Software Engineering Disciplinary Commons Portfolio

Sriram Mohan

Dept. of Computer Science and Software Engineering
Rose-Hulman Institute of Technology

November 6, 2010

Contents

Introduction

This portfolio documents my work as a part of the Software Engineering Disciplinary Commons. A Disciplinary Commons involves a group of educators from diverse institutions who teach within the same discipline meeting monthly during an academic year to share, reflect on and document their teaching. Participating in the commons has been an invaluable experience and provided an opportunity to reflect on my teaching, course syllabus and identify strengths and weaknesses and compare our practices with other participants.

The goal of this portfolio is to document my teaching philosophy and provide a platform to understand the rationale behind a course on Software Requirements and Specification. I have also included several appendices that provide supporting information including my current syllabus, assignments, project milestones and rubrics for evaluating said milestones.

Chapter 1

Teaching

Growing up, my childhood was filled with stories of my father and my aunt tutoring their friends in high school and university. It has been constantly impressed upon me, that the best way to learn something is to be able to teach it to others. I completed my undergraduate studies in Computer Science and Engineering in India. During this time, my opportunities to teach on a formal basis were extremely limited. However, my residence used to be the hub for students in my graduating class. I had a wonderful time exploring new concepts in computer science & engineering and explaining them to my friends. My aunt, impressed upon me the need to continue my education if I wanted to pursue my passion for teaching.

I continued my doctoral education in Computer Science at the Bloomington Campus of Indiana University (IU). My graduate education provided me with several opportunities to observe and learn from exemplary teachers. As a student, I took a class on "Teaching Computer Science". The course helped me develop an idea for the general principles of teaching and provided some practical experience in teaching computer science. An important aspect of the class was microteaching, in which we were required to prepare and lead classroom sessions on various topics. Each session was followed by a critical analysis and discussion led by faculty and staff from the IU School of Education and the Office for Campus Instructional Consulting. This experience was invaluable in my growth as a teacher.

I turned down the opportunity to receive funding via research assistantships to focus on developing as a teacher through teaching related assistantships. I served as the lead teaching assistant for our introductory course on *Networking Technologies*. I developed and led various labs that introduced the students to basics of computer networking. My zeal and passion for teaching and stu-

dents was recognized by the lead instructor who recommended me to serve as the teaching assistant for the undergraduate as well as the graduate version of *Software Engineering for Information Systems*. I served as the teaching assistant for this course for 5 years, delivering many lectures, leading class discussions, mentoring and providing technical assistance to student teams and student project managers. I was intimately involved in the conception, design and development of a unique class that simulated software development in the industry and exposed student to prevailing practices in the real world. During the summer semesters, the Computer Science department at Indiana University invited me to be a full time instructor. I served in this position by teaching *C212 - Introduction to Software Development* for four years. I helped revamp the curriculum, by designing a) new labs to help students understand the concepts discussed in class, b) lecture sessions that focused more on interactive problem solving and c) a final project that tested their ability to develop software. Students enjoyed my classes, resulting in positive student evaluations and increased class enrollments. These experiences gave me a chance to experiment with different teaching methodologies and helped hone my teaching skills and confirm my passion and desire for teaching.

This led me to apply for positions at institutions that placed the highest value on education, classroom interaction and instruction. I was encouraged to apply for the position at Rose-Hulman by a Rose Alum who felt that I would be a great fit at his alma mater. I was thrilled, when Dr. Cary Laxer approached me to schedule a phone interview and my interview visit to Rose-Hulman. I had an incredible experience talking to the students, the faculty and the staff. I had received offers to teach at colleges of similar stature, but was convinced that Rose-Hulman was the place for me.

I still remember my early morning conversation with Cary, when he called to make me an offer – I jumped at the opportunity and gave him a verbal confirmation of my acceptance of the offer. The last two and half years have reinforced my initial impression of Rose-Hulman. It is a privilege and a pleasure to teach at Rose-Hulman and I look forward to serving here for many years.

1.1 Teaching Philosophy & Methodologies

"Develop a passion for learning. If you do, you'll never cease to grow" - Anthony J. D'Angelo

The ideas that I have implemented as a teacher had their initiation during my days as a student, observing teachers who have left a deep impression on me. These ideas continue to be refined with my own experience as a teacher and as a mentor to high school students. The most important observation I have had is that *good teachers do not concentrate on effective teaching but on effective learning*. This is something that I have taken to heart and apply to all facets of student learning.

Learning:

Be it the theoretical aspects or the more engineering/system oriented aspects of Computer Science, it is my belief that it is not enough for students to understand a particular concept, and be able to repeat it from rote memory. Instead, students must develop the ability to apply what they have learned. I believe that a student's comprehension of the concepts is greatly enhanced, if they are engaged with the material. As such, I utilize a number of active learning techniques in the classroom including:

1. *Case study discussions:* In my classes on software engineering, where context is key, I introduce case studies from industry/academia that detail software product success stories and failures. Students are expected to prepare a pre-discussion report that is due before the class session. During the class session, we analyze the case study from the perspective of the concepts learned in the classroom and we try to identify the a) rationale behind the failure, b) key reasons behind the success of a strategy and c) mechanism to improve the software process. These case studies provide the student with an opportunity to apply their knowledge by reflecting on a real world example.

2. *White board problem solving:* I tend to make use of end-of term group projects in my classes. To encourage group work and engage the students in the classroom, I routinely ask student groups to work on a problem, using the various whiteboards / flipcharts around the room. The students are required to solve the problems as a group in a specified period of time and then reflect on the solution to the classroom. This technique helps a) develop a group identity and b) develop a sense of cohesiveness while simultaneously improving their presentation skills.

3. *Mock Projects:* I have used mock projects in the classroom to provide context to our discussions. This technique is extremely useful in courses

on software engineering such as Software Requirements. Rather than just teaching students the art of gathering and specifying requirements, I have organized the discussion around project scenarios, where my teaching assistants and I play the role of client stakeholders and the student teams act as the software developers. The mock projects provide students the ability to apply the software management and development practices they learn in the classroom. It also helps them identify the techniques that are applicable to various project situations. Computer Science is a rapidly changing field and the project scenarios are selected with an eye towards exposing students to new practices and software technologies.

4. *Live programming sessions:* I believe that to learn the art of programming, the student must a) understand the syntax of the programming language and b) learn to solve a problem using said programming language. In my experience as a teacher, I have noticed that student typically find it difficult to learn the latter. To help students get over this hump in my class on database management, I organize Live programming sessions where we write database scripts together as a class with the evolving solution on a projector and the instructor (me) acting as the driver of the solution.

5. *In-class labs:* I organize the class in a studio format and intermingle the lecture with a lab component. The class is organized in ten-minute sessions with a lecture on a concept alternating with a lab where the students have to a) implement said concept as a part of a scaffold or b) apply the concept to solve a problem.

6. *Solve a Problem:* As a part of the class discussion, students either individually, or as a group solve a problem and thus learn how the concept is used and how to apply it. *The critical idea in this technique is that each individual/group contributes only one step in the solution. The next step in the solution comes from a different student/group*. This iterative process is repeated until the problem is solved. Initially, students were afraid of making mistakes and were reluctant to speak up. To get around this, I helped them understand that every person including the teacher makes mistakes and they were not going to be penalized; the more important thing was to learn from the experience. Students have indicated that they enjoy this segment of the class and that it has also helped shy students actively participate in class.

7. *Research Reflection:* This is similar to the notion of a case study discussion and is a technique that I have used in advanced electives. Students are required to read research papers from journals and peer-reviewed conferences. Students are expected to prepare a pre-discussion report that is due before the class session. During the class session, we analyze the paper, identify and discuss a) the key contributions b) advantages & disadvantages of the proposed techniques and c) future research directions. The papers provide the students with an opportunity to learn, reflect and stay current on research. I have supplemented this activity by requiring students to propose and implement extensions to these techniques as a part of their end-of term research projects.

8. *Written Reflection:* Reflective exercises where students, write a couple of paragraphs in their own words and then share them with the class. This provides some insight into their ability to comprehend the concepts covered in the classroom.

9. *Daily Quizzes:* This was an idea; I was introduced to by Dr. Curt Clifton. This model is based around the concept of "Test First Teaching" [1] introduced by Dr. Cheryl Dugas and Dr. Mark Ardis. I have expounded more on my take on "Test First Teaching" in the next page. I have led the successful introduction of Daily Quizzes in the following classes : a) Software Project Management, b) Software Requirements & Specification, c) Advanced Databases & d) Software Quality Assurance. I have a few objectives in mind, when I write my quizzes:

    - A mechanism for me to evaluate whether the students have understood the concepts that I covered in class. This provides me with immediate feedback at the end of the day. If a majority of students answer a particular question incorrectly, I re-teach the concept in the next class.
    - A mechanism for the students to take notes on the important concepts of the class.
    - A method for me to get to know the students better. I usually put in an optional personal question such as "*Name your favorite place to visit*".

[1] Mark A. Ardis and Cheryl A. Dugas. Test-first teaching: Extreme programming meets instructional design in software engineering courses. In *34th ASEE/IEEE Frontiers in Education Conference*. IEEE, 2004.

- I count it towards participation points. It gives the shy students an opportunity to have their thoughts heard.

To further encourage students to apply concepts confidently and in innovative ways, I have used incentives such as *extra time* in assignments and *extra credit*. The emphasis on active learning requires each class session to be planned in a careful manner. As a computer scientist, I like algorithms and I follow an algorithmic procedure while creating a class session. The procedure detailed below forms the basis for my take on " Test First Teaching":

1. Identify the learning outcomes for the class session.
2. Identify the concepts to be conveyed to help the students understand the outcomes.
3. Identify the teaching techniques that work for each concept.
4. Identify the in-class activities to best convey the concept.
5. Identify a mechanism to assess the student's comprehension of the concepts.
6. Connect the activities into a classroom session.

Reinforced Learning: I am an avid proponent of a three-tier model of reinforced learning. Each technique covered in class using active learning exercises is augmented with a homework assignment practiced using project based scenarios and personas. The students are then required to work on a end of the term project to provide them with an opportunity to directly apply the concepts they have learned.

Keeping with my core principle of effective learning, my projects and exams, focus on *critical thinking* and verify whether a student has *understood the material*. It is important for students to develop the ability to reason and think about solutions. To develop this aptitude, projects are initially seeded with suggestions to get the students thinking about possible solutions. I help student groups divide the project into significant milestones; to enable students to develop problem-solving skills utilized in complex projects. During the later stages of the term, this initial harness is slowly removed as students get more comfortable dealing with sizable projects. During weekly milestone meetings, I encourage students to reason about the chosen strategy, discuss

its merits and demerits and create a plan of action for the subsequent week.

I believe that exams, if used carefully, are a good way to gauge student progress. My exams focus on a student's ability to apply a concept and do not force the student to recite from memory. For instance, in my "Introduction to Database Systems" course, rather than asking students to state the definition of Entity Relationship Diagrams - a key component of database design; the exam problem, provides the student with a real life scenario, that students have to think through and solve - the entities they have to create, the relationships that need to be identified and the constraints that need to be imposed on the data. In essence, I have strived to model my exams to create a realistic application scenario for the knowledge they have gained in the classroom.

1.2 Environment

I am fascinated by Computer Science and it is important for me to *convey and communicate this passion* for the field to my students. One of the goals I have for my classes is to create an environment, which challenges and stimulates students. I want my students to experience an *engaging, interactive, energetic, dynamic and comfortable learning environment*. I am a huge proponent of active learning as laid out in the previous section. The use of various active learning exercises helps create a very engaging environment. I rarely "lecture" in my classes. I firmly believe in involving the students in the classroom – be it via a discussion or via problem solving.

I have frequent seminars in my classes wherein we talk about the rapid strides that are being made, the research that is being carried out, the exciting opportunities that are available, and how students can make a difference in the field. I have used these seminars to talk to students about my own research in information security and privacy, the challenges involved, its importance and how they can get started with research. I have encouraged and helped students apply for "Research Experience for Undergraduates" programs at several research oriented schools. I have a practice of inviting guest lectures from the industry and academia to my classroom. I have used industry guest lectures to help supplement the software engineering practices I convey in the classroom. Guest lectures from academia (primarily from Indiana University - my alma mater and Purdue University) have been very useful in exposing our students to the latest research trends in the areas of Databases, Software Engineering and Human Computer Interaction. Stu-

dents have welcomed my initiatives to provide them with this opportunity. For instance,

- "*Sriram was always enthusiastic about the subject and went to great lengths to make the course more informative. For instance, he gave us interesting real-world examples to read and had guest speakers come to talk to us about the subject*"

Each student is different and brings a unique perspective to a class. Understanding the student is very important to me as it helps me relate to them, gauge their knowledge and build on it in class. Learning is individual and is content and context specific. As a teacher, I am sensitive to these factors and *adapt my teaching methodologies* to suit the material being covered and the students in the class. Teaching has multi-faceted goals and is not just restricted to classroom activities such as explaining course material, enriching student knowledge, but is also about *molding an individual and maximizing potential*. The latter requires the teacher to *establish a good rapport with the students*. From my experience both as a student and as a teacher, I have realized the importance of the 'student-teacher' comfort factor in enabling students to approach the teacher with problems they might be facing. I have always welcomed frank and open discussions with my students and have used my experience to point them in the right direction. Students have appreciated my efforts to get to know them. For instance,

- *"He has always been available to help me when I have needed him, but has come to me whenever he sees me in the lab to check up on me and make sure all is well. He is an outstanding professor and an amazing person. I look forward to future classes with Sriram*"

- *"I really believe Sriram made this class exciting and relevant for me. His teaching methods were adaptive to students needs and he did an excellent job listening and adjusting to what was expected/needed. He exhibited an excellent knowledge of the material and was always available. This was an excellent course*"

- *"He taught as if I was the only one in the room... Whenever I missed class he was always willing to meet with me to make it up. He would walk through the daily quizzes with me and make sure I understood everything before moving on to the next subject*"

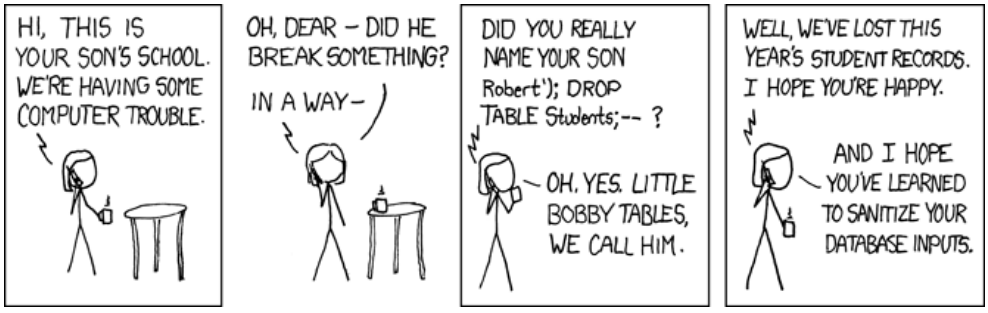

Figure 1.1: An Example of an XKCD Cartoon Strip used in the Class

I have used humor in the classroom and outside the classroom to help create a comfortable learning environment. I have picked up on the department tradition of "cartoon of the day". These cartoons are from a variety of sources, but as is common in the Computer Science arena, I tend to favor Dilbert and XKCD .[2] The comic strips are selected with a nod to the topic under discussion. For instance, while discussing SQL Injection attacks in my databases class, I have incorporated the XKCD Cartoon Strip shown in Figure 1.1.

I have always viewed teaching as a *collaborative learning process*. I let my students know that their *opinion and thoughts are valued and always welcome* in my classroom. To encourage students to reflect on the classroom practices, and to provide me with regular feedback, I encourage students to stop by my office with any feedback they may have. I have also setup an anonymous feedback form on Angel. This feedback form is configured with an agent, that sends an email to me when a student provides feedback on the class. I have a point of responding to students about their concerns in the next classroom session. I provide my response to the feedback and explain the rationale behind the change or the lack thereof in the class.

Dr. Kay Cee Dee introduced to me the practice of a plus-delta assessment. I perform *two plus-delta assessments* on the course and my instruction during a quarter. The first session is conducted at the end of Week 3 and the second at the end of Week 6. These provide the students with the opportunity to reflect about the class and provide me with additional feedback. These evaluations - both good and bad- have led me to introspect on my performance as a

[2]Dilbert is available from http://www.unitedmedia.com/comics/dilbert/. XKCD is available from http://xkcd.com/

teacher, enabling me to refine my teaching to better suit my students. Student have indicated their immense satisfaction with this process. For instance,

- *"Also great is how he listens to and looks for feedback related to the course. He gives 2-3 plus-delta surveys throughout the quarter that allow us to give feedback. He takes these seriously and actually makes changes to the course"*

I have regular evaluations(biweekly in my design courses) in my classes, during which I meet with students individually and provide them feedback on their performance. During the evaluations, I discuss concrete short-term and long-term goals for the student and inculcate a *sense of personal responsibility* towards achieving them. The personalized attention given to each student has given weaker students a sense of belonging and has enhanced their learning. For instance,

- *"Sriram has helped to hone my abilities as a software engineer through his helpful and thorough feedback, as well as through his dedication and willingness to spend time guiding me when I needed it. "*
- *"I have been able to approach Sriram with my own goals and he has given great advice."*

I believe I have been successful in creating a welcoming and engaging classroom environment, where students have a sense of ownership and responsibility towards their learning. My course evaluations reveal that students appreciate and welcome the rapport that I have developed with them. For instance,

- *"He is an asset to the department. I have never had a professor here that cared about the material, cared about the welfare of the students, knew about the material, and was able to communicate with the students to the level he could"*
- *"He was great. He was very interactive with the students. Each quiz had a personal question on it so he got to know us better. He was always available outside of class to help or chat with and was really interested in us as people. His teaching methods were good and he always had material ready that he wanted to cover. He moved through the content quickly in class without it appearing like he was in a rush - he still had time to stop and answer questions"*
- *" Sriram was a good teacher because he is passionate and makes you do work. Doing is what teaches me the most and his methods helped me out significantly"*

- *"Sriram is a great asset to the department. He is always available for help and connects well with students. I've learned a lot from him about requirements and database"*

This *comfortable, yet intellectually stimulating environment* has enhanced class participation and rekindles their curiosity about the subject and increases learning. To conclude, as a teacher I believe that

"The mind is not a vessel to be filled, but a fire to be kindled - Plutarch"

Chapter 2

Teaching

2.1 Introduction

The vital role played by software requirements in the eventual success or failure of a software product has been document by research through the years [6, 7, 14, 19]. A study by the Standish Group[1] paints a very gloomy picture of the state of the software industry. The report reveals that 31% of projects will be canceled before completion and that 52.7% of the projects will cost about 189% of their estimates. The report also identifies three common factors behind these sobering numbers:

- Lack of user input (13 % of the projects)
- Incomplete requirements and specifications (12 % of the projects)
- Changing requirements and specifications (12 % of the projects)

The report also duly identifies several factors that were common to successful projects. The three most important factors that were commonly cited include:

- User involvement (16% of the projects)
- Clear statement of requirements (14% of the projects)
- Support from upper management (12% of the projects)

A recent study[22] of software organizations in the United States and Australia reveal that requirements continue to be a problem for software development and one of the most common causes of runaway projects.

This problem is further compounded by the cost associated with fixing requirement errors. *"If a unit cost of one is assigned to the effort required to detect*

and repair an error during the coding stage, the cost to detect and repair an error during the requirements stage is between five to ten times less" [10].

Software requirements engineering is commonly taught under the aegis of software engineering. The emphasis placed on software engineering in undergraduate education is varied - ranging from optional, to one or two course sequences in software engineering, to coverage through a senior capstone project, to complete degree programs. An analysis of current undergraduate curriculum reveals the following themes:

1. Students typically do not encounter issues of large scale requirements, design, or other principles until their senior year [15].

2. A lack of emphasis on the importance of requirements in computer science education (Most programs tend to cover this area using a couple of class periods) [12].

3. Use of simulated project examples with faculty clients to convey the art/science of requirements in software engineering programs. The simulated experience, while beneficial does not provide the student with a realistic flavor of client-developer interaction and tends to mask some of the challenges in requirements elicitation [12].

As Walker et. al point out [23] "the process of software development is learnt by developing software under conditions similar those used in industry". The importance of using real clients is further confirmed by [2] and [8]. The rest of this paper describes our experiences with a new approach to teaching requirements engineering. This course utilizes a three tiered model of learning and emphasizes hands-on experience with various facets of requirements engineering through interaction with real clients with real needs.

2.2 Educational Context

It is important to understand the educational context within which this new course on software requirements has been implemented. We are a four year engineering college with an emphasis on undergraduate education. Currently our department offers undergraduate programs in computer science and software engineering. As a part of these programs, graduating software engineering students are required to complete a year long senior capstone experience. This experience requires students to work on teams of about four students on a project for a client to understand the need for, design, and develop a system that meets the identified needs, and to deliver and refine the system

in response to feedback about those needs. Past examples include companies like Microsoft, Omni Software, Turner Broadcasting, Mayo Clinic, Naval Surface Warfare Center, etc.

To help educate students on the skills necessary for a successful senior capstone experience and to provide them with a sound background in software engineering, we require our students to take a series of courses on the following subject areas:

1. Quality Assurance
2. Requirements and human computer interaction
3. Project Management
4. Architecture and Design
5. Construction, Maintenance and Evolution
6. Formal Methods

Over the past couple of years, in order to create an experience similar to those faced by software professionals and to provide our students with more opportunities to work with real clients, we reorganized some of the above courses into a Junior Design Sequence. The courses that comprise the junior design sequence share a common project (similar in idea and execution to that of the senior capstone experience). These courses[1] include the following:

1. Requirements and Human Computer Interaction
2. Project Management
3. Architecture and Design
4. Construction, Maintenance and Evolution

It must be noted that students in the computer science program have the option of pursuing either a senior capstone experience or a senior thesis to complete their graduation requirements. However, irrespective of this choice, the computer science students are required to participate in the junior design sequence through the following courses

[1]It must be noted, that software engineering students would have received a background in quality assurance fundamentals including test-driven development, metrics and quality assurance before they begin the junior sequence.

1. Requirements and human computer interaction
2. Architecture and Design

To align the Junior Projects with the classes the students are taking, we follow a hybrid process. This begins with classical requirements elicitation and documentation, where the students also learn to interact with project stakeholders and hone their verbal and written communication skills. The teams then move into a more iterative development process, regularly delivering successively more refined systems and documentation.

The junior projects tend to have a smaller scope than a traditional senior capstone experience. And because the requirements phase is more structured, Junior Projects can sometimes accommodate projects that might be less clearly defined at the outset. For Junior Projects we favor proposals that can be implemented using object-oriented languages and systems. This constraint supports the educational outcomes of our Software Architecture and Design course, which emphasizes object-oriented design patterns.

2.3 Our Course on Requirements

We established a curriculum based on our experiences in the industry and by surveying current requirements analysts and software engineers. Given the nature of courses that followed our course in the junior design sequence, we organized the student learning outcomes to target areas of software engineering that highlighted working with the end user and other stakeholders to elicit their needs, define the features and develop a usable interaction experience. Accordingly, the course had the following high level objectives.

Students who successfully complete the course should be able to:

1. Explain the role of requirements engineering and its process.
2. Formulate a problem statement using standard analysis techniques.
3. Determine stakeholder requirements using multiple standard techniques.
4. Produce a specification with functional and non-functional requirements based on the elicited requirements.
5. Decide scope and priorities by negotiating with the client and other stakeholders.
6. Manage requirements.

7. Apply standard quality assurance techniques to ensure that requirements are: verifiable, traceable, measurable, testable, accurate, unambiguous, consistent, and complete.

8. Produce test cases, plans, and procedures that can be used to verify that they have defined, designed and implemented a system that meets the needs of the intended users.

9. Design and prototype user interfaces to validate requirements.

10. Prepare and conduct usability tests to evaluate the usability, utility and efficiency of the developed user interface.

To meet the above learning outcomes, the course covered the following topics:

- Requirement Preliminaries
  - Why requirements?
  - Role of requirements in the software life cycle
  - Responsibilities of a software team
- Problem Analysis
  - Identifying need and defining a problem
  - Problem statements and formats
  - Root cause analysis
  - Domain modeling
  - Personas
- Understanding Needs
  - Problems in obtaining requirements
  - Needs versus features versus requirements
  - Interviewing
  - Brainstorming
  - Storyboarding
  - Data gathering and analysis

  - Requirement workshops

- Defining and Refining the Software System
  - Use cases
  - Vision document
  - Non-functional requirements
  - Ambiguous requirements
  - Assessing requirements quality
- Scope Management
  - Scope triangle
  - Managing clients
  - Traceability
  - Managing change
- Interaction Design and Prototyping
  - What is interaction design?
  - Cognition
  - Low fidelity prototyping
  - high fidelity prototyping
- Evaluation
  - Usability testing
  - Analytical evaluation
  - Evaluation frameworks
- The Transition
  - Requirements to design
  - Requirements to testing

We have refined the list of topics mentioned above over the last three years, based on feedback we have received from students, alumni and our industry board of advisors. The transition from requirements definition to system design can be a smooth process if handled in the right manner. We are currently considering the introduction of system sequence diagrams, and operation contracts to assist in the transition of user requirements to provide an ideal launch platform for our course on software architecture and design.

2.4 Textbooks

A review of books used in various schools indicates the following books are commonly used:

1. Software Engineering by Ian Sommerville (2010) [21]
2. Software Engineering - A Practitioner's Approach by Roger Pressman (2009) [18]

These two seminal books provide an overview of software engineering, but do not provide an in-depth coverage of the various stages of the software development life cycle. For instance, their coverage on requirements is restricted to a couple of chapters and was not sufficient for our use. Good books on requirements are hard to come by. Our search led us to consider the following books:

1. Requirements Engineering, Processes and Techniques by Gerald Kotonya and Ian Sommerville (1998) [9]
2. Managing Software Requirements: A Use Case Approach by Dean Leffingwell and Don Widrig (2003) [10]

We decided to use the Leffingwell & Widrig text [10] as our primary text due to the in-depth treatment of the list of topics under consideration for the class and the use of a consistent case study throughout the book to explain the various facets of requirements engineering.

The role of prototyping and user interfaces in removing barriers to requirements elicitation is well documented. Our desire to expose students to interaction design as an integral aspect of requirements engineering led us to consider the following books:

1. Interaction Design: Beyond Human Computer Interaction by Helen Sharp, Yvonne Rogers and Jenny Preece (2006) [17]
2. Designing the User Interface - Strategies for Effective Human Computer Interaction by Ben Shneiderman and Catherine Plaisant (2009) [20]

We adopted the text by Sharp, Preece & Rogers [17] as it provides an in-depth coverage of interaction design and evaluating user interfaces. Additionally, the conversational writing style and the use of examples makes it very approachable to undergraduate students in computer science.

2.5 Learning Model

Gathering requirements is as much art as it is science. There is no magic formula that can be applied to understand the need behind a system and document the features required in a software system. The technique to use depends on several factors including a) the type of system, b) the technical know how of the clients and the end users, c) the geographical location of the stakeholders. It is not enough to teach students the problems with gathering requirements, the techniques used to elicit, capture, organize and trace requirements. Hands-on experience is particularly important to help students understand the finer details of requirements elicitation. With this in mind, we decided to utilize a three-tier learning model to provide numerous opportunities for our students to practice and develop a mastery of these vital skills. The three tier model includes the following:

1. Tier 1 - In-class Elicitation
2. Tier 2 - Elicitation via Homework Projects
3. Tier 3 - Elicitation through Junior Design

Tier 1 - In-class Elicitation

We believe that a student's comprehension of the material is greatly enhanced when they are engaged with the material. Each and every technique / concept covered in the class is coupled with suitable active learning exercises that provide a chance for the student to apply the knowledge gained in the classroom.

A common technique utilized in the classroom is the notion of personas; we combine this with mock projects in the first tier of our learning model.

- Mock Projects: We have used mock projects in the classroom to provide context to our discussions. We have organized the discussion on requirements elicitation around project scenarios, where the teaching assistants and the course instructors play the role of client stakeholders and the student teams act as the software developers. The mock projects provide students the ability to apply the software management and development practices they learn in the classroom. It also helps them identify the techniques that are applicable to various project situations. We provide the students with a brief product statement and the student teams then identify the needs, features and requirements as

we work our way through the various elicitation techniques and other facets of requirements engineering and interaction design.

Computer Science is a rapidly changing field and the project scenarios are selected with an eye towards exposing students to new practices and software technologies. For instance, here is a problem statement we used last year:

A system that integrates micro-blogging services like twitter to run in-class assessments and to gather feedback during class. This could be generalized into a system that integrates with PowerPoint and helps gather student feedback in classes. Faculty can conduct quizzes, gather data via polls, and conduct plus / delta evaluations.

- Personas: A software engineering professional will encounter a wider variety of clients. To give our students a sense of how clients (who think differently and approach problems differently may behave), we developed various persona's [3] (engineers, marketing managers, technophobes and people from different cultures). Students were provided with a description of these persona's and trained faculty played the roles of people with these persona's and students were then required to work in groups to elicit requirements of a very small project. We have enclosed a description of a sample persona:

  Shanice: Went to Indiana University as an undergraduate, majoring in Economics with a minor in Chemistry. Then she started working in the Project Management Department at Eli Lilly in Indianapolis. During her time as manager, Shanice obtained a MS in Information Systems from Kelley School of Business in Indianapolis. She is currently managing a project to change the software for pharmaceutical research and production systems.

Tier 2 - Elicitation via Homework Projects

We identified homework assignments as a valuable resource to provide students with more opportunities to practice and reflect on requirements gathering. We designed a series of assignments on the simple scenario provided below:

Bank de Fleur *was established in 1875 to assist employees of Fleur Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley.* ***Bank de Fleur*** *has ATM's for convenient access at various locations on campus and in several locations downtown and the adjoining*

rural areas.. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers and enhance the overall security.

We enlisted our teaching assistants to represent the bank's management and IT divisions. We also identified some students participating in the senior capstone experience to play the role of the Bank's customers. Each teaching assistant was provided with a list of features and they were instructed to answer only the questions posed by the students.

Feedback was provided by the assistants at the end of an interview session on the student's approach, the questions that were asked and the questions they should have asked. The course instructor provided additional feedback at the very end. The assignments were designed to help the students gain experience in the following areas:

1. Creating a problem statement, identifying stakeholders and constraints
2. Identifying features and tracing features to needs
3. Creating storyboards, data flow diagrams and domain models
4. Identifying actors and creating and refining use cases for the high priority features
5. Creating low fidelity prototypes using post-it notes for the identified use cases.
6. Identifying supplementary specifications such as usability, reliability and performance requirements. Students are also required to identify usability violations on the prototypes from the previous assignment based on the usability requirements and make the necessary changes
7. Developing acceptance test cases based on the identified use cases.
8. Specifying usability and user experience goals
9. Performing usability tests and re-design based on user feedback

Copies of the assignments have been included in the Appendix.

Tier 3 - Elicitation through Junior Design

The Junior design sequence is the final tier of the learning model. The homework assignments and in-class activities were designed to help students apply elicitation and management techniques as they work with real clients on a project. We gather project proposals throughout the year from alumni, from companies that hire our students, from not-for profit organizations and from the campus community. It should be noted that these organizations **do not** pay a fee to participate in the junior design sequence and are appraised of the educational objectives associated with the junior design sequence.

The projects are made available to the students and student opinion on project / team preferences is taken into account in determining the final list of projects. It must be noted that each team (composed of 3 - 4 members) has its own client and project. The team is retained for the entire sequence of classes that comprise the junior design sequence.

Each team is also assigned a teaching assistant who functions as a project manager and provides the team with valuable guidance as they navigate the development lifecycle. Each team has weekly meetings with their client to update them on their progress. Teams also have weekly meetings with their project manager and instructor (if necessary). Agendas and reports are maintained for each meeting, and teams are also required to provide their manager and instructor with weekly stop light reports. The manager has a responsibility similar to a first line manager in a professional situation. Of course, the manager cannot literally fire and hire students, or suggest raises or salary reductions. However, the manager will periodically evaluate each member of the team. This evaluation will be given strong consideration in determining the individual grade for the project.

To help the student teams as they worked on their projects, we divided the work into five distinct milestones as follows

1. Milestone 1: Current System Analysis, Client Stakeholder Analysis, Feature Listing & Problem Statement.

2. Milestone 2: Use Cases, Data Flow Diagrams & Domain Models

3. Milestone 3: Supplementary Specifications:
   (Non-functional requirements) & Initial user interface design

4. Milestone 4: Change Control Plan, Coding Standards & Acceptance Test Cases and Plan

5. Milestone 5: Usability Test Plan, Test Report and Revised Interface Design

Teams were required to make changes as necessary to various milestones to keep the documents up to date with changing requirements. Students were also encouraged to maintain electronic journals as a reflective exercise. We have also experimented with peer reviews of requirements specifications by other project teams and teams from the senior capstone experience. Details of the rubrics used to grade the milestones and the content of the various documents have been included in the Appendix.

Teams were also strongly encouraged to receive feedback on the various milestones from their project managers. We do require teams to turn in a draft to the manager a week prior to the final submission. It must be noted that we had to carefully schedule homework assignments and milestone deadlines to ensure students had sufficient time to receive and incorporate feedback from their homework and in-class exercises on the project.

Case Study Discussions

The final component of our learning model was the use of discussions revolving around case studies. We introduced case studies and papers (pertinent to requirements) from industry/academia that detail software product success stories and failures. Students are expected to prepare a pre-discussion report that is due before the class session. During the class session, we analyze the case study from the perspective of the concepts learned in the classroom and we try to identify the a) rationale behind the failure, b) key reasons behind the success of a strategy and c) mechanisms to improve the software process. These case studies provide the student with an opportunity to apply their knowledge by reflecting on a real world example. Our list of case studies include the following

1. BAE Automated systems: Denver international airport baggage handling system [13]

2. Why (some) large computer projects fail ? - An analysis of the FAA Advanced Automation System [5]

3. When professional standards are lax: The Confirm failure and its lessons [16]

4. Anatomy of the Boston Big Dig project

5. Requirements engineering challenges in multi-site software development organizations[4]

Most of the above case studies can be either purchased through Harvard Business School or can be purchased as a part of the book - *Software Runaways: Lessons Learned From Massive Software Project Failures* [5].

2.6 Evaluation and Discussion

The classroom processes we have outlined here did lead to junior college students delivering substantial projects at the end of their second or third (of three) terms in the school year. Indeed, the majority of these projects were successful, over three years of experience in using these processes in the requirements course. Failures attributable to incorrect requirements were quite low [3 out of 36]; we believe this was because of the quality work the students did in eliciting and managing the requirements in the course described. They showed their client prototypes early and often, so that misunderstandings about requirements were known early. They did a professional job of trying out their human interface designs on real users in the class, which improved their projects' odds of success. We also believe that they learned to follow good practices for maintaining a relationship with their client, so that they were able to relate realistically with them regarding such things as introduction of changes to requirements after they had begun building the system.

This requirements course was invented largely because the school also has a software engineering major. However, since its inception in 2003, the course always has been a required part of the computer science curriculum, as well. Requirements engineering is not easy to teach to undergraduate majors for a variety of reasons which have been noted by others. As Lethbridge, et al. [11] described teaching soft topics to undergrad CS students, "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".

We believe that, if students see clearly the end result of all the work they put into a project, including using their soft skills like requirements engineering, they do appreciate the value of learning these skills. In our junior sequence, they all carry their requirements projects forward into design and development, sufficiently that they can see the importance of the time spent learning to do requirements. That is, we believe the methodology described

inherently helps to overcome the problem of students not valuing such material. Further, the use of case studies and in-class discussions on the role of requirements in the software life cycle and the problems caused due to various failures directly attributed to requirements helped emphasize the vital nature of the material and provided the necessary motivation for the students.

We conducted end of the term anonymous evaluations to gauge student feedback about the course. The students were pleased with the outcomes and felt that they gained a lot from the class. They liked the exposure provided to real world practices through the project and conveyed that their level of learning was higher in this class when compared to other courses. Students felt that the the three-tier model of reinforced learning was instrumental in their ability to understand and apply the concepts covered in the classroom. Over the last three years, 104 students completed the surveys and the course averaged the following:

- Quality of learning 4.12/5
- Use of supporting material to reinforce concepts 4.5/5
- Quality of course 4.01/5

Response from the students and alumni on free-from questions is filled with positive feedback about the quality of learning, material and the course in general. More details are not included due to space constraints, but we will be happy to share the results. We include the following comment to convey the expressed sentiments.

"Course homework and activities enforced the concepts we needed to learn. The project really is the best part of this course. It is really cool to work with an actual client, and it feels like we are doing something useful. Unlike other courses, there is some motivation, besides grades, to actually do well on the project. The structure of the course was excellent"

2.7 Conclusion

To be a science, computer science should be based on a firm foundation; in most cases this is taken to include a clear expression of the problem to be solved, such that it leads to a precise resolution. However, most computer science students enter careers in software development, in areas where, usually, clarity of requirements is fuzzy and changing. As we said, this imprecision is known to be a major culprit in the failure of a high percentage of the

projects they will embark on. Understanding how to interpret these requirements well is analytical work and lies very close to selecting the proper algorithm and to the art of coding, usually felt to be the core parts of the computer science curriculum. By knowing, as well, the art which goes into requirements elicitation and management, students gain a deeper understanding of the meaning of those requirements. We believe that, even if an undergraduate student's intent is not to become an expert in requirements engineering, their learning of how this expression of the problem comes to be, with real customers, and the vagaries of that work, greatly improve their ability to fit into real software organizations upon graduation.

We also believe that the staged learning of the skills in requirements engineering, which we have outlined here, is a good teaching method for the students to transition into this world they will work in, where problem definition is much less predictable or containable than in the traditional classroom.

Bibliography

[1] Standish group chaos: A recipe for success. *Standish Group International*, 1999.

[2] R. Bruhn and J. Camp. Capstone course creates useful business products and corporate-ready students. *ACM SIGCSE Bulletin*, 2004.

[3] A. Cooper. *The inmates are running the asylum*. SAMS publishing, 2004.

[4] D. Damian and D. Zowghi. Requirements Engineering challenges in multi-site software development organizations. *Requirements engineering*, 8(3):149–160, 2003.

[5] R. Glass. *Software runaways*. Prentice Hall, 1998.

[6] R. L. Glass. How not to prepare for a consulting assignment, and other ugly consultancy truths. *Commun. ACM*, 41(12):11–13, 1998.

[7] R. L. Glass. Frequently forgotten fundamental facts about software engineering. *IEEE Softw.*, 18(3):112–111, 2001.

[8] H. Koppelman and B. van Dijk. Creating a realistic context for team projects in HCI. *ACM SIGCSE Bulletin*, 2006.

[9] G. Kotonya and I. Sommerville. *Requirements engineering*. Wiley Chichester, 1998.

[10] D. Leffingwell and D. Widrig. *Managing Software Requirements: a use case approach*. Pearson Education, 2003.

[11] T. Lethbridge, J. Diaz-Herrera, J. Richard Jr, and J. LeBlanc. Improving software practice through education: Challenges and future trends. 2007.

[12] L. Macaulay and J. Mylopoulos. Requirements Engineering: an educational dilemma. *Automated Software Engineering*, 2(4):343–351, 1995.

[13] R. Montealegre, H. Nelson, C. Knoop, and L. Applegate. BAE automated systems (A): Denver International Airport baggage-handling system. *Harvard Business School Teaching Case*, page 311, 1996.

[14] T. Moynihan. How experienced project managers assess risk. *IEEE Softw.*, 14(3):35–41, 1997.

[15] J. Myers Jr. Software engineering throughout a traditional computer science curriculum. In *Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference*, 2000.

[16] E. Oz. When professional standards are lax: the CONFIRM failure and its lessons. *Communications of the ACM*, 37(10):29–43, 1994.

[17] J. Preece, Y. Rogers, and H. Sharp. Interaction design: beyond human-computer interaction. 2006.

[18] R. Pressman. *Software Engineering - A Practitioners Approach*. McGraww Hill, 2009.

[19] K. D. Schenk, N. P. Vitalari, and K. S. Davis. Differences between novice and expert systems analysts: what do we know and what do we do? *J. Manage. Inf. Syst.*, 15(1):9–50, 1998.

[20] B. Shneiderman and C. Plaisant. *Designing the user interface - Strategies for Effective Human Computer Interaction*. Addison-Wesley Reading, MA, 2009.

[21] I. Sommerville. *Software Engineering*. Addison Wesley, 2010.

[22] J. Verner, K. Cox, S. Bleistein, and N. Cerpa. Requirements engineering and software project success: An industrial survey in australia and the u.s., 2004.

[23] E. Walker and O. Slotterbeck. Incorporating realistic teamwork into a small college software engineering curriculum. *Journal of Computing Sciences in Colleges*, 2002.

Course Syllabus

Information Packet

CSSE 371
Software Requirements and Specification
Fall 2010

Computer Science and Software Engineering
Rose-Hulman Institute of Technology

CSSE 371 – Software Requirements and Specification - Fall 2010

Computer Science and Software Engineering 371
Software Requirements and Specification
Fall 2010

Instructor: Sriram Mohan

Office: Moench Hall, Room F226

Phones: 877-8819 (Office) ; 812-219-9658 (Cell)

Email: mohan@rose-hulman.edu

Office Hours: I usually live in Moench Hall, Just drop by when you need to see me.

Project Managers:

Eli Baca
Tim Ekl
Mark Jenne
Eric Stokes

371 Course Meeting Times:

(Section 1) 8th period MTRF – Olin 169
(Section 2) 9th period MTRF – Olin 169

371 Course Prerequisite:

CSSE 230 (Fundamentals of Software Development III) or equivalent
RH 330 or equivalent
Junior standing

371 Course Description: Basic concepts and principles of software requirements engineering, its tools and techniques, and methods for modeling software systems. Topics include requirements elicitation, prototyping, functional and non-functional requirements, object-oriented techniques, and requirements tracking.

371 Course Outcomes: Students who successfully complete the course should be able to:

1. Explain the role of requirements engineering and its process.
2. Formulate a problem statement using standard analysis techniques.
3. Determine stakeholder requirements using multiple standard techniques.
4. Produce a specification with functional and non-functional requirements based on the elicited requirements.
5. Decide scope and priorities by negotiating with the client and other stakeholders.
6. Manage requirements.
7. Apply standard quality assurance techniques to ensure that requirements are: verifiable, traceable, measurable, testable, accurate, unambiguous, consistent, and complete.
8. Produce test cases, plans, and procedures that can be used to verify that they have defined, designed and implemented a system that meets the needs of the intended users.
9. Design and Prototype user interfaces to validate requirements.

CSSE 371 – Software Requirements and Specification - Fall 2010

10. Prepare and conduct usability tests to evaluate the usability, utility and efficiency of the developed user interface.

371 Course Texts (Both Required):

- Managing Software Requirements: A Use Case Approach, Second Edition, by Dean Leffingwell and Don Widrig
- Interaction Design: Beyond Human-Computer Interaction, Second Edition, by Jennifer Preece, Yvonne Rogers and Helen Sharp

Course Evaluation and Feedback:

Please feel free to provide me feedback about the course at any time. Also, an anonymous feedback box under the ANGEL account for this course will be available for feedback throughout the course; I typically check it once a day and will respond to feedback during the next class session. There will also be two anonymous plus-delta evaluations of the course.

I recommend that you keep a "course evaluation log" somewhere to make notes that you can use for the course evaluation at both midterm and the end of the course.

Course Average Determination:

50%	Software Team Project Work (details below)
20%	Exams
20%	Homeworks & Case Studies
10%	Class Participation (including attendance and quizzes)

Team Project Work Breakdown:

10%	Manager's Evaluation
5%	Clients' Evaluation
10%	Peer Evaluations
10%	Weekly Summary Reports
60%	Other Project Artifacts
5%	Project Presentations

Note: Our assessment of your contribution to your team, and your team's peer evaluation of that contribution, becomes a fudge factor in final grades. This can be a significant adjustment in the positive or negative direction. Thus it is possible for a student to get a low grade even if their team does an exceptional job and vice-versa.

Course Grade Division:

90-100	A
85-89	B+
80-84	B
75-79	C+
70-74	C

CSSE 371 – Software Requirements and Specification - Fall 2010

65-69	D+
60-64	D
0-59	F

Exam Policy:

Exams will be in-class, closed book, and closed notes except for one 8.5 by 11 sheet of paper which you can put notes on using both sides of the page. No exams will be "dropped". If you have a conflict with a scheduled exam, you should notify me immediately. Giving a makeup exam for an unexcused absence is at the discretion of the instructor. Any requests for re-grading must be made in writing by the beginning of the next class period after the exams are returned.

Homework Grade:

There will be approximately one homework assignment each week. The homework's include a variety of tasks that will help each student prepare for the various milestones in the project. The homework will be of great help as you work on the project and can significantly affect the final grade. Unless otherwise stated, all homework assignments are to be done independently. All homework assignments are to be submitted to the Angel drop box by 11:55 PM on the day it's due.

Case Studies:

We will be doing case studies on most Fridays. You should prepare (and submit) a write-up of your opinion (and any questions as stated) of the case study. The pre-discussion write-up should be no more than 1/2 page (approximately 1 paragraph). It will be due before class (2:30 PM Friday).

After the in-class discussion, you are to write how your opinion on the case study changed and/or why it stayed the same. The case study follow-up should, again, be no more than a 1/2 page (approximately 1 paragraph). It should be contained within the same document as your pre-discussion write-up. It will be due to Angel Monday at 8:00 AM.

Ethics and Professional Practice:

You are expected to act honestly and professionally in this course at all times, in a manner consistent with the schools honor code.

Class Participation Policy:

There are 40 meeting times during the term. You can potentially receive 10 points towards the class participation portion of your grade for each of those classes in the following fashion:

- If there is a quiz during class, you can earn up to 10 points on it.
- If there is no quiz during class and you attend and make an effort to participate (since with a small class there will be lots of discussion), you will earn 10 points.
- If the class for some reason does not meet, you automatically receive 10 points.

Attendance Policy:

CSSE 371 – Software Requirements and Specification - Fall 2010

Up to 2 unexcused absences allowed. In accordance with the Rose-Hulman attendance policy, additional unexcused absences may result in you receiving a failing grade for the course. You are responsible for making up any missed work.

Laptop Policy:

You will generally need to use your laptops during at least a portion of every class period. Please be sure to bring your laptop, a power brick, and a network cable to class.

During class discussion, please do not use your laptops. Laptop use during discussions is distracting to your classmates and also keeps you from focusing on the material. If you typically use your laptop for note taking, please talk to your instructor so he can make an exception.

Collaboration:

You are encouraged to discuss the homework and other parts of the class with other students. Such discussions about ideas are not cheating, whereas the exchange of code or written answers is cheating. However, in such discussions of ideas, you should distinguish between helping and hurting yourself and the other student. In brief, you can help the other student by teaching them, and you can hurt them by giving them answers that they should have worked out for themselves. The same applies to tutoring and getting help from the instructor.

"Hurtful help" most commonly occurs in giving away a key idea needed to solve a problem. For example, suppose you have studied a programming problem for an hour or so and finally found that the key to the solution is to use a helping procedure you call "critical". Your friend, after working on the problem for 15 minutes, says "I just can't see how to do this" and you say, "try using a helping procedure called 'critical'.

Although it takes more time, your friend will learn more if you say something like: "How are you approaching the problem, what's your plan?" (Knowing that if your friend is not planning, no helping procedure will be found). If your friend hasn't planned, you should let them do it; if they have trouble planning, tell them to think about problems discussed in class that were similar, etc. If, after planning, your friend still hasn't found helping procedure 'critical', you should say something more direct like, "what helping procedures do you have?" or "how do these helping procedures help you get closer to the solution?" or "can you solve part of the problem?" The idea is to guide the other person's thinking process.

Perhaps a more common way to fall into the hurtful exchange of giving away the key idea is when you're talking over a problem that no one knows the answer to yet. Once one of you comes up with the key idea, it is tempting to blurt it out, impressing the others with your brilliance. If this happens, you should write "developed in cooperation with ..." on your solution. (Note that this disclaimer cannot be used to get away with cheating, but we're not discussing exchanging written code or answers.) It would be better for the one who comes up with the key idea say "I have it, but now I can't tell you what it is" and then try to guide the others to the solution as described above.

If you use reference materials (other than the course texts) to solve a problem, please cite the referenced material. Similarly, if you discuss a solution with another student, give credit where credit is due by making a note such as "the following idea was developed jointly with Alyssia P. Hacker," or "the following idea is due to Ben Bittwiddle." You cannot be charged with plagiarism if you cite

in this way. (However, don't expect to excuse cheating with such a citation. That is, you cannot exchange code even if you say it was developed in cooperation with someone else. Cooperation refers to the exchange of ideas, not code or written answers.)

General Writing Issues:

Written communication is important in this course, as it is in the profession in general. Remember that a software document has several unique and important characteristics:

1. Technical documents are often the result of group authorship, thus it requires planning and final tweaking.
2. Specificity and organization are more important than flow, hence technical documentation is often ordered around lists and tables rather than paragraphs.
3. Documentation is often the reader's only source of information on the particular subject or product, hence it must be thorough and complete.
4. Documentation is often used to answer a specific question; hence it should facilitate finding a specific piece of information (navigation).
5. Documentation must bridge from general specifications to particulars of implementation and operation, hence it must make abstract concepts concrete and make concrete facts fit generalized concepts.
6. Documentation can be presented in many forms: online via HTML, MS help files, just plain text, and on paper as reference manuals, tutorial, quick reference guides, etc. It is important to choose the correct medium and even more important to write to fit the medium.

You can always drop by my office or consult with your project manager, if you have any questions regarding your document. **We would be happy to look at it and suggest some changes**. You should also be aware of the service provided by the Learning Center.

Late Submissions:

Late quizzes and case studies will not be accepted. Homework assignments, and Project milestone deliverables will also not be accepted late, with the following exception:

You have four "late day" credits. You may use one of them on any Homework assignment, or Project deliverable, which will allow you to submit that assignment up to 24 hours after the due time. Homework's or project assignments, which are more than 24 hours late, will receive a deduction of at least 10% per late day (or not be accepted at all), depending on the circumstances and the degree of lateness.

You may earn a maximum of one additional "late day" by submitting an assignment or a project deliverable 24 hours before the due date. Please send me an email alerting me to the same to obtain the "late day" credit.

If you submit something late for which late day credits are allowed, I will assume that you want to use one of your credits unless you tell me otherwise.

Project Grade:

CSSE 371 – Software Requirements and Specification - Fall 2010

The various artifacts you will produce, as a part of the project will be organized into milestones. There is no set scale or weighting for individual milestones. They are there to give you concrete immediate objectives, valuable feedback and metrics for evaluating your progress. The success of a team depends on the contributions of each and every team member; a member who does not participate in and contribute to his or her team project can be removed from the team on the recommendation of the project manager.

Project Schedule and Deliverables:

During the fall quarter, the teams will interact with their respective clients to elicit requirements, define the system, and perform traditional project management activities. In addition, each team will also produce a test plan to verify that the developed system meets the user needs. The team will also produce a usable and effective user interface and verify the same via usability tests.

Deliverable	Contents	Due Date
Milestone 1	Individual Engineering Journal • Current System Analysis • Client Stakeholder Analysis • Feature Listing • Problem Statement	September 24
Milestone 2	Individual Engineering Journal • Use Cases • Data Flow Diagram	October 8
Milestone 3	Individual Engineering Journal • Supplementary Specification • Initial Design/Paper Prototype	October 19
Milestone 4	Individual Engineering Journal • Change Control Plan • Coding Standards • Test Cases	October 29
Milestone 5	• Initial Interface Design • Usability Report • Final Interface Design	November 9
Final Deliverable	Individual Engineering Journal • Final Updated Versions of all milestones transitioned to new team (includes	November 12

CSSE 371 – Software Requirements and Specification - Fall 2010

	signoff) • Client Comments • Lessons Learnt	
Client Presentation		November 11,12
Post Partum Presentation	Experiences ...	TBA

Deliverables in *italics* are due a week before to the project manager. Milestones 3 and 5 are due Oct 10th and Nov 5th to the project manager while the presentations are due two days before to the project manager.

During the winter quarter, the teams as a part of CSSE 374 will develop a detailed design document for the system. Using the detailed design document as a guide, the team will complete implementation of the system. The team will perform acceptance, unit and integration testing (based on the test plan defined in the fall quarter) to verify the system. Usability tests will also be performed to check for ease of use and other factors that determine overall usability of the user interaction.

Individual Engineering Journal:

A. What's an engineering journal?

Engineers and designers use these to write down ideas and to document what they did. They always have dated entries, because they sometimes are used as "proof" of who thought of an idea first, like in patent applications.

At NCR and at Bell Labs, these were hardbound books, and we always wrote project and meeting notes in these, as well as our own ideas about new stuff. The big joke was that, at a meeting, the engineers would all pull out an engineering journal, and the managers would pull out a day planner, and this was how you knew who was who! Anyway, in design meetings, we documented who said what and how decisions were made, which was very important to have as a reference.

Now that we're in the post-modern age, you probably want to try doing such a journal electronically (via twitter, a project blog, or as part of project management system such as Trac).

We've done these for a long time in other SE classes, and I'll show a couple examples of contents below.

B. How will we use these in class?

For 371, what I'd like you to do is the following:

1. Start your own personal journal. (I.e., One per person.)
2. Use it to record activities related to your *team project*.
3. Keep it up to date soon after team activities, if not "during."
4. Turn it in during every milestone, so I can look at it and/or grade it as a part of your work. Turn in *the whole thing* every time.
5. At the end of the course, use it to consider what was most important to you, etc.

What I will look for in the journal:

CSSE 371 – Software Requirements and Specification - Fall 2010

1. It describes what you contributed personally to your teams' project that week.
2. It discusses team interactions, from your perspective. E.g., what happened in team meetings?
3. It describes new ideas you had, as these came to mind.
4. It also describes new problems you thought of, related to your team's project.
5. It says from your perspective how the team made decisions and dealt with action items. E.g., How did the team decide the look and feel for the user screens?
6. It reflects on earlier decisions – later entries should look back on what you did before.
7. It talks about plans for completing the project, like how risks are being handled, and what you plan to do about it.

C. Examples of journal entries from earlier SE classes:

Today we decided to use a map-based mental model for the interface. This model will focus the interview questions that are asked before the software is shown. It also has an impact in how the software will be built.

Determined that a user must be notified of an incoming request, so created a method DisplayNotification which takes a type and a user (both Strings). The arguments tell what type of message to display and the user the notification is coming from. By using the notification, a user may accept or reject a file transfer.

As a consequence, the DisplayNotification has been made to accommodate for other types of dialog messages for any future messages to be displayed to the user. Also, I will create another DisplayNotification, which only takes a type to accommodate for notifications, which do not come from another user. The DisplayNotification method allows for the creation of multiple types of dialog message to display to the user.

We interviewed several users for the system. We used our people we knew who were students at Rose. We used a variety of majors so we got a variety of technical skill levels. We picked BMEs, MEs, Math, Econ, etc. The interviews went smoothly. After asking the pre-test questions, we gave a quick overview of the application concepts and we let them loose. Then, we asked the post-interview questions. For three of the participants, I wrote down the participants' answers and typed them up afterwards. Jeremy took notes for the other three.

Wednesday, April 9, 2008
-Use cases:

-Considered administrative use cases, but could not think of any.
-The system does not need to be installed, it does not have any excess files, and all in game items are user-level.
|The only thing administrative would be distribution of the files, which seems excessive to put into a use case.
-Remaining documents were updated as seemed appropriate. Most of the documents were kept up to date locally.
|Final modifications were made to get them up to par with this project.

- Ran into timing problems as an overlooked project overlapped with time slotted for this one.

Dear Journal,
We discussed today that it was ok to do page redirects within the View, as doing so does not interact with the Model in any way. Also, we originally had the Model sending redirect requests, but determined that these requests should be sent to the Controller first and the Controller should be the one doing the redirecting.

Prior to conducting the interviews, it was not decided if the kitty should be thrown to the left side or the right side. However, as the interviews revealed, all users seem to unconsciously prefer throwing the kitty to the right side.

CSSE 371 – Software Requirements and Specification - Fall 2010

It should be noted that all interviewed users are right handed, and have also played similar implementations of this game. It could be that right-handed individuals simply expect objects to travel across the screen from left to right. Or it could also be that the users are all unfairly biased from having played similar games before.

Project Planning:

In order to complete the project successfully, it is necessary for the team to work on several tasks at the same time. Each task has a significant lead-time – for contacting a client, reviewing a document, or simply careful deliberation. Do not expect to complete a milestone if you haven't started working on it at least a couple of weeks before it is due. All of this requires that steps be planned and that the plan be monitored. The project schedule should help you get started on a plan. While preparing the project plan (you will be doing this as a part of CSSE 372), you should outline a schedule that will help you meet the various milestones. Parallel schedules must be developed where necessary.

Project Teams:

You will be assigned to a *Project Team* by the 8th of September. A list of project choices will be available by the 3rd of September. Based on your choices (use the survey available on Angel), you will be assigned into teams by the Instructor and the Project Managers. Team assignment will be done with an eye towards providing each team with the necessary balance of skill and capabilities. From time to time, the management might reassign an individual from one team to another, based on changing estimates of man power needs and team abilities. Very rarely, a team member might be "fired" using the mechanism described below.

Project Manger and their Responsibilities:

Each team will have a project manager. The project managers will be seniors who have taken the CSSE 371/372/374/375 combination earlier. They will provide valuable help to the teams.

Although a manager will often give a team technical guidance, their one specific responsibility is assisting the team with the process. To this end, each team must hold a "project manager meeting" each week, where process related matters are reviewed. Please note that the manager is not the one in charge of these meetings – in fact, the team is still responsible for the conduct of the meeting. The most important process matter is project planning, as reflected in the **Weekly Assessment Report** section below; Of course technical matters are often covered at the meeting; it is a good occasion to hold a "walk-through" of some topic. Furthermore, the team often will need to meet more than once a week – perhaps with the manager, with the clients, with the instructor or with other stakeholders.

The manager has a responsibility similar to a first line manager in a professional situation. Of course, the manager cannot literally fire and hire students, or suggest raises or salary reductions. However, the manager does have at least two ways to encourage performance by the team members.

First, the manager will periodically evaluate each member of the team. This evaluation will be given strong consideration in determining the individual grade for the project. Since the project is a very significant portion of the final grade, this should provide incentive for all team members to perform.

A second, more drastic mechanism is available to the manager if a student is unable or unwilling to work appropriately as a team member. The manager can recommend that the instructor "fire" the

student. The instructor will carefully evaluate the student's performance in the context of the team and, if warranted remove the student from his/her team, in effect firing the student from the project. Being fired from a project is sufficient grounds for failing the course unless appropriate remedial action, as agreed with the instructor, is taken. For instance, a "fired" student can be given an appropriate individual project assignment by the instructor. The grade for the replacement project will be reduced by one letter grade, so that the maximum grade is a B. A "fired" student can try to get "hired" by another project manager, but the student will have to make a very good case to succeed.

Client Feedback

I will be soliciting feedback from your client at midterm and at the end of the term. **You will not get above a D in this class if you do not deliver a satisfactory product to your client!**

Client Meetings

A key to you gaining satisfactory feedback from your client is a regularly scheduled meeting. We expect to see an agenda (submitted at least 24 hours in advance) and meeting minutes.

Peer Feedback

Also at midterm and the end of the term, we will be soliciting feedback from your other team members. We expect feedback given here to be constructive.

Weekly Assessment Reports

Every Friday at noon, you should submit (as a team) a weekly status report. Status reports are to follow the template (this will be the only template you'll be given for this class) shown below, should be signed, and delivered in hardcopy to your instructor. These will be graded based on adherence to the template, completeness, accuracy, and content. An updated copy of the Weekly Status Report must be sent to the project manager 12 hours in advance of the "project manager meeting".

The purpose of the report is to record the following in a consistent manner and place: what tasks must be done, when they must be begun and finished, who is responsible for accomplishing them, and where they stand with respect to their completion.

Status Report for Team _____

Week **:** <Use Rose Week>

Last week's status: <Green, Yellow, Red>

Current project state: <Green, Yellow, Red>

Tasks completed last week: <bulleted list of high-level summary of tasks and who performed them>

Tasks to be completed next week: <bulleted list of high-level summary of tasks and who's to perform them>

CSSE 371 – Software Requirements and Specification - Fall 2010

Key issues/problems: <list of key issues/problems of concern, mitigation strategy, and revisit date>

Weekly metrics:

Team Member Name	Actuals from Previous Week	Variance for Last Week	Forecast Time for Upcoming Week	Totals to Date	Signature
One for each team member	In Minutes	(Negative in Parentheses) In Minutes	In Minutes	In Minutes	Sign here

Team Roles:

Roles are associated with ongoing or regularly recurring responsibilities. In some sense a role is just a continuing task, but roles are typically associated with process matters while tasks are more likely to work towards products. Of course, there are many more roles than there are team members; this necessitates that each team member may be assigned several roles. For example, the various liaison roles may be combined and assigned to one person. Sometimes a role is done collectively by the entire team and thus assigned to the team; only a few roles such as task assignment are suitable for the collective responsibility.

The project roles are:

- *Secretary:* Takes minutes during meetings and communicates them to all concerned parties.
- *Manager/Instructor Liaison*
- *Client Liaison:* It is important that all team members meet with the client for information gathering, presentations and related activities. However, unless there is one person who schedules meetings, makes commitments etc., the client (and the team) is likely to get confused.
- *Task Assigner/Monitor:* Schedules, assigns and monitors tasks based on deliverables and activities identified by the team.
- *Convener:* Calls meetings, organizes intra-team communication.
- *Librarian:* Maintains a repository of documents and programs; monitors configuration. Particularly relevant during implementation.
- *Guru:* Toolkit/Language/Platform Specialist. The guru should teach the tool(s) to other team members. Occasionally this role is split to address multiple tools.

There are many ways to organize a small team, but the most common is to have a single leader who makes all the key decisions and divides up the work for the other team members. A more democratic organization in which the team members share the work can also be successful, but the team must be sure that all the key responsibilities are actually met. You are free to choose your own organization, but you are advised to consult with your project manager.

In any case, you must designate one person each as an Instructor/Manager Liaison, a Client Liaison and a Secretary (these are not necessarily different people and the identity of the individuals in these positions may change over time). The contacts serve as the primary coordination channel between the team and the Instructor or the Manager respectively.

CSSE 371 – Software Requirements and Specification - Fall 2010

Document Contents:

All milestone documents produced for this class must include the following sections

1. *Title page:* **This page must contain the team's name and the document title, and must have the signatures of all the team members.** A team member's signature indicates that they have read and approve the contents of the entire document and not just the portion they were assigned.
2. *Table of Contents:* Every section and sub-section in the document along with the page numbers should be mentioned here. If you are using Microsoft Word, use the Table of Contents wizard. If you are using Latex, this is automatically generated for you.
3. *Executive Summary:* This will be the first section in the document and must briefly summarize the contents of this document.
4. *Introduction:* This will be the second section of the document and will state the purpose of the document and its relation to the rest of the milestones and the software development life cycle.
5. *Milestone Specific Content:* This will be the main content portion of the document. Follow the template given in the book for the various milestones. The book does not have templates for a Usability Report. A Sample report will be made available on Angel.
6. *References:* Any paper, website or any other external resource you might use to write your milestone document has to be duly referenced. If you are using Microsoft Word, we recommend the use of EndNote. If you are using Latex, you can achieve similar results by using bibtex.
7. *Appendix:* Any extra information that improves the overall readability of the document.
8. *Index:* An index of popular words and the pages they are mentioned.
9. *Glossary:* A list of words (that might be new to the reader) along with their definitions.

Deliverable Submission:

You will submit your milestone documents using a large three-ring project binder. You will be required to submit successive milestones in the same binder, so that each is in the context of the preceding and the entire history of project milestones is collected together. To reduce bulk and save trees, you are encouraged to print all documents duplex.

We expect the report to be free of spelling and grammatical errors. All documents to be turned in should be written in a professional manner. **Each team member has to take ownership and read and approve the complete document (and not just the parts for which they were responsible) for submission.** As the instructor, I would be happy to review any portion of the document at any time and suggest changes.

Client Presentation:

The last few days of the class will be devoted to project presentations. Each team will have approximately twenty minutes to describe the product that they intend to produce. There are several reasons for giving presentations. First of all, they give you a chance to exercise your communication skills. You will find that whatever kind of job you choose; you will have to present your ideas to others at one time or another. Secondly, they help the class as a whole since everyone learns about a large number of projects. Finally they provide a basis for comparing the teams for grading purposes.

CSSE 371 – Software Requirements and Specification - Fall 2010

Your presentation should be based mostly on the Requirements that were elicited during client interviews. It should explain (i) who the client is (ii) why the product is desirable and useful (iii) what requirements the client has iv) initial interface design and v) usability report and the final interface design. You should plan to do a professional job. It will help if you imagine the presentation as a pitch for a client company to invest resources for your project. Remember that you only have a limited time – you will be cut off if you run long and this will hurt your score. It is a good idea to practice your presentation several times before you get up in front of the class.

In targeting the audience of the presentation, imagine that you are speaking to managers who are savvy but not necessarily familiar with the intimate details of technology. You are encouraged to invite your clients to the presentation or to repeat it for them at their site. The instructor might invite other faculty members in the department and at Rose-Hulman Ventures to sit in on the presentation.

For pedagogical reasons, every member of the team is required to participate in the presentation. It is not necessary that every member talks for exactly the same length of time, but everyone should have something meaningful to say, and everyone should have a designated part of the product that they will be responsible for answering questions about. No written report is necessary for the project presentation.

Post partum Presentation[1]:

This presentation gives everyone a chance to benefit from what you have learned during this course and you a chance to learn from others too. We learn most effectively by facing and overcoming problems. If, upon attempting something for the first time, everything goes well, we are likely to assume that the task is easy or that we are especially skilled. This may lead to a casual attitude which could be costly later. On the other hand, if we have trouble, we are more likely to be aware of the pitfalls and avoid them the next time. This enhances the affect by calling our attention to the problems we faced. The intent of the post partum analysis is then NOT to find fault but rather to avoid pitfalls in the future. It is an opportunity for your team to sit down and analyze its mistakes and accomplishments.

Your class presentation should describe your most significant unanticipated problems, both avoidable and unavoidable. Explain how you overcame the problems and how costly it was in terms of time. Suggest how to avoid such problems and assess the risk of the unavoidable problems. Your written analysis should be similar to your oral presentation, but should be more detailed. The written report should be turned in directly to the instructor.

If you so desire, you can also use this presentation to talk about the changes you desire in the class - things that did not go well and have to be changed the next time the course is taught, and things that went well and you would like to see continue.

[1] This presentation is often labeled "post mortem" ("after death" in Latin), reflecting the fact that the process has been completed; however, "post partum" ("after birth") reflects the fact the process has resulted in the delivery of project specifications.

Homework Assignments

Computer Science and Software Engineering 371
Software Requirements and Specification
Fall 2010
Homework #1

Due: Tuesday September 14 at 11:55 PM
Purpose: To gain experience in creating a problem statement, identifying stakeholders and constraints

In the remaining homework assignments, we will be working on an example scenario that is intended to help you with the team project. Your Project Manager will be (role) playing various stakeholders. Students at Rose-Hulman will act as account holders, while your Project Manager represent the Bank. Please make sure that you talk to a couple of students and interview your Project Manager, before you turn in the assignments. Please turn in your homework (.pdf file) to the 371 Homework #1 Drop Box in ANGEL. Please remember that all documents must be written in a professional manner.

"***Bank de Rose***" was established in 1875 to assist employees of Rose Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley. "***Bank de Rose***" has ATM's for convenient access at various locations at the Rose-Hulman campus and in several locations in downtown Terre Haute. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers.

For Homework #1, you have to turn in

1) A problem statement for this software, using the format we discussed in class. Note that a problem statement includes an elevator statement

2) A fish bone analysis of the main problems from
   a.) The customers perspective
   b.) The bank's perspective.

3) Make a table of at least 10 different stakeholders for an Automatic Teller Machine. Remember that people with different roles in an organization are considered different stakeholders.
   a.) For each stakeholder, write a sentence describing his or her concerns about the ATM.

4) What are the possible constraints that can be imposed on the ATM software? Use the template from Table 5-5(pg56) as an example.

Computer Science and Software Engineering 371
Software Requirements and Specification
Fall 2010
Homework #2

Due: Friday September 17 at 11:55 PM
Purpose: To gain experience in identifying features

In the remaining homework assignments, we will be working on an example scenario that is intended to help you with the team project. Your Project Manager will be (role) playing various stakeholders. Students at Rose-Hulman will act as account holders, while your Project Manager represent the Bank. Please make sure that you talk to a couple of students and interview your Project Manager, before you turn in the assignments. Please turn in your homework (.pdf file) to the Homework #2 Drop Box in ANGEL. Please remember that all documents must be written in a professional manner.

"***Bank de Rose***" was established in 1875 to assist employees of Rose Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley. "***Bank de Rose***" has ATM's for convenient access at various locations at the Rose-Hulman campus and in several locations in downtown Terre Haute. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers.

For Homework #2, you have to turn in

1) Make a table of the features you have identified for an Automatic Teller Machine.
   a.) For each feature, describe it using the table format discussed in the lecture. See Pg 100.
   b. Correlate the specific client that is satisfied by each feature.

Please turn in your homework to the 371 Homework #2 Drop Box in ANGEL.

CSSE 371 Software Requirements and Specification
Fall 2010
Homework 3

Due: Thursday September 23 2010 (Hand it in during Class)

In the remaining homework assignments, we will be working on an example scenario that is intended to help you with the team project. Your Project Manager will be (role) playing various stakeholders. Students at Rose-Hulman will act as account holders, while your Project Manager represent the Bank. Please make sure that you talk to a couple of students and interview your Project Manager, before you turn in the assignments. Please remember that all documents must be written in a professional manner.

"***Bank de Rose***" was established in 1875 to assist employees of Rose Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley. "***Bank de Rose***" has ATM's for convenient access at various locations at the Rose-Hulman campus and in several locations in downtown Terre Haute. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers.

Part A

Purpose: Practice creating storyboards

1. **Reusing storyboards:** On the Web, find a suitable storyboard about an ATM stakeholder. Using this image you find, which you also should turn in, tell your own story about that stakeholder's interactions with the system. If necessary, replace dialogue shown with your own. For example, if we hadn't already used it, you might've stolen this image from

http://www.cartoonstudio.co.uk/Pages/AbbeyNatATM.html, and made up a story about the second guy stealing the first guy's secret number, which is not what this is really about. And I probably could think of things for the character at *right* to say!

3. **Making your own storyboards:** Drawing them yourself, create a set of 3 "passive" storyboards for the **Bank de Rose project**, which show a stakeholder performing some key action, in a way which graphically brings home the essence of that action. (This art will not be graded, *per se,* beyond its needing to be recognizable as telling the story. However, to be fair to any artists, fab artwork will get a bonus!)

4. **Documenting the story:** After you draw them, describe the story they tell about your stakeholders, using the 3 essential elements of a storyboard.

5. **Stretch the story:** As you did with the stolen artwork, use the storyboard in a different way. This time, show it to someone else (your roommate, say), and ask them to describe what it means without seeing what you wrote about it. Include a synopsis of what they said in the work you turn in.

6. Hand in both sets of art work and their associated descriptions. We are assuming here that you'll physically hand in this homework, rather than having to scan it in for Angel.

Part B

Purpose: Practice creating DFD's

1. Create a context DFD for the main functions offered by the ATM interface

2. For at least 2 functions (Withdraw and Deposit for instance) provide Level 0, 1 and Level 2 diagrams.

3. Use the naming conventions that we discussed in the lecture. Write a sentence or two to describe each DFD.

Computer Science and Software Engineering 371

Software Requirements and Specification

Fall 2010 Homework #4

Due: Monday September 27 In Class (No Late Days)

Purpose: Practice Creating Use cases.

In the remaining homework assignments, we will be working on an example scenario that is intended to help you with the team project. Your Project Manager will be (role) playing various stakeholders. Students at Rose-Hulman will act as account holders, while your Project Manager represent the Bank. Please make sure that you talk to a couple of students and interview your Project Manager, before you turn in the assignments. Please turn in your homework (.pdf file) to the Homework #4 Drop Box in ANGEL. Please remember that all documents must be written in a professional manner.

"***Bank de Rose***" was established in 1875 to assist employees of Rose Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley. "***Bank de Rose***" has ATM's for convenient access at various locations at the Rose-Hulman campus and in several locations in downtown Terre Haute. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers.

The management at "Bank de Rose" has decided that the following features are critical for the proposed redesign.

a) Secure Authentication
b) User must be able to Withdraw Money from their various accounts at the bank.
c) User must be able to transfer money between their various accounts at the bank.
d) User must be able to determine the account balance for their various accounts at the bank.

For Homework 4 Please answer the following questions for any 3 of the above features. Please remember that ease of use is an important concern for all 4 features.

1. Identify all the actors for the ATM.

2. Identify all the use cases for the ATM.

3. Describe each use case. That is, list the sequence of events describing the basic flow and list the sequences of events for alternate flows. Please include a good description, pre and post-condition for all use cases.

Computer Science and Software Engineering 371

Software Requirements and Specification

Fall 2010 Homework #5

Due: Friday October 1 in class

Purpose: Practice Creating Paper Prototypes.

In the remaining homework assignments, we will be working on an example scenario that is intended to help you with the team project. Your Project Manager will be (role) playing various stakeholders. Students at Rose-Hulman will act as account holders, while your Project Manager represent the Bank. Please make sure that you talk to a couple of students and interview your Project Manager, before you turn in the assignments. Please remember that all documents must be written in a professional manner.

"***Bank de Rose***" was established in 1875 to assist employees of Rose Polytechnic to manage their money and investments. It has now grown to be one of the most successful banks in the Wabash Valley. "***Bank de Rose***" has ATM's for convenient access at various locations at the Rose-Hulman campus and in several locations in downtown Terre Haute. They have recently released plans on redesigning and updating their ATM software to enable a better user experience for their customers.

The management at "Bank de Rose" has decided that the following features are critical for the proposed redesign.

a) Secure Authentication
b) User must be able to Withdraw Money from their various accounts at the bank.
c) User must be able to transfer money between their various accounts at the bank.
d) User must be able to determine the account balance for their various accounts at the bank.

Please remember that ease of use is an important concern for all 4 features.

1. Create paper prototypes for the various user screens for any 2 of the above features for the ATM. Each major screen should have its own base sheet of paper. Window objects (widgets) should be placed on the screen using post-it notes or a similar mechanism.

2. For at least 1 features provide an alternative design. That is, the alternatives are not simply rearrangements of the same widgets.

3. Hand in the paper prototypes. Be sure to put your name on the back of each sheet of paper and the back of each moveable widget.

CSSE 371 Software Requirements and Specification
Fall 2010
Homework 6

Due: Tuesday, October 5 2010 – In Class

Purpose: Practice specifying supplementary specifications.

Note: This is a team assignment. If your team does not have enough time to meet to work on the assignment, you can work on it individually.

What to do:

1. You can use any of your teams' 371 - Homework 5 submissions to complete this assignment.

2. Specify usability, reliability and performance requirements for the Bank De Rose Project.

3. Comb through the selected Homework 5 submission to see if the usability requirements mandate a change in any of the prototype screens. For one such screen, describe how the requirements affect the screen? Describe the changes that you need to make and draw a rough sketch of the new screen. You must turn in the old screen.

CSSE 371 Software Requirements and Specification
Fall 2010
Homework 7

Due: Tuesday, October 12 2010

Purpose: Practice writing test cases. Please bring a print out to class.

Note: This is a team assignment.

What to do:

1. You can use any of your teams' 371 Assignment 4 submissions to complete this assignment.

2. Sketch the various test cases for the Withdraw Money and Transfer Money Use cases using the format described in the book (See Chapter 26 of The Requirements Text).

CSSE 371 Software Requirements and Specification
Fall 2009
Homework 8

Due: Tuesday, October 27 2009 (In Class)

Purpose: Practice specifying usability goals and user experience goals

What to do:

1. Assignment from Chapter 1(Page 37) of Interaction Design text.

CSSE 371 Software Requirements and Specification
Fall 2009
Homework 9

Due: Tuesday, November 3 2009 – In Class

Purpose: Practice designing, usability testing and re-design based on user feedback.

Note: This is a team assignment

What to do:

1. "The Butterfly Ballot: Anatomy of disaster" is an interesting account written by Bruce Tognazzini, that you can find by going to AskTog.com and looking through the 2001 column. Alternatively go directly to: http://www.asktog.com/columns/042ButterflyBallot.html
2. Read Tog's account and look at the picture of the ballot card.
3. Make a similar ballot card for a class election and ask 10 of your friends to vote using the card. After each person has voted ask whom he or she intended to vote for and whether the card was confusing. Note down their comments. Turn in your ballot card along with the comments.
4. Redesign the card and perform the same test with 10 different people. Turn in your redesigned ballot along with the comments.
5. Report your findings.

Project Milestone Rubrics

CSSE 371

Milestone 1 Grade:_____ Team:_____

Executive Summary: (5 Points)

Introduction: (5 Points)

Main Content (65 Points)

Client Background: (5 Points)

Current System: (8 Points)

Alternatives and Competition: (2 Points)

User Profile: (10 Points)

User Needs: (10 Points)

Product Perspective: (5 Points)

Feature Listing: (20 Points)

Constraints: (5 Points)

Writing: (15 Points)

Formatting:
Grammar:
Spelling:
Presentation:

Index and Glossary: (5 Points)

References: (5 Points)

Milestone 1 Specific Content: Here is a rough sketch of what the main content for Milestone 1 should resemble

a) Client Background – What the client does? It must describe what the client does, both in a broad overall sense and in day-to-day affairs, and describe how the project will fit into the day-to-day workings for the client. Use this section to provide a general overview; specifics can be given in the next section.
b) Current System – A brief description of the current system(if any) and identify its features
c) User/Stakeholder Description - Use this section to identify your users(other stakeholders) and how they will use the system. Must contain the following subsections
    1. User/Stakeholder Profiles (See page 439 of requirements text)
    2. User Environment(See page 439 of requirements text)
    3. Key Needs – List the key problems or needs as perceived by the user. For each user need, answer the following questions
        i. What is the need? (Use the problem statement template shown in Page 46 of requirements text)
        ii. How is it solved now?
        iii. What is a possible solution?
    4. Alternatives and Competition (See page 440 of requirements text)
d) Product Overview – Use it to provide a high level overview of your proposed system. It must contain the following subsections
    1. Product perspective (See page 440 of requirements text)
    2. Elevator Statement/pitch as discussed in class
    3. Summary of Capabilities(See page 441 of requirements text)
    4. Assumptions and Dependencies(See page 441 of requirements text)
    5. Rough Estimate of the Cost
e) Features
    1. Describe the attributes you have chosen and their meanings (See page 442 of requirements text)
    2. Provide a listing of features (See page 425 of requirements text)
f) Solution Constraints (See Page56 of requirements text)

CSSE 371

Milestone 2 Grade: _____ Team:_____

Executive Summary: (2 Points)

Introduction: (2 Points)

Content: (80 Points)

Project Background (10 Points)
(Need, Features, Client, current System…)

Use Case Identification (5 Points)

Use Case Layout (5 Points)

Use Case Description/Other Functional Requirements (30 Points)

Use Case Feature Mapping (10 Points)

Context Flow Diagrams (5 Points)

Level 0, 1, … Diagrams (15 Points)

Writing: (10 Points)

Formatting:
Grammar:
Spelling:
Presentation:

Index and Glossary: (2 Points)

References: (4 Points)

Milestone 2 Main Content: Here is a rough sketch of what the main content for Milestone 2 should look like

a) Include any relevant sections from Milestone 1 that might be relevant to understand this document
b) Use Cases: For each use case that you identify, follow the template discussed in page 450 of requirements text. See page 427 for an example.
c) Table to Map the use cases to the various listed features of the system.
d) Data flow Diagrams: Do context, Level 0, Level 1(as many levels as needed) to explain the flow of data for both the current and proposed systems.

CSSE 371

Milestone 3 Grade: _____ Team:_____

Executive Summary: (2 Points)

Introduction: (2 Points)

Content: (80 Points)

Project Background + Functionality Not in Use Cases (10 Points)

Usability Requirements (8 Points)

Performance Requirements (8 Points)

Reliability Requirements (8 Points)

Supportability Requirements (3 Points)

Hardware and Software Interfaces (3 Points)

Documentation, Installation, Legal and Licensing Requirements (5 Points)

Design Constraints (5 Points)

User Interfaces (30)

Writing: (10 Points)

Formatting:
Grammar:
Spelling:
Presentation:

Index and Glossary: (2 Points)

References: (4 Points)

CSSE 371

Milestone 4 Grade: _____ Team:_____

Executive Summary: (3 Points)

Introduction: (3 Points)

Project Background (10 Points)(Needs, Features etc… to add detail and context to the document)

Coding Standards (8 Points)

Change Control (15 Points)

How do you receive requests? What information do you expect?

How do you manage change? Decision Making etc

How do you handle changes to project artifacts? Source Control etc

Test cases: (45 Points)

Test Case Description (30 Points)

Completeness (10 Points)

Test case layout (5 Points)

Writing: (10 Points)

Formatting:
Grammar:
Spelling:
Presentation:

Index and Glossary: (2 Points)

References: (4 Points)

CSSE 371

Milestone 5 Grade: _____ Team:_____

Executive Summary: (3 Points)

Introduction: (2 Points)

Project Background (7 Points) (Needs, Features etc… to provide project background)

Usability Report: (45)

Process (15 Points)

Analysis (10 Points)

Findings (15 Points)

Presentation (5 Points)

Interaction Architecture (5 Points)

Initial Interface Design (8 Points)

Revised Interface Design (15 Points)

Writing: (10 Points)

Formatting:
Grammar:
Spelling:
Presentation:

Index and Glossary: (2 Points)

References: (3 Points)

Evaluations Letters

I have enclosed letters of evaluation written by Dr.Huzefa Kagdi and Dr. Ken Surendran.

Software Engineering Commons Dr. Huzefa Kagdi

A Summary of a Peer-Observation Exercise

Prepared by Huzefa Kagdi

1. Context

Subject/Observed: Dr. Sriram Mohan, Rose-Hulman Institute of Technology, Terre Haute, IN
Observer: Dr. Huzefa Kagdi, Missouri University of Science and Technology, Rolla, MO
Class: CSSE 333-Fundamentals of Database Systems or Introduction to Databases
Date: December 11, 2009
Purpose: Peer Observation exercise for Software Engineering Commons

2. Items to Observe

There was no one specific item that was the focus of observation. Sriram's preference was to have a fairly open observation session on his classroom performance. When asked about items that he would like to be observed, he responded "I would like a general observation focussing on my teaching abilities and ability to get the material across to the class.".

3. Observation Summary

Sriram was well prepared, organized, punctual, and delivered the lecture in a timely and interactive manner. He appeared very confident on the subject/topic of discussion (not my major area of expertise) and was very engaging with the students. The material was delivered in logical progression and at steady pace. There was a clear indication that students were actively learning; they were responding to, as well as asking, questions. He was respectful of the students and accommodating (e.g., he knew his students' names!). A combination of viewgraphs, white boards, and hands-on computer examples were used in a non-obtrusive manner (e.g., clear fonts/slide background, good handwriting, and screen size). After the conclusion of the class period, Siram and I had a good discussion on a couple of points that could be considered for potential future improvement.

Overall, I feel that Sriram is a dedicated and effective instructor, and I rate him high on a number of key teaching aspects (see below). I do not see that he needs to make any major shift from what he is doing.

4. In-Class Performance Rubric

4.1. Class Organization

Started class on time
Introduced lesson (overview or focusing activity)
Paced topics appropriately.
Sequenced topics logically.
Related lesson to previous or future lessons or assignments.
Summarized or reviewed major lesson points.
Ended class on time.

4.2. Delivery

Presented or explained content clearly.
Used good examples to clarify points.
Varied explanations to respond to student questions or needs for clarification.
Emphasized important points.
Used graphics or visual aids or other enhancements to support presentation.
Used appropriate voice volume and inflection
Presented information or led discussions with enthusiasm and interest.
Responded appropriately to student behaviors.

4.3. Student Interaction

Encouraged student questions.

Software Engineering Commons Dr. Huzefa Kagdi

Asked questions to monitor student understanding.
Waited sufficient time for students to answer questions.
Provided opportunities for students to interact together to discover/discuss or practice content points.
Showed enthusiasm for the content.
Showed respect for student questions and answers.

4.4. Content
Presented content at an appropriate level for the students.
Presented material relevant to the purpose of the course.
Demonstrated command of the subject matter.

5. Observation Notes During Class

“

- Sriram and I arrived about five minutes before the class. There were about 2 students then. The lab assignment and daily quiz were handed out.
- About 20 students (only 1 female).
- Used viewgraphs – Powerpoint slides.
- Started on time.
- Started with a summary from the previous class.
- Main topic – supquries – queries within queries - indirect information, and their related operations (e.g., IN, Exists), inner/outer joins
- A few students showed up late.
- Use of whiteboard – a good thing
- Use of whiteboard and hands-on stuff seemed a bit repetitive.
- Students were asked questions, and a couple of students also asked questions
- Student had access to slides (on their own machines).
- Example/output/test driven approach – my suggestion for later discussion.
- Given time from the lecture period to do quiz questions.
- Order of select and modification queries? – my suggestion for later discussion.
- Spent 1 hour and 5 minutes on the lecture portion.
- Rest of the period was for lab (due on Monday).
- A TA and the instructor were around to help with the lab.

“

Software Engineering

Peer Observation of: Dr Sriram Mohan, Rose-Hulman Institute of Technology

Class Observed: CSSE 376 – Software Quality Assurance on 03/18/2010 2:30 – 3:20
(by Ken Surendran, Southeast Missouri State University)

Background:
Sriram is teaching this course for the second time. On that day, the topic was *use of Mocks in unit testing*. This topic is an addition this time (perhaps as part of continuous quality improvement). The course is meant for Juniors in Software Engineering. These students had a course in Java earlier on – so they are familiar with the concept of interfaces, which is needed for this topic.

Setting the stage:
The class started promptly at 2:30. The students, as they walked in, picked up a quiz that had seven short pointed questions pertaining to the topic of discussion. The presentation slides for the class were posted ahead of the class; so were also the code examples discussed in class. Earlier, Sriram was keeping the students at ease as they were assembling – chatting on topics of general nature. This demonstrated the rapport Sriram had with his students. When the class started, he established the context in testing where mocks are used by providing a one-cell class diagram on the white board and referred to it during his discussions.

In depth Concepts:
Sriram first differentiated mocks from stubs and used simple examples to illustrate the use of mocks in interaction testing. He typed in the necessary codes for the interfaces as he was deliberating and showed how mocks are created manually. These examples were simple, drawn from commonly known domains, and complete – in the sense the students could see the test results.

Through these workings he showed how tedious it could be for creating mocks manually and the psychological implications leading to skip such testing altogether. At this point he introduced a range of tools available for dealing with mocks (frameworks handling implementations) and showed the class how to use one of them.

During this process he constantly checked with the students to ensure they are in sync with him! The students responded very positively and, at times, they made statements that exhibited understanding of the concepts. They were also appreciative of the quality – comprehensive nature - of the class sides. After explaining and illustrating a concept, Sriram drew the students' attention to the appropriate question they should be in a position to answer at that juncture. The students were highly engaged – learning each concept and applying it immediately by way of answering the questions in the quiz.

Final moments
Sriram reiterated that he intends to walkthrough the steps for using mocks in testing once again at the follow-on lab session the next day. By logically planning his sessions – three concept sessions followed by a lab session – the students get an opportunity to internalize the learned concepts. As the students walked out of the class, they turned in the quiz to Sriram. These quizzes are part of summative assessment for the course.

Style and analysis
Sriram seems to be using the *problem-based learning* approach in this course – which generally works well in such courses. By using a quiz-driven presentation, he is able to ensure the essential concepts are understood by the students. He has tailored his teaching to suit the audience. The current generation of traditional students is known for their multi-tasking. Sriram seems to have found a way of keeping these students engaged by letting them multi-task in class. The difference however is that all these tasks – listening to deliberations, following the slides, working through the example, answering the quiz questions - are channeled toward learning the concepts in the topic.

Sriram has excellent voice projection, varies his tone –everyone was attentive at this afternoon class, and uses the isle-space effectively to retain the attention of all students. Sriram is an enthusiastic teacher and has his focus on students' learning. I could sense his concern for the students' learning all the time. He used the class time very effectively – efficient use of time – to achieve his teaching objectives. All in all, it was a pleasure to observe Sriram's dynamic performance in the classroom.

Suggestion
The only suggestion I could make is that Sriram could expand a bit more on the class diagram he drew on the white board at the beginning. For the examples he used, he could have drawn complete class diagrams (with 3-cells) showing the interfaces (with the method names in them) and the classes that realize these methods. Even though it is not a design course; showing a class diagram for the code that follows may serve as a better abstraction – a picture that is easier to communicate and to remember the concept. There is very little that I could suggest to Sriram regarding his class preparation or delivery performance, since they are simply excellent. He just needs to maintain this momentum and build on them.