Introducing Requirements Engineering in an Undergraduate Engineering Curriculum: Lessons Learnt

Dr. Deepti Suri

Assistant Professor
Department of EECS
Milwaukee School of Engineering
Milwaukee, WI 53202
suri@msoe.edu

Abstract

Requirements Engineering (RE) is the process of determining, analyzing, documenting, validating and maintaining the services and constraints of the systems that needs to be designed. Research indicates that on a typical software project, the percentage of time spent on RE and Design, Implementation, and Testing are 40%, 20% and 40% respectively, whereas for successful projects these numbers are 60, 15 and 25%. Because of the high importance of RE in the design of software systems, the need to introduce RE as a required course in the undergraduate Software Engineering (SE) and Computer Science (CS) curricula is getting more attention. This paper summarizes the author's experiences in developing and teaching a RE course to juniors in the new Software Engineering degree program offered at Milwaukee School of Engineering (MSOE).

One of the major issues holding back broad acceptance of RE in the software industry today is the widespread belief that major effort in software development lies in programming and testing. Our students have similar perceptions and beliefs and like some seasoned practitioners, are more interested in "how" to solve problems instead of discovering "what" to solve, i.e. gathering the requirements. This paper details how working on "real" industrial projects with external clients for the first time in unfamiliar domains, spending an entire term writing documents (instead of programming), being cognizant of ethical issues, and having to deal with ambiguous and conflicting customer requirements made this course very different and challenging for students. The challenges faced by the instructor in developing and teaching this course are also summarized.

1. Introduction

The chances of a product being developed on time and within budget are dependant on thorough and precise analysis of the client's current situation and needs. Informally, the client's needs are also called "requirements." A "requirement" is a specification of what should be implemented by a product. The IEEE standard defines requirement as "A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or

other formally imposed document. The set of all requirements forms the basis for subsequent development of the system or system component" [8] [9].

Requirements are primarily of two types: functional and non-functional. Functional requirements relate to the actions that the product must carry out in order to satisfy the fundamental reasons of its existence. Non-functional requirements are the desirable properties/qualities that the product must have. These are the characteristics that make the product fast, usable, portable, reliable, attractive etc. [1] [2]. Requirements Engineering (RE) is the process of determining, analyzing, documenting, validating and maintaining the services and constraints (i.e. the functional and non-functional requirements) of the systems that need to be designed for the client. The use of the term "engineering" implies that systematic and repeatable techniques are used to ensure that the requirements are complete, consistent and relevant. One early intermediate product of RE is the Software Requirements Specification (SRS) document [3] that describes all the externally observable behaviors and characteristics expected of a software system. A quality SRS is one that contributes to successful and cost-effective creation of software that solves real-user needs and usually incorporates the viewpoints of all the stakeholders who have an interest in the product.

The cost of discovering a requirement during construction of the product, or worse when the client starts using the product is expensive and inefficient and yet a large number of companies typically spend only about 10% of the total time allocated for the project on requirements gathering, 10% on specifications, 15% on Design, 20% on Coding and 45% on Testing (and hence it is said that a typical project follows the 40-20-40 rule). By the time the project is typically done (usually over budget and late), many companies discover that 50-80% of their total budget is spent on rework. It has been discovered that successful projects follow the 60-15-20 rule where 60% of the total time is devoted RE & Design [4].

Because of the high importance of RE in the design of software systems, RE is a required junior level course in the undergraduate Software Engineering (SE) curriculum at the Milwaukee School of Engineering (MSOE). SE-382: Software Requirements & Specification was developed and taught for the first time in our SE curriculum during the Winter Quarter 2000-2001. This paper summarizes the author's experience in developing and teaching this course.

2. Course Format for the First Offering

The academic schedule at MSOE is based on a quarter system with three quarters in an academic year. Each quarter involves ten weeks of instruction with the eleventh week devoted to final exams. SE-382 met three times a week for fifty minutes each and did not have an associated lab period.

The course objectives (as required by ABET) for this course are that:

Upon successful completion of this course, students will

- Be able to understand the role of requirements engineering in a variety of software development models,

- Be able to elicit requirements from system stakeholders and to overcome common obstacles to the elicitation process,
- Be able to analyze and negotiate software requirements,
- Be able to specify software requirements with use cases, formal methods, and other documentation techniques,
- Be able to model software system requirements using UML and CASE tools,
- Be able to validate software requirements,
- Understand the importance and common methods of managing software requirements, and
- Be able to communicate software requirements in written documents and oral presentations.

During the first offering of the course, the following topics were covered over a 10-week period:

Week 1:	Introduction to RE. Terminology and Challenges in the current environment.
Week 2:	Process Assessment and Improvement (CMM Models and Software Lifecycle models), Requirements Elicitation techniques.
Week 3:	RE Analysis & Negotiation, Role of Use Cases and Con-Ops in RE.
Week 4:	Ambiguity in RE, Software Requirements Specification (SRS).
Week 5:	SRS (contd.), Review & Midterm.
Week 6:	Setting & Managing Expectations, System Modeling.
Week 7:	Student paper presentations. [A list of research papers were provided to the students. In groups of 3, they were required to select, read, analyze and present the paper in class.]
Week 8:	Requirements Reuse, Change Management.
Week 9:	Validation, Review.
Week 10:	Student presentations of their final projects.

The book by Sommerville and Sawyer [1] was used as a textbook for the course. Even though as an instructor I found the book to be a good and readable book with lots of practical advice and guidelines, it was not very popular with the students. The main shortcoming noted in course evaluations about the book was that it did not walk the student through all the stages of the RE process with a specific example.

Several external projects were solicited from the industry and other academic departments at MSOE. Six candidate projects were received, out of which, four were chosen by the students at the end of Week 1. A team of four students worked on a project during the quarter. A brief synopsis of two (out of four) projects is presented below. For the other two projects (Projects 3 and 4) details are not being presented herein because of confidentiality agreements entered into with the industrial clients.

Project1 – A Medical Image Conversion Program: Using computer tomography (CT) images as the input, physical models of patients' bones and tissues can be built and using rapid prototyping (RP), it is possible to fabricate 3-D models. The goal of this project was to specify the requirements for the "interpolation module" of the software being developed to convert the CT image data to a RP format.

*Project 2 – The Golf Scoring Syste*m: The goal of this project was to specify the requirements for a portable golf scoring system. The system should have the capability to record characteristics of

golf courses (including pars per hole and difficulty ratings), players and league scores, calculate player's handicap (rules obtained from USGA web site) and present the results in a graphical manner. [Note: There are quite a few such systems already available but they all seem to be Windows-based. Our client was primarily interested in a portable system, which could also be run on a Linux/BSD environment.]

The project sponsors (also the primary stakeholders) provided all the documentation they had with respect to the project. The students were expected to sort through the information and then schedule an elicitation meeting with the stakeholders and gather information about the project. Since I had some previous experience with the two industrial projects and did not want the clients to be bombarded by a set of questions that I may have already asked them, I asked the students working on Projects 3 and 4 to route all questions to me and if I could not answer them to forward them to the primary stakeholders. In my opinion, I was stepping in as a mediator between the students and the sponsors and saving both of them time; but it denied the students the valuable experience of dealing directly with the clients.

During the course of the quarter, the students worked on four assignments (detailed later in this section), which were sent to the project sponsors for review as soon as they were submitted. The project sponsors and I reviewed the assignments and gave our feedback to the students. The sponsors were requested to send their comments within a week and almost always sent them back within that timeframe. All the industrial sponsors were present for the final project presentations.

The four assignments that the students were assigned are:

Assignment 1:	List all use cases, document major Use Case, Prioritize Requirements, Risk Analysis (if any) [Assignment due in two weeks after it was assigned]
Assignment 2:	SRS for the project [Assignment due in two weeks after it was assigned]
Assignment 3:	Model one sub-system of the project [Assignment due in two weeks after it was assigned]
Assignment 4:	Complete and finalize the SRS. Final project presentations to the project sponsors. [Assignment due in two weeks after it was assigned]

It should also be noted that no special project time was scheduled for the students to do these assignments. They were treated as Homework assignments and the students were expected to spend 2-3 hr per week outside of the class working on them. This created many scheduling problems for the students – not only with each other but also with the stakeholders.

3. SE-382 vs. a Traditional Software Engineering Course

Based on my experience, the students found the course on RE to be very different from what they had seen so far in their SE curriculum for the following reasons:

(i) No coding: Even though SE-382 is a software engineering course, it is the first course for SE students in which they do not do any design or implementation. Instead, majority of their time is spent writing documents and specifying the project requirements completely and correctly. Our students, like many seasoned programmers, believe that the major effort in software development is programming and testing. Consequently, they found the task of documenting the functionality

of the product boring, un-challenging and un- inviting. Some of the comments that I received on the evaluation in this regard were:

- "The course is not particularly interesting and all the emphasis has been on making the requirements." *(on "What would you like to be done differently in this course?" from preliminary assessment survey, SE-382, Winter 2000-2001)*
- No. It seems that our opinions of what the project is and how it is to be done differs from that of the professor. *(on "Has the course met your expectations so far? Why or why not? Please elaborate" from preliminary assessment survey, SE-382, Winter 2000-2001)*

(ii) Unfamiliar domains: As mentioned earlier, the students worked as teams on the same project through out the quarter. The projects that the students were working on were derived from domains that were unfamiliar to them, which created some anxiety because students were not able to converse very fluently with the clients (atleast initially) on the terminology being used. Since all four projects were new and very diverse, the students could not rely on the help that they had typically received from their peers in previous courses. Some of the comments that I received from the students that highlight this aspect are mentioned below.

- I think that the projects in this course are too diverse for learning requirements. Each group is in entirely different situations, which makes understanding very difficult. *(on "What would you like done differently in this course" from preliminary assessment survey, SE-382, Winter 2000-2001)*
- Familiarity with projects before they are assigned. *(on "Things which could be improved", from Final Evaluation, Winter 2000-2001)*
- More even load across projects. *(on "Things which could be improved", from Final Evaluation, Winter 2000-2001)*
- Too much expression of opinion, not enough hard fact. *(on "Things which could be improved", from Final Evaluation, Winter 2000-2001)*
- Homework assignments on a known or simple application domain. Medical systems too complicated. *(on "Things which could be improved", from Final Evaluation, Winter 2000-2001)*

(iii) External clients: Apart from going through the daily rigors of the course; the students were also dealing with external clients, who played an important role in determining their final grade. This contributed to students' general anxiety about the course. Further, due to the intrinsically different nature of this course, some SE students who had not done well in traditional programming courses did better than students who were considered strong programmers. This paradigm shift was not taken kindly by some students who considered themselves to be strong programmers.

4. Lessons Learnt From the First Offering

Diverse Examples: In the first offering of this course, my idea was to expose the students to a lot of different examples and show them how the same techniques could be used to solve a diverse set of problems. This did not work every well for them. They wanted to see one example (constantly being referred to in each class) and addressed in as much depth (if not more) as they were expected to go in for their projects.

Heeding to their criticisms and various discussions on learning styles [10] [11] which demonstrate that retention and learning is better when students work on the problems themselves – in the second offering of the course (Winter 2001-2002) I have decided to take one case study and tackle it in depth. I let the students work on a particular aspect of the process (generating contextual diagram, brainstorming on the list of stakeholders, business events, use cases etc, writing a use case, enumerating functional and non- functional requirements) and then discuss it in class. Typically, they put their solutions on the board and then we discuss each solution. Obviously, the time spent on each aspect has increased tremendously (which means certain topics have been eliminated) but the student retention and class participation has increased and the students seem to be having more fun!

Introduction to ethics in Software Engineering: I believe that our students, who will work as software professionals tomorrow, should be provided with opportunities so that they can practice their ethical reasoning skills on hypothetical yet realistic problems and they should get feedback on their responses. I believe that such experiences prepare them to face similar dilemmas later, when the stakes are much higher. It may be noted that ABET also requires the SE programs must also expose their graduates to ethics in engineering.

The course objectives during the first offering did not clearly mention ethics as an objective of the course so when an ethical dilemma was posed to them as part of the "take-home" portion of the midterm exam, some students perceived it to be unfair. In the second offering of the course, we have had some discussions on how failure to capture some important requirements (specially for critical systems) can lead to severe ethical dilemmas later but I have refrained from incorporating such questions on the exams.

Familiarity with the new tool: As part of the course, the students were introduced to use a new tool, "Rational Requisite Pro". In the first offering, the students were asked to become familiar with the new tool by going through the tutorial and asking for clarifications if/when they had questions. They were provided the time to learn the tool, but there were no specific tasks that had to be submitted in this case. When the students were required to use the tool as part of their first assignment, their lack of familiarity with the tool led to frustrations because they did not know how to do things that were expected of them (e.g., create a project, create a use case document, create requirements etc.).

In the second offering of the course, one class period has been devoted to going over the tool with the students. As part of their first assignment, they were supposed to submit a Requisite Pro project with a sample use case and a sample software requirements specification document which forces everybody to use the tool.

Scheduling problems: As mentioned earlier, in the original offering of the course, the students had scheduling problems; not only with the stakeholders but also amongst themselves to work on the project. I now schedule one class period every two weeks for project discussions. This is a class period – where the students can ask me questions related to their projects, use the class period to have project discussions amongst themselves or invite the primary stakeholder for further discussions.

Access to the primary stakeholders: As mentioned earlier, my acting as a mediator denied some students the valuable experience of dealing directly with the stakeholders and learning from mistakes. In the second offering of the course, I have backed off and let the students deal directly with the sponsors. The sponsors have a certain level of commitment towards the students and hence they agree to spend their time and energy towards sponsoring the project. I know that the students find the interaction with their sponsors very rewarding. I hope that sponsors also gain valuable insight in their interactions with the students.

5. Current Format:

The course is currently being offered again during the Winter 2001-2002 quarter. Based on the feedback received during the first offering, the current course format is as follows:

Week 1:	Introduction to RE. Software Life Cycle Models and RE.
Week 2:	Introduction to the CASE tool, Project Blastoff.
Week 3:	Event driven Use Cases, Trawling for Requirements.
Week 4:	Functional & Non-Functional Requirements.
Week 5:	Review & Midterm.
Week 6:	Writing Specifications. Validation Criteria.
Week 7:	Student paper presentations. [A list of research papers were provided to the students. In groups of 3, they we re required to select, read, analyze and present the paper in class.]
Week 8:	Quality Gateway, Prototyping & Scenarios.
Week 9:	Reusing Requirements, Change Management.
Week 10:	Student presentations of their final projects.

Based on the student comments regarding the previous textbook, the text was changed to the book by Robertson [2]. Based on the preliminary feedback so far, the students seem to like this book.

Projects: The four projects that the students are working on are

Project 1: ABET assessment repository Every engineering program at MSOE has its goals and objectives and so does every course in the curriculum. For assessment purposes, ABET requires that various programs track that the course objectives align itself to the program objectives. Similarly, assessment data has to be collected by each faculty to make sure that students enrolled in respective courses are meeting all the course objectives. Faculty typically assess whether the students have achieved the course objectives via lab-assignments, quizzes, exams, homework's etc.

In the long run, it is envisioned that MSOE will have its own in house software that will make it easy to track and assess the student data and also acts as a central repository for all assessment information required for ABET. The goal of this project is to elicit and specify requirements for this project.

Project 2: Dynamic documentation for data-navigator E-funds Data-Navigator is an online tool that automates the still mostly manual functions of transaction research, exception management, custom reporting and continuous settlement in the EFT (Electronics Fund Transfer) domain. It is

a product with client-server architecture; with the server component composed of many modules and each module being responsible for a particular task. This project has grown tremendously over the last 5 years and new people are constantly being added to the project. For a new person joining the project, there is a very steep learning curve because of the very complex design of the product.

There is a need for a tool that will dynamically create/update a drill-down document that will show the various modules that are involved in Data-Navigator and how they relate to each other. The goal of this project is to elicit and specify requirements for this tool.

Project 3: Web-based tool for capturing and tracking requirements Currently, we are using Rational's Requisite-Pro as the CASE tool for Requirements tracking. This is a tool that is available only on the Window's platform currently and includes a Windows and a Web client. The power of the tool lies in the fact that it not only uses documents (Word documents to be precise) to capture requirements but also tracks requirements using a database. Storing requirements in a database helps track, prioritize, organize requirements and maintain traceability.

The goal of this project is to gather requirements for a system that is portable (primarily web based) and incorporates all the features of requirements tracking, traceability and change management.

Project 4: Engineering case study repository Faculty members from MSOE and UW-Milwaukee are collaboratively working on a proposal for NSF (National Science Foundation) to create a publicly accessible web-based repository of engineering case studies from various disciplines to emphasize the positive impact the "engineering" profession plays in the society. Each case study will contain the following information regarding a case study: Application Area, Background Information, Engineering Disciplines Involved, Positive Impact of the Study, Goals and Objectives of the Study, Various hands on experiments associated with the study along with the learning outcomes of each experiment and additional resources associated with each one of them, Reviews/Feedback received regarding the case study etc.

As the repository grows, it is envisioned that other authors (from around the world) will start making contributions to the repository to support its expansion and will also be able to evaluate its usefulness in diverse settings. The goal of this project is to elicit and specify requirements for this repository.

The five assignments that the students were assigned are:

Assignment 1:	Familiarity with the CASE tool. [Assignment due in a week after it was assigned]
Assignment 2:	Project Blastoff (i.e. enumerate goals, identify stakeholders, enumerate constraints & risks, contextual diagram [Assignment due in two weeks after it was assigned]
Assignment 3:	Specify the major use cases for the system. [Assignment due in two weeks after it was assigned]
Assignment 4:	SRS for the project. [Assignment due in two weeks after it was assigned]
Assignment 5:	Complete and finalize the SRS. Final project presentations to the project sponsors. [Assignment due in two weeks after it was assigned]

6. Conclusions

This paper summarizes the author's experiences in developing and teaching an undergraduate course in Software Requirements and Specifications. The major criticisms and feedback received from the first offering are presented along with specific strategies that were used to address these concerns during the second offering. In my experience in a course of this nature, special effort needs to be made early on to set the student expectations that this course does not involve any design/coding. It is also important to continue to emphasize that spending a significant amount of time in a software life cycle on RE results in a higher quality end product.

7. References:

[1] *Requirements Engineering: A Good Practice Guid*e, by Ian Sommerville and Pete Sawyer, John Wiley & Sons, 1997.

[2] *Mastering the Requirements proces*s, by Suzanne Robertson and James Robertson, Addison Wesley 1999.

[3] *"Identifying and Measuring Quality in a Software Requirements Specification",* Alan Davis, Scott Overmayer et. al., Software Requirements Engineering, Second Edition, IEEE Computer Society, pp 194 – 205.

[4] *Software Engineering: Principles and Practice,* Hans Van Vliet, Second Edition, John Wiley 2000.

[5] *Exploring Requirements: Quality Before Desig*n, Donald Gause and Gerald Weinberg, Dorset House Publishing 1989.

[6] *Software Requirements,* Karl E. Weigers, Microsoft Press 1999.

[7] *Software Engineering,* Ian Sommerville, Sixth Edition, Addison Wesley 1984.

[8] *Requirements Engineering: A Roadma*p, B. A. Nuseibeh and S. M. Easterbrook, In A. C. W. Finkelstein (ed) *"The Future of Software Engineerin*g". (Companion volume to the proceedings of the 22nd International Conference on Software Engineering ICSE'00), IEEE Computer Society Press.

[9] *The Meaning of Requirements,* M. Jackson, Annals of Software Engineering, Baltzer Science Publishers, Vol. 3, pp. 5 – 21, 1997.

[10] *A First Step Toward Improved Teaching,* James E. Stice, Engineering Education, Vol. 66, Number 5, pp. 394 – 398, 1976.

[11] *Using Kolb's Learning Cycle to Improve Student Learnin*g, James E. Stice, Engineering Education, Vol. 77, Number 5, pp. 291 – 296, 1987.

DEEPTI SURI

Deepti Suri is Assistant Professor of Computer Science in the Department of Electrical Engineering and Computer Science at the Milwaukee School of Engineering (MSOE). She primarily teaches Software Engineering courses in the areas of Software Requirements & Specifications, Object Oriented Design, Design Patterns, Verification & Validation.

Prior to joining MSOE, Ms. Suri worked in the industry for seven years. She has provided systems solutions for the Electronic Design Automation (EDA), financial, and health-care industries. Her experience includes working in all aspects of the project's life cycle.

Ms. Suri holds three degrees B.S. (1989), M.S. (1991), and Ph.D. (1999) in Computer Science. She has written several articles in the areas of Robotics, GUI design and parallel computing and presented her work at national as well as international conferences.