

The Effectiveness of Teaching System Requirements Analysis

Regina M. Gonzales and Sara White
Klipsh School of Electrical and Computer Engineering
{regonzal, sarwhite}@nmsu.edu

As engineers, developing products that are a good match to user's needs is our livelihood. We do not want to produce products that are either rejected outright or expensively redeveloped. Nor do we want products that are expensive to fix or enhance because of poor preparation for testing and modification. These problems highlight the need for quality requirements engineering.

Not only should the requirements specify the project objectives and required product behavior, but the requirement specifications themselves need to be true, complete, feasible, verifiable (testable) and maintainable. Eliciting requirements may be difficult since requirements often are revealed in a staggered fashion and different stakeholders have different "win conditions." There may be conflicting requirements or nebulous, unverifiable requirements. Users may not be able to articulate their needs, as in "I don't know what I want, but I'll know it when I see it!" Some requirements may not be revealed until the user tries out a prototype, alpha model or the final product.

Requirements engineering methods have been developed to assist in specifying requirements. Teaching these methods to college engineering students should lead these students to develop more complete, consistent, and testable requirements. Teaching system requirements analysis in a university setting is difficult because of the breadth of knowledge and experience that is required. Understanding the big picture that the technical nuts and bolts fit into is difficult for students, but this knowledge is required before a beginning engineer's knowledge can become operational.

The objective of the course titled "Software Systems Engineering" is to walk senior and

graduate level students through a realistic project that is in an early conceptual phase. The project is of medium to large size, one that can not be brought to completion (code and hardware) in one or two semesters worth of work. The size is crucial since the goal is to give the students the opportunity to perform system requirements analysis for a non-trivial system.

The rest of this paper discusses the course design and the results of teaching requirements methods in the context of a project that was brought to some level of completion for the Boeing Corporation. The project is a Software Metrics Toolset (SMT), a tool that Boeing intends to use internally. The effectiveness of teaching requirements analysis is based on comparing the final set of requirements from the SMT Software Requirements Specification to the developed SMT

2. COURSE DISCUSSION

The course is not approached as another project oriented university course, but as an actual project would be approached in an industry setting. The instructor, who has over ten years of industry experience developing computer-based systems, acted as a facilitator (Gonzales/Wolf, 1996). She taught the necessary methods for each phase of development, as those methods were required.

2.1 Course Particulars

The Spring 1997 course started with funding from Boeing and a vague statement of what was needed for the Software Metrics Toolset. The first task was to develop a project management plan. This gave the students practice in requirements elicitation and project management.

The students developed a conceptual model for the system based on customer input. They then identified all the stakeholders for the project. Using the conceptual model and stakeholder profiles they identified the mission, objectives, assumptions and constraints for the project. From there they went on to develop an organizational chart, life-cycle model, and task breakdown for the project.

This was in essence the project management plan and it was difficult to set the stage for the project without it. That is why the instructor decided that it would be a component of the course even though there is no requirements analysis text that covers the development of a project management plan. The development of a project management plan is found in Systems Engineering and Software Engineering in separate texts as in (Blanchard, 1991) and (Thayer, 1987). The efforts spent on the project management plan made the remaining concentration on requirements analysis more fruitful as the plan provided necessary context for the requirements.

In order to arrive at a System Requirements Specification the students first used Structured Analysis Real-Time. During this time Operational Scenarios for the system were also created. Several levels of data flow diagrams were created until process specifications could be written. After about six classes of doing Structured Analysis, we went on to Object Oriented Analysis. We spent less time on Object Oriented Analysis and it took the students a little bit of time to adjust their thinking after doing Structured Analysis. A complete Object Model was developed including identification and specification of all of the system objects, classification and assembly structures, instance connections and transaction paths, and a table of all transactions at the system level. At this point enough was known about the system to write the System Requirements Specification. The Structured Diagrams and the Object Diagrams were included as appendixes in the specification.

The decision was made by the instructor to cover two analysis methods in greater detail rather than

cover more methods so that students fully understood the process of developing a complete system model. More than one model is desirable so that the students learn to look at the problem in different ways. For example the students discovered that Structured Analysis forces the analyst to more clearly define the system boundaries at the outset. Object Oriented Analysis is less clear on the process for defining system boundaries.

The course was offered for three credit hours, but the students agreed to meet four hours a week for two hours at a time. This allowed the flexibility of giving an hour lecture and an hour practice within one day. There was extensive team interaction during class and when time came for reviewing or inspecting deliverables, entire class periods were dedicated to that effort.

Homework was given as action-items. Each time the class met progress was made toward analyzing the system. This was done in an interactive manner. When the end of the class period came, goals for making further progress were made and action items were assigned to each team member. This is a common model used in industry. When the class met, each student presented the results of his or her action item and further analysis took place.

Part of their grade was based on weekly quizzes given to encourage reading material given and study of concepts presented. A percentage of the grade was based on the students' participation in class and the effort made on the action items. Since there were no wrong or right answers to most of the work, the grading was based on perceived effort and was a cumulative result of that effort. This evaluation method is also widespread in industry.

The remainder and larger percentage of the grade was a team grade based on the deliverables. The deliverables for first semester class were a System Project Management Plan, a System Requirements Specification, and a Users Manual. The first two deliverables required a first draft and a final to be turned in. The Users Manual

was considered a Final Examination for the course. The Users Manual was an exercise in taking material given throughout the semester and material embodied in the other deliverables and presenting it from the users perspective of a completed system. The outlines for the project management plan (Thayer, 1987) and the systems requirements specification (IEEE Std 830-1993) were given to the student, but they had to develop the users manual outline.

The customers for the project included initially a Senior Software Process Engineer at Boeing, George Yamamura. Subsequently, Mike Yanega, a Senior Software Engineer at Boeing, was assigned to follow through on the project. He was involved in the requirements process and reviewed all documents created. The students would e-mail questions to him and he would answer or give direction to the students. The customers assigned a grade to the final deliverables. The grade given by the customer was 50% of the grade assigned for that deliverable.

There exists no text to our knowledge that covers the material presented in this class from a systems perspective. Most notably material on project management and the material on developing a users manual are traditionally dealt with separately. The material presented in this class is a chronological progression of knowledge needed to develop a project, yet the treatment of this material by textbooks is very compartmentalized. The text by R. S. Pressman (Pressman, 1992) comes the closest but it is presented purely from a software perspective. A text that covered the bulk of the material from a systems perspective was selected (Wieringa, 1996). This text was used for the course and supplemented with material from other sources.

2.2 Teams

A large component of the course content was teaching the students to perform as a team. The intent was to have a project of significant size and complexity so that a team of from 3 to 5 people was required to accomplish the objectives.

Performing as a team did not come naturally to the students and team behaviors had to be modeled in order for the students to learn how to function as a team. More than half of the class time was spent working as a team would in industry. The students met with frustration at times in trying to develop deliverables, but the instructor did not allow the group to function disjointly. By the end of the course the students could carry on as a team during class time, and other meeting times, with or without the instructors input.

The initial instruction in the class was done to develop and emphasize the need for team participation and team coordination. An exercise in negotiation from the Harvard Law School (Harvard Negotiation Clearinghouse, 1993) was given as a lab exercise to emphasize the need for effective communication. The diversity that existed within the team was similar to industry teams (Lumsdaine/Lumsdaine, 1994). There were Computer Science majors and Electrical and Computer Engineering majors. There were graduate and undergraduate students. There were male and female students, and there were people from other countries for whom English was a second language. This offered diversity in the students' viewpoints and experience.

Each team member was made the leader for a particular deliverable or a component of a deliverable. Each leader still required the input and effort of the other team members to accomplish the deliverable that was assigned to them, but they had to lead the coordination effort of the team for that deliverable.

2.3 Effectiveness of Requirements Effort

The various requirements elicitation methods where used to acquire the complete range of system requirements. These requirements were used to develop a conceptual model. That conceptual model is documented in a Software Requirements Specification (SRS) which is reviewed by the customer. Working under the principle "its hard to edit a blank page", the overall goal of the requirements elicitation

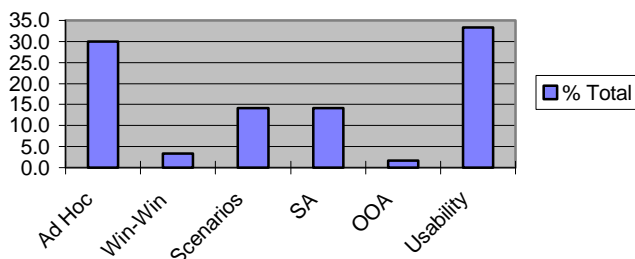
methods used to create the conceptual model is to provide as complete requirements coverage as possible. The user can correct wrong requirements, and conflicting requirements can be resolved, but completely missed requirements provide little the opportunity for customer feedback.

The requirements engineering (elicitation) methods taught and used in the classes were:

1. Brainstorming (Ad Hoc) from User Supplied Documentation
2. Boehm's Win-Win Conditions
3. Operational Scenarios
4. Structured Analysis (SA)
5. Object Oriented Analysis (OOA)
6. Usability Analysis

These methods elicit requirements by using personal past experience, industry standard "best practices", product lifecycle, stakeholders goals, system modeling, and events flow modeling to discover and solidify requirements. These methods are used to clarify vague or unverifiable requirements and turn them into specific, testable requirements.

Figure 1. Percent of Total Requirements By Method Type



Getting requirements ad hoc from user supplied documentation is often the only requirements elicitation method used to determine the system requirements. The practitioners of this ad-hoc method list the provided verifiable requirements and use past experience as a guide to help solidify vague or unverifiable requirements into specific, testable requirements. This can lead to incomplete requirements, especially with inexperienced practitioners and/or poor

documentation. The SMT supplied requirement documentation contained several non-verifiable goals, such as "the system should be easy to learn". In the SMT case, less than one third (36/120) of the verifiable requirements elicited were found after the completion of the Ad Hoc method! Figure 1 shows the percentage of total SRS requirements that were clarified enough as a result of the methods to be written as the verifiable requirements

It is clear from the graph that when it comes to requirements elicitation methods, "one size does not fit all". No one requirements elicitation method contributed more than one third of the final requirements. Yet all together, they presented enough clarity to create a conceptual model that was effective in bringing the developers and customers to agreement on the requirements of the SMT with very few, minor changes occurring during the actual SMT development.

The different methodologies looked at the system from different angles and worldviews, and as a result they clarified vague requirements or uncovered new requirements that were missed by other methods. The Structured Analysis and Object Orientated Analysis had the most overlap system clarification. This is not surprising since both modularize the system and define modular action and interaction, even though the two methods represent two different computing paradigms.

The use of multiple analysis methods with similar goals was not totally redundant. Since analysis methods usually form the basis of design approaches, important insight into the system structure (control driven or data driven) was gained from the multiple approaches. In addition to the verifiable requirements, the Win-Win Condition analysis introduced several important non-verifiable installation and maintenance requirements that were clarified and solidified by later methods. The Object Orientated Analysis provided the foundation for the database design, even though few new

requirements not already detailed from the Structured Analysis were found.

After the conceptual model and SRS were revised using the customer SRS review feedback, a limited prototype of the conceptual model (called an Alpha Model) was developed and provided to the customer for evaluation. (An alpha model has very limited functionality, but fairly detailed user interface. It models the way the user will interact with the product. Alpha models can even be paper simulations of the user interface flow.) The requirements changes that result from the alpha model evaluations were incorporated into another SRS revision. The final goal of these activities was to provide complete, consistent, testable requirements specifications with fine enough granularity to identify risks and aid in risk abatement efforts.

After reviewing the Alpha Model, the existing requirements were re-prioritized by the customer. The customer added additional requirements and several existing requirements were modified or deleted. Negotiations covering which requirements were to be included in the first version of the SMT were made. Using the Alpha model to elicit any final requirement changes before SMT development shaved the SMT development time. One new requirement added by the customer after evaluating the SMT Alpha Model would have meant total data entry screen re-development if this requirement change had come after development. And the data entry screens that would have been scrapped would have taken two and a half as much time to develop in the first place than those finally required!

4. CONCLUSIONS

The goal of this System Requirements Analysis class and ongoing curriculum development in this area is to create synergy between industry and New Mexico State University. The area of computer-based systems analysis is an area where having realistic projects on which the students can work is crucial to giving students the necessary systems analysis and design skills.

It also gives industry an opportunity to influence the finishing touches that a student receives in order to make the students more productive in an industry setting.

5. REFERENCES

- B. S. Blanchard, *System Engineering Management*. John Wiley and Sons, New York, 1991.
- R. M. Gonzales and A. L. Wolf, "A Facilitator Method for Upstream Design Activities with Diverse Stakeholders." *Proceedings of the Second International Conference on Requirements Engineering* (Colorado Springs, CO, April 15-18, 1996), pp. 190-197.
- Harvard Negotiation Clearinghouse, *Clearinghouse Catalog 1993*. Program on Negotiation, Harvard Law School, Cambridge, MA, 1993.
- D. J. Hatley and I. A. Pirbhai, *Strategies for Real-Time System Specification*. Dorset House, New York, 1987.
- IEEE Standards Board, "IEEE Recommended Practice for Requirements Specifications Software, IEEE Std 830-1993." IEEE Press, 1994
- E. Lumsdaine and M. Lumsdaine, "Team Thinking That Measures Up to the Task at Hand." *IEEE Potentials*, pp. 4-9, December 1994.
- R. S. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, 1992.
- R. H. Thayer, *Tutorial: Software Engineering Project Management*. IEEE Computer Society Press, 1987.
- R. J. Wieringa, *Requirements Engineering: Frameworks for Understanding*. John Wiley and Sons, New York, 1996.