-8
7	RE Tools	Week 10-11
8	Risk Analysis	Week 11
9	RE & Agile Methodologies	Week 12
10	SE Ethics and Professionalism	Week 14-15

Team Project: An important part of the course is that it incorporates a collaborative learning component, in the form of a semester-long team project. In particular, student teams with 3-4 members interact with a client to develop software according to the client's specifications. During the semester, teams report on client meetings and present their progress, including prototypes, to the class. Additionally, each student submits their individual meeting preparation, including text, mockups, etc. Final team deliverables include fully developed application code plus documentation, such as SRS (Software Requirement Specification) and user manual, followed by a team presentation of the final product.

Delivering a full software product is in contrast to similar courses found in the literature where student teams produce an SRS or similar deliverable[4]. Even though this presents possible extra work for the students, the goal is to increase student motivation: students in the software field are most interested in working towards a functioning end-product rather than documentation or prototypes. Furthermore, this gives them the experience of how prototypes and client feedback translate into the finished product. This was also observed by Mohan and Chenoweth [9]: it is important that students "carry their requirements projects forward into design and development, sufficiently that they can see the importance of the time spent learning to do requirements" [9]. Nevertheless, it is important to note that student teams spend the larger part of the semester (about 70%) interacting with the client, building mockup/prototype(s) and obtaining client feedback, which they then use to create their SRS (specific tasks and deliverables are discussed in the next paragraphs). Therefore, teams carry out design, implementation, and testing in a much smaller scale than requirements analysis, as design, implementation, and testing are the focus of other courses.

The team project deliverables are shown in Table 2. Student teams receive fewer points for completing early deliverables such as finding a client and submitting their proposal (3%), and more points for advance deliverables such as SRS and Test Plan (9%). A large part of the project grade (60%) comes from the final deliverables and presentation at the end of the semester.

Table 2. Project Deliverables and timeline

Deliverable	**Timeline**
Project Proposal (group)	Week 2
Meeting #1 summary (group), Signed client form (individual)	Week 4
Meeting #2 summary (group), Signed client form, Peer review form (individual)	Week 6
Meeting #3 summary (group), Signed client form, Peer review form (individual)	Week 8
Software Requirements Specification (SRS), Test Plan (group), Peer review form #3 (individual)	Week 11
Presentations (group)	Weeks 10 and 16
Source Code, Test Results, User Manual (group), Peer review form #4 (individual)	Week 16-17

As seen in Table 2, the course requires three meetings with the client: the first meeting is focused on the team understanding the client needs, while the second meeting's goal is to conduct a detailed interview. Students are required to prepare diagrams and/or mockups prior to the meeting. The third meeting is for the team to present their prototype to their client and obtain their detailed feedback. Each client has to sign an individual form for each student at each meeting where the student details the student's preparation before the meeting and notes taken by the student during the meeting. This is besides answering student questions and providing feedback on mockups and prototypes.

After this, the first presentation is for the teams to demonstrate their prototype and present their client feedback, and show their plan for the rest of the semester. The final presentation is to demonstrate their working product and summarize their results. Students also complete four peer review forms in order to formally review their teammates. The final peer review has an overall review and rating for each student, which results in influencing the individual student project grade.

Clients: Two potential clients (other faculty) and projects were proposed by the instructor. In the end, student teams picked their own clients and not the suggested ones by the instructor. Clients ranged from Faculty at the University, to University organizations, to local business owners or employees. Even though not all clients were "real clients" from industry, most of the clients had a personal and/or professional interest in the application developed by the students. For example, one team developed a scoreboard system for a relay race and other track events with the Intramural Sports & Special Events Coordinator as client.

Hybrid Course Delivery: The course was developed to include an online (off-classroom) component that replaced some of the in-class lecture-based meetings (about a third of the lectures were off-classroom). On one hand, the motivation behind this was to add time and flexibility for project-related activities including client meetings. This avoids scheduling problems often observed in project-based courses[4]. According to the literature[11, 12,] as well as discussions with other faculty, a blended or hybrid course would take advantage of technology to remove the space and time class-meeting constraint and thus offer more flexibility: in such a course, "students can get on with their everyday life, without having to adapt systematically to a specific space and time, as they are obliged to do in face-to-face […] All this motivates the students' interest in the subject, which encourages learning and leads to better outcomes" [12]. In the case of our course for example, the project teams could collaborate via online meeting mechanisms and work on shared online documents or code.

At the same time, the course employed an inverted classroom model: in lieu of traditional lecture-based class meetings, students had tasks to complete before class meetings, such as readings and exercise activities; face-to-face class meetings incorporated discussion and hands-on application of the material studied off-class, in order to promote student engagement and active learning, as well as project-related activities. In summary, a partially hybrid course was selected for CEN 3073 as it still includes face-to-face time for lectures on complex concepts, hands-on activities, and guided project time, while it allows students the flexibility for organizing their project meetings and individual studying.

Student Assessment and Example Activities: In addition to the team project described earlier, the course included quizzes, activities, and one in-class comprehensive exam. The role of the quizzes was to assess student completing the assigned reading and exercises before face-to-face meetings in class. The in-class component was enhanced and assessed with in-class activities.

In-class activities were designed in multiple ways to introduce variety and further engage students. One type of activity was Think-Pair-Share: for example, a module on Requirements Elicitation posted before class included a few questions comparing different elicitation techniques. During class, each student was paired with another student to discuss the questions; some of the times the students were asked to submit their answers in writing. After the allotted time had passed, selected students shared their answer with the class allowing other groups to comment.

For the module related to SE Ethics and Professionalism, students were given assigned reading (such as the ACM Code of Ethics[a], and the article "Professional and ethical dilemmas in software engineering"[13]) along with a quiz to assess the reading before class. During class, the students worked in groups to discuss selected case-studies from "An Introduction to Software Engineering Ethics" [b], a curriculum module available from the Markkula Center for Applied Ethics at Santa Clara University. The student teams were given various questions such as "Who were the stakeholders involved?" and "Let's say you were employed in this project. How would you have reacted/behaved?" and they were instructed to discuss and submit their results in writing.

Another type of activity was a lab-type activity. For example, students were given introductory material for UML and State Diagrams (or statecharts) before class (note that students were introduced to UML diagrams and concepts in the SE Fundamentals course in the previous semester). During class, students were asked to generate a statechart given the problem statement below:

"The hypothetical system is an online system for buying tickets for events such as concerts:

- The system has 2 types of users: Customer & Administrator.
- The customer first searches for an event based on event type, performer, date, and/or location. They are able to click on an event and see more information, as well as available ticket prices, then select ticket(s).
- After the customer makes their ticket selection, they are able to view assigned seats and the seating chart. They are now given the option to accept the assigned seats or start another search.
- If the customer accepts the seats, the system starts the checkout process; otherwise the ticket selection starts again.
- The administrator is able to create/cancel events, and to view customer orders. They can also add content to events such as pictures or links, as well as change event location, date, time, etc."

Students were instructed to use an open-source tool, StarUML[c], although they were free to choose another tool. During class, the instructor and undergraduate TAs provided assistance with

[a] http://www.acm.org/about/se-code
[b] http://www.scu.edu/ethics/practicing/focusareas/technology/software-engineering-ethics.html
[c] http://staruml.sourceforge.net/v1/about.php

the UML tool as well as guidance for the notation and concepts related to statecharts. Students had additional time after class to work and submit their diagram for grading by the instructor.

Assessment

The CEN 3073 class presented in this paper had an enrollment of 36 SE majors, forming nine teams with 3-4 students per team (one student withdrew from the class). The overall course average was 89.2%.

Software Engineering at FGCU has set achievement standards in junior level courses to target that 40% of students in a course are at 85% or above, 70% of students are at 70% or above, and 80% of students are at 65% or above. Table 3 displays the performance achieved by the students in the course for each target level for specific course objectives using different items e.g. assignments or exam questions. For example, assessing the performance of the students on the quiz which is related to the first course objective, it is shown in Table 3 (in first row, last column) that 50% of the students scored 85% or more on the quiz, 85% of the students scored 70% or more, and 91% of students scored 65% or more. As shown in Table 3, all of the items assessed for the different course objectives meet or surpass the target goals.

Table 3. Target Performance achieved by students on course objectives.

Course Objective	**Items**	**Target Performance Level**	**Results (n=35)**
Learn techniques, processes, and challenges involved in requirements elicitation, analysis, validation and management	Quiz; Exam Questions; Assignment	Goal: 40% of the students score 85% or above	50%; 65%; 74%
		70% score 70% or above	85%; 97%; 94%
		80% score 65% or above	91%; 97%; 94%
Work in a group to perform activities and provide documentation for all phases of software development, focusing on requirements analysis, while also carrying out design, implementation, and testing	Deliverable 1; Deliverable 2; Overall Project Grade	Goal: 40% of the students score 85% or above	88%; 89%; 88%
		70% score 70% or above	100%; 100%; 100%
		80% score 65% or above	100%; 100%; 100%
Construct UML diagrams (e.g. Sequence Diagram, State Diagram)	UML Lab Activity; Exam Question	Goal: 40% of the students score 85% or above	88%; 65%
		70% score 70% or above	100%; 100%
		80% score 65% or above	100%; 100%
Conduct an oral presentation and demonstration of a software project developed by a group	Final Presentation	Goal: 40% of the students score 85% or above	88%
		70% score 70% or above	100%
		80% score 65% or above	100%

Student Evaluations and Instructor Observations

As in most institutions, our students evaluate course instruction by completing a University survey at the end of semester. This survey includes both Likert scale questions as well as open-ended questions. Specific statements from the survey were selected to gauge the student response to the course. Table 4 shows the selected statements from the University survey, along with a high-end rating/response for each statement, and the percentage of students from each course that gave that rating for the statement. The scale on each of these statements ranged either from Strongly Disagree to Strongly Agree. For example, 96% of students who took the survey for the course in this paper agreed or strongly agreed that they learned a great deal about the subject.

Table 4. Student responses that "Agree or Strongly Agree" with selected statements on the University end-of-semester student survey.

Survey Statement	**Responses (n=22)**
"I was always prepared for class"	96%
"The assignments helped me understand the subject"	96%
"The instructor uses a variety of instructional materials/methods in the course"	84%
"I learned a great deal about the subject"	96%

Moreover, students were given an anonymous survey with additional course-related questions and the results are shown in Table 5. As seen from the Table, a large number of students regard the project as a very positive experience as well as the client interaction. Also, the vast majority of the class felt that the hybrid course delivery included enough instructor interaction and in-class hands-on exercises and/or explanations.

Table 5. Anonymous End-of-Semester Survey results

Survey Statement	**Responses (n=34)**
Developing software for a client increased my motivation significantly	71%
I feel I gained some to a lot of experience and knowledge from my project	75%
There was enough time in class/after to interact with instructor about course material/ project	97%
There were enough in-class hands-on exercises/explanations about concepts	97%

In the survey, students were also asked open-ended questions listed below along with comments that serve as a representative selection of all comments entered by the class:
Question: "Which part/component of this course do you like the best? Briefly explain."

- "I liked the project part of the course the most. With the addition of needing a client feel like I learned a lot more this semester."
- "It was nice having some meetings off-classroom. I think the in-classroom material was just enough for the in-class time we had."
- "The online component made it easier to focus."
- "I liked the in-class group work (daily activity to turn in) learned about the topics easier."
- "Early prototype requirement. Having this due early really helped out project get rolling early on."

Question: "Which part/component of this course do you think could be improved? How?"

- "More in-class group activities"
- "I think more time spent on ethics section would be nice. Some extra emphasis on writing and evaluating requirements would also be beneficial, especially outside of this course."
- "I feel that RE tool should be towards the start of the class, so students can learn to use them as they work on their projects."
- "Have some kind of channel to find clients or provide project ideas to use as a jumping off point."
- "I think the topics from the modules could be more integrated into the project's assignments. Realistic applications make it easier to learn rather than just quizzes."

Overall, based on the student feedback and the observations of the instructor, the students responded well to the project, especially having the client interaction as well as using a mid-semester prototype. Regarding the modules, students reported high interest in the topics related to RE tools, and Agile and RE. In addition, several of the students communicated to the instructor that they were very interested in the SE Ethics case studies and would have liked more time to "go deeper". Beyond these, the students stated that the course could expand on writing and evaluating requirements as well as UML.

It was also the instructor's observation that the in-class activities, as well as the group interaction with the client and the instructor resulted in high class participation; overall, the students seemed to be engaged and to have fun in the course. Based on student comments and instructor observations, future offerings of the class would expand on Agile and RE, as well as on formal methods, and include more real-world examples and RE in practice. Preparation of the course material for the first time and especially the off-classroom components required effort. In addition, monitoring the progress of the groups (reading meeting summaries, individual client forms, etc.) can quickly become a burden on the instructor. Perhaps introducing an RE-course alumni as student supervisor[14] or project manager[9] would alleviate this burden and allow more frequent feedback for the groups.

Conclusions

This paper presents the experiences of the author developing and teaching a Software Specifications course for the first time. The course was delivered in a hybrid (blended) way, where a fraction of in-class lecture meetings were replaced by assigned readings and exercises off-classroom. Some of the off-classroom time was intended for additional flexibility to aid with student project scheduling. Additionally, student teams of conducted at least three client meetings, and developed documentation, mockups, prototypes, and a final software product. Overall, students seemed motivated by the structure of the course and of the project, and reported that they learned a lot from their project. Improvements suggested by students and supported by instructor observations include expanding on specific topics such as writing requirements, and Agile and RE, as well as offering more suggested clients and project ideas.

Acknowledgements

Support has been provided by grant SBAHQ-10-I-0250 from the U.S. Small Business Administration (SBA). SBA's funding should not be construed as an endorsement of any products, opinions, or services.

References

1. Berenbach, B., "A Hole in the Curriculum", *International Workshop on Requirements Engineering Education and Training (REET)*, 2005.
2. Macaulay, L., and Mylopoulos, J., "Requirements Engineering: an educational dilemma", *Automated Software Engineering*, 2(4), 1995, pp. 343–351.
3. Memon, R. N., Ahmad, R., & Salim, S. S. "Problems in Requirements Engineering Education: A Survey", *Proceedings of the 8th International Conference on Frontiers of Information Technology*, 2010, pp. 5.
4. Suri, D., "Introducing Requirements Engineering in an Undergraduate Engineering Curriculum: Lessons Learnt," *ASEE Annual Conference and Exposition Proceedings*, CD-ROM, Montreal, Canada, 2002.
5. Lethbridge, T., Diaz-Herrera, J., Richard, J., and LeBlanc, J., "Improving software practice through education: Challenges and future trends", 2007.
6. Callele, D., and Makaroff, D., "Teaching requirements engineering to an unsuspecting audience", *ACM SIGCSE Bulletin*, 38(1), 2006, pp. 433-437.
7. Zowghi, D., and Paryani, S., "Teaching requirement engineering through role playing: Lessons learnt," *International Conference Requirements Engineering*, 2003, pp. 233–245.
8. Al-Ani, B., and Yusop, N., "Role-playing, group work and other ambitious teaching methods in a large requirement engineering course," *IEEE International Conference Workshop Eng. Computer-Based Systems*, 2004, pp. 299–306.
9. Mohan, S., and Chenoweth, S., "Teaching requirements engineering to undergraduate students", *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education SIGCSE*, 2011, pp. 141-146.
10. Kilcay-Ergin, N., and Laplante,P., "An Online Graduate requirements Engineering Course," *IEEE Transactions on Education*, 56(2), 2013, pp. 199-207.
11. Kaleta, R., Skibba, K., and Joosten, T. "Discovering, designing and delivering hybrid courses". In A. G. Picciano & C. D. Dziuban (Eds.), *Blended learning: Research perspectives, Needam, MA: The Sloan Consortium*, 2007, pp. 111-143.
12. Alonso, F., Manrique, D., Martinez, L., and Vines, J.M., "How Blended Learning Reduces Underachievement in Higher Education: An Experience in Teaching Computer Sciences," *IEEE Transactions on Education*, 54(3), 2011, pp.471-478.
13. Berenbach, B., and Manfred, B., "Professional and ethical dilemmas in software engineering," *Computer*, 2009, pp. 74-80.
14. Gabrysiak, G., Guentert, M., Hebig, R., and Giese, H., "Teaching Requirements Engineering with Authentic Stakeholders: Towards a Scalable Course Setting", *First International Workshop on Software Engineering Education Based on Real-Word Experiences*, 2012.