

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

REQUIREMENTS ENGINEERING PROCESS SIMULATION IN
UNDERGRADUATE SOFTWARE ENGINEERING COURSES

by

Stephanie Ludi

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

May 2003

UMI Number: 3084655

UMI[®]

UMI Microform 3084655

Copyright 2003 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

REQUIREMENTS ENGINEERING PROCESS SIMULATION IN
UNDERGRADUATE SOFTWARE ENGINEERING COURSES

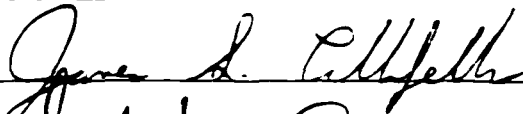

by

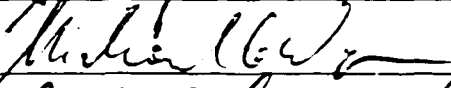
Stephanie Ludi

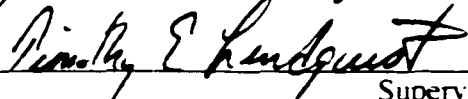
has been approved

July 2002

APPROVED:


_____.Chair






Supervisory Committee

ACCEPTED:



Department Chair



Dean, Graduate College

ABSTRACT

Software development is a complex process. Computer Science students progress through their degree programs and enter industry so that they may contribute to the software development endeavor. In industry, software engineers work on a variety of projects with varying complexity. While the types of projects vary, the size and complexity of the projects is greater than the small group projects developed in mandatory undergraduate Software Engineering courses. While the small group project, consisting of a team of 4 to 5 students, is intended to be a small version of a "real-world" project, discrepancies are evident that impact the real-world experience. A primary aspect of a small group project is that the instructor minimizes the likelihood that students' projects will fail at an early stage. During the requirements elicitation activity where students ask the instructor (the customer) questions, students expect that the instructor will provide correct information. Due to course restrictions, Requirements Analysis and Requirements Validation activities are conducted in a limited fashion. In the area of Change Management, students do not respond to the customer's feedback with the addition or revision of requirements in the system after delivery. Projects in industry do not provide such assurances.

In order to provide students with a more realistic opportunity to explore the software development lifecycle, this dissertation describes a simulator that provides undergraduate Software Engineering students with the Requirements Engineering knowledge and skills normally acquired while working on a large project. The simulator and the underlying model, based on System Dynamics Modeling, is intended for use in the undergraduate Software Engineering classroom. As the use of this research is to

enhance instruction. the hypothesis was tested whereby students, enrolled in an introductory course, using the simulator can increase their level of understanding, based on Bloom's Taxonomy, of the Requirements Analysis and Specification activities utilized in a long-term project.

ACKNOWLEDGEMENTS

Thank you Dr. Collofello, for your support in this research and my teaching in general. Too many professors do not consider educational research (or even teaching) to be worth doing, but you take the teaching responsibilities seriously and it is apparent in the classroom and out.

I also extend my thanks to my other committee members, as well as the other dedicated teachers and professors that I have had over all these years.

To my husband..

Who has been supportive of my educational endeavors over all these years.

TABLE OF CONTENTS

		Page
LIST OF TABLES.....		xi
LIST OF FIGURES.....		xii
CHAPTER		
1	INTRODUCTION.....	1
	Overview.....	1
	Statement of the Problem.....	2
	Proposed Solution to the Problem.....	3
	Research Contributions.....	6
	Why Requirements Engineering?.....	9
	Organization of Dissertation.....	9
2	BACKGROUND.....	11
	Bloom’s Taxonomy.....	11
	The Software Engineering Body of Knowledge Project..	14
	System Dynamics Modeling.....	19
	Educational and Training Simulations.....	21
	Summary.....	27
3	SIMULATION AND MODEL REQUIREMENTS.....	29
	Best Practices.....	30
	Course Content.....	32
	Topics Covered in the Simulation.....	35

CHAPTER		Page
	Summary.....	41
4	MODEL AND SIMULATION DEVELOPMENT AND METHODOLOGY.....	43
	The Simulation Structure.....	43
	Assumptions of the Simulation.....	45
	The Simulation Interface and Flow of Topics.....	48
	Initial Details.....	49
	Requirement Elicitation.....	50
	Facilitated meetings.....	50
	Interviews.....	52
	Requirement Analysis.....	54
	Requirement Validation.....	54
	Development Feedback.....	56
	Delivery.....	57
	Maintenance as Change Management.....	57
	Closure.....	58
	Methodology.....	58
	Scope of the Model.....	66
	The Development of the System Dynamics Model.....	66
	The Validation of the Simulator.....	70
	Summary.....	74
5	CASE STUDY.....	75

CHAPTER		Page
	Hypothesis Revisited.....	75
	Design of the Assessment.....	76
	Pilot Case Study Procedure.....	78
	Additional Pilot Case Study Design Considerations.....	80
	Pilot Case Study Analysis.....	81
	Requirements Elicitation.....	82
	Requirements Analysis.....	85
	Requirements Validation.....	88
	Requirements Management.....	90
	Case Study Procedure.....	98
	Additional Case Study Design Considerations.....	95
	Case Study Analysis.....	96
	Requirements Elicitation.....	97
	Requirements Analysis.....	100
	Requirements Validation.....	105
	Requirements Management.....	108
	Summary.....	110
6	CONCLUSIONS AND FUTURE WORK.....	111
	Conclusions.....	111
	Future Work.....	115
	REFERENCES.....	119

APPENDIX

A	SWEBOK KNOWLEDGE AREA MAPPINGS [8].	125
B	SWEBOK TOPICS ORGANIZED BY KNOWLEDGE AREA. AND CLASSIFIED ACCORDING TO BLOOM'S TAXONOMY.....	128
C	SIMULATOR SWEBOK TOPIC DEFINITIONS.....	151
D	SIMULATOR MODEL.....	156
E	SIMULATOR SCREENSHOTS.....	169
F	THE ASSESSMENT.....	177
G	ASSESSMENT TRACEABILITY MATRIX.....	188

LIST OF TABLES

Table		Page
1	Characterization Matrix for Simulation.....	4
2	Bloom's Taxonomy.....	12
3	Knowledge Areas and Related Disciplines.....	17
4	Topics Covered in Introductory Software Engineering Course.....	33
5	Requirements Engineering Topics and Sub-topics from SWEBOK.....	35
6	Summary of Software Requirements Topics in Various Instructional Contexts.....	38
7	Summary of Software Requirements Topics in the Simulation and the Corresponding Mapping to Bloom's Taxonomy.....	39
8	Simulation Topics Mapping to Bloom's Taxonomy.....	76
9	Summary of results for the facilitated meeting technique.....	83
10	Summary of results for interview technique.....	83
11	Summary of results for requirements analysis.....	86
12	Summary of statistical significant results for requirements analysis.....	87
13	Summary of results for requirements validation.....	88
14	Summary of statistical significant results for requirements analysis.....	90
15	Summary of results for requirements management.....	91
16	Summary of results for the facilitated meeting question #1.....	97
17	Summary of results for interview question #2.....	98
18	Summary of results for the facilitated meeting question #23.....	99
19	Summary of results for interview question #12.....	100

Table		Page
20	Summary of results for requirements analysis question #3 – priority.....	101
21	Summary of results for requirements analysis question #4 – scope.....	101
22	Summary of results for requirements analysis question #5 – volatility...	101
23	Summary of results for requirements analysis question #6 – requirement type.....	102
24	Summary of results for requirements analysis question #7 – requirement type.....	102
25	Summary of results for requirements analysis question #13 – requirement type.....	102
26	Summary of results for requirements analysis question #14 – scope.....	103
27	Summary of results for requirements analysis question #15 – priority...	103
28	Summary of results for requirements analysis question #16 – volatility.	103
29	Summary of results for requirements analysis question #17 – scope.....	104
30	Summary of results for requirements analysis question #18 – priority..	104
31	Summary of results for requirements analysis question #19 – volatility.	104
32	Summary of results for requirements validation question #8.....	105
33	Summary of results for requirements validation question #9.....	106
34	Summary of results for requirements validation question #10.....	106
35	Summary of results for requirements validation question #20.....	106
36	Summary of results for requirements validation question #21.....	107
37	Summary of results for requirements validation question #22.....	107
38	Summary of results for requirements management question #11.....	108
39	Summary of results for requirements management question #24.....	109

Table	Page
40 Summary of results for requirements management question #25.....	109
41 Summary of results for requirements management question #26.....	109
B1 SWEBOK Topics Organized by Knowledge Area, and Classified According to Bloom's Taxonomy.....	129
C1 Simulator SWEBOK Topic Definitions.....	152
D1 Model Definitions.....	161
G1 Assessment Traceability Matrix.....	189

LIST OF FIGURES

Figure		Page
1	Timeline for SWEBOK.....	16
2	The Organization of a Knowledge Area Description.....	18
3	Example of SDM using schedule, documentation, and maintainability	20
4	Layers of the Simulation.....	29
5	Layers of the Simulation Revisited.....	44
6	Flow of Interaction.....	49
7	Sample Facilitated Meeting Status Information Screen.....	51
8	Sample Interview Status Information Screen.....	53
A1	SWEBOK Knowledge Area Mappings.....	126
D1	Facilitated Meeting and Requirements Analysis Segment of Model....	157
D2	Interview and Requirements Analysis Segment of Model.....	158
D3	Requirement Validation and Implementation Phase of the Model.....	159
D4	Overrun Calculation Segment of the Model.....	160
E1	Project Description.....	170
E2	Facilitated Meeting: Selecting the Stakeholder.....	170
E3	Facilitated Meeting Status Screen.....	171
E4	Interview: Selecting the Stakeholder.....	171
E5	Selecting Interview Questions.....	172
E6	Final Interview Status Screen for a Topic.....	172
E7	Requirements Analysis: Classification Based on Scope.....	173

Figure		Page
E8	Initial Validation Screen.....	173
E9	Inspecting Requirements.....	174
E10	Feedback from an Inspection.....	174
E11	Sample Developer Feedback During Product Development.....	175
E12	Customer Feedback after Delivery.....	175
E13	Sample Change Submission.....	176
E14	Sample Feedback After Change Submission Analysis.....	176

Chapter 1. Introduction

1.1 Overview

The purpose of the undergraduate Software Engineering course is to provide undergraduate Computer Science and Computer Systems Engineering students (at Arizona State University) with the skills and knowledge needed to enable them to successfully participate in the software development process in industry. While the techniques used in industry may not be identical to those used in class, the overall lifecycle and activities are essentially the same. In class students are provided the opportunity to apply their newly acquired knowledge and skills in a small team project that lasts the majority of the semester. The project encompasses the Requirements phase through the Testing phase using the Waterfall process model. While the opportunity is provided, gaps are present that do not allow students to apply a variety of techniques in the same activity or to experience the interaction with customers in a more realistic manner than is currently accomplished via the instructor (and customer).

The objective of this research is to define and evaluate a model to provide undergraduate Software Engineering students with the knowledge and skills normally acquired while working on a large project. The model is based on System Dynamics Modeling (Abdel-Hamid & Madnick, 1991) and a large project simulator, intended for use in the undergraduate Software Engineering classroom. The use of the research to enhance instruction calls for an additional hypothesis, where the student who uses the simulator can better understand the Requirements Analysis and Specification activities in a long-term project.

1.2 Statement of the Problem

Software development is a complex process. Computer Science students progress through their degree programs and enter industry so that they may contribute to the software development endeavor. In industry, software engineers work on a variety of projects with varying complexity. While the types of projects varies, the size and complexity of the projects is greater than the small group projects that students develop while enrolled in the required undergraduate Software Engineering course at Arizona State University (and at many other universities). While the small group project consists of a team of 4 to 5 students is intended to be a small version of a "real world" project, discrepancies are evident that impact the real world experience.

The primary aspect of a small group project is that the instructor minimizes the likelihood that students' projects will fail at an early stage. During the requirements elicitation activity where students ask the instructor (the customer) questions, students expect that the instructor will provide correct information. In industry, customers may not always know the answers to developers' questions or may provide vague or incomplete answers unintentionally. In order to compensate for the compressed nature of a semester course and for the fact that students are learning Software Engineering concepts, instructors generally offer more complete answers quickly. Instructors want their students to develop a timely project and will not offer them such real world experience as to intentionally prolong the requirements analysis process. Students also receive projects that the instructor believes can be completed by the students in the course timeframe. Resource allocation and scheduling in industry are not so certain. Instructors

are biased towards the students' instructional needs and rightly so, but this does affect the impact of the project mirroring industry.

Other issues that affect the impact of a small group project include the process model used, the size of the development team, inconsistency of individual experience, the missing Maintenance phase, and time. While a variety of process models are presented in lecture, the primary model that is followed during the project is the Waterfall model. Students do not have the ability to experience other process models that are used in industry due to time constraints. Since the small group project is a miniaturized version of a project, devised for a team of about 5 students, students do not experience a large project in a large team. The management issues and communication overhead associated with larger teams is lost to students in small group projects. In addition larger teams can have a variety of structure and roles that members have whereas most student teams contain members with similar experience. Students also lose the opportunity to work with a domain expert or specialist while the instructor presents the foundation material, and students in the team do some research. Time constraints and limited access to resources does not allow for domain experts to directly work with student teams for the duration of the project or even during the Requirements phase.

1.3 Proposed Solution to the Problem

In order to address the shortcomings of the use of a small group project and lecture, a large project simulator is proposed that addresses some key areas that are currently absent from the Software Engineering curriculum. Issues including areas as requirements management for a large project, requirements analysis, and the roles that

stakeholders play in development are not presented adequately so that the students may appreciate the tasks and activities required to produce a set of requirements that lay the foundation for the project. By not allowing students to succeed or fail in a more realistic manner, students may not learn the importance of the concepts and their relationship to the development process to the extent that instructors could provide for. The scope of the project is shown in Table 1 using the Simulation Characterization Grid (Kellnet, Madachy & Raffo, 1999). The purpose of the grid is to enable researchers to present the scope and objectives of their simulation among the various aspects of development and perspectives of development that exist.

Table 1
 Characterization Matrix for Simulation

Purpose \ Scope	Portion of Lifecycle	Development Project	Multiple Concurrent Projects	Long-term Product Evolution	Long-term Organization
Strategic Management					
Planning					
Control and Operational Management					
Process Improvement and Technology Adoption					
Understanding					
Training and Learning		Future Work			Future Work

The scope of this simulation is shaded in black in Table 1, while the areas for future work are shaded gray. The simulation of the various aspects of a large project can

enable students to have a higher level of understanding of the tasks themselves, the roles of the tasks in the project as a whole (including over multiple increments), and the need for the tasks in the Requirements phase.

In order to address the need to enhance students' exposure to the best practices in a large software project, this research proposes a model and a simulation that can be utilized in the undergraduate Software Engineering classroom. Specifically, the main objective of this research is to:

Define and evaluate a model to provide undergraduate Software Engineering students with the knowledge and skills normally acquired while working on a large project.

This research has resulted in a simulation based on the System Dynamics Model that presents the consequences and impact of the choices that students make in a variety of processes and tasks during the Requirements phase. While the Requirements phase is emphasized, the entire development lifecycle, including maintenance, is represented to some extent. The model is modular – allowing the addition of more concepts in future and to allow customization by the course instructor. The model was transformed into a large project simulation for students to use while enrolled in an Introduction to Software Engineering or equivalent course. Students can use the simulation to experiment and learn from mistakes. During the course of the simulation and in the report at the end of the project, students can see how their decisions affect the project's quality, schedule, and cost.

As the model, and subsequent simulator, is intended as part of the instruction process an additional hypothesis is presented:

H1: Undergraduate students using the simulator will increase their level of understanding, based on Bloom's Taxonomy, of the Requirements Analysis and Specification activities utilized in a long-term project to a greater extent than with a traditional course without a simulator.

The hypothesis serves as the ultimate test of the overall objective. The results of the hypothesis assess the impact of the lessons learned by the students. Given that the students can better understand Requirements Analysis and Specification activities more effectively than if the simulator had not been used, they can carry the lessons learned on to their subsequent projects. The assessment method is outlined in Chapter 5.

The expected results are for students to have a significantly greater understanding of the importance of requirements analysis and specification activities when the simulator is integrated into a course with a small group project.

1.4 Research Contributions

This research is intended to make an addition to research in three distinct ways. The modeling of the Requirements phase of a project using System Dynamics Modeling enables the modeling technique to be applied to a new area of development. The model can then be used to produce a simulator that covers several aspects of development. Second, the project simulator itself is a unique product to be used in the classroom or in training environments. Utilizing the model as the foundation, the simulation is highly interactive in order to engage the students with the various activities in the context of a large project. Third, in order to test the effectiveness of the model and simulator, the test

of the hypothesis is also a significant contribution to Software Engineering education. Besides instruction in the best practices in Software Engineering, specifically Requirements Engineering, enables students to better understand the concepts, their purpose, and reasoning behind it. As such, the importance of using an effective large project simulator to measure this may motivate researchers to find new ways to simulate the development process for the classroom.

The immediate use of the simulator is for use as a supplemental aid to lecture material. While the importance of conducting requirements analysis and specification activities is discussed in class and students are required to traverse through a variety of activities during the course of their small group project, the simulator provides an opportunity that can be more realistic than the project. The simulator reinforces the role that the activities play in a large project in industry in ways the lecture and a small group project cannot present. Advantages to using a simulator include the following:

- Consequences of student decisions can include failure.
- Different process models can be experimented with, which time could never allow otherwise.
- Students need to ask questions and respond to customer feedback in a formal setting in several iterations, which cannot be fully explored in class.
- A larger project can be used with a larger team than the student team could consist of.
- Feedback can be displayed on a continuous basis, allowing the student to adjust his or her strategy during the various activities.

- The maintenance phase can be portrayed, including the possibility for multiple increments. The course could not accommodate this phase.
- The simulator can offer some consistency in the development experience that the group project cannot offer, while introducing enough variety to offer differing experiences to a large group of students.
- A wider variety of domains, increased scope and complexity can be portrayed than a course project could accommodate.
- Risks can come true in the simulator, which may not happen during the course of a course project.
- A simulated customer could mislead the student or offer a vague response, whereas the course instructor would not mislead the student. The student trusts that the course instructor would not provide incorrect information.

As such, students can better understand and appreciate the need for Requirements Engineering activities as they proceed to industry.

Beyond the uniqueness of simulating the Requirements phase, the simulator does not merely lead the student through the process, but instead is a means through which the student can be a part of the whole process. By participating in the project the students needs to make decisions and to live with the decisions made. Through this experience, the students can make the lessons learned by their success and failure (the consequences of their actions) meaningful when compared to merely reading about the best practices or escaping from situations not encountered in the team project.

1.5 Why Requirements Engineering?

The Requirements phase has been selected by the author since it is often overlooked in Software Engineering research though it is critical to a project's success. In regards to Software Engineering education specifically, the author feels that students do not appreciate the requirement elicitation activities (among others) enough to take it seriously during the course project. The instructor guides the students along, especially since the students are new to the project as a whole, and so the understanding of the requirements engineering phase is inadequate for the majority of students. The author feels that by emphasizing requirements engineering in the simulation, the students are more likely to notice the importance of the activities and best practices than what happens in the normal course of the project (and the course). The context of the large, complex, project, where quality, schedule, and cost are ongoing concerns, the student can get more realistic consequences to their decisions. Since such a project requires interaction with varied stakeholders, the student better understands the importance of working with stakeholders besides just the instructor as oracle.

1.6 Organization of Dissertation

Chapter 2 presents the background information regarding the Bloom's Taxonomy, the Software Engineering Body of Knowledge, systems dynamics modeling, related work, and the use of simulation in the Computer Science classroom. These topics provide the foundation for the development of the instructional goals, selection of the

topics, the selection of the simulation approach, and the application of the research respectively. Chapter 3 presents the requirements of the simulation and underlying system dynamics model. Chapter 4 describes the structure of the simulator and the methodology used to develop the system dynamics model and the simulation interface software. Chapter 5 describes the case study, the evaluation of the simulation in the undergraduate Software Engineering course. Chapter 6 presents the conclusions and future work that result from the research.

Chapter 2. Background

Introduction

The objective of this chapter is to provide background information regarding the technical and educational material related to the research, specifically Bloom's Taxonomy, the Software Engineering Body of Knowledge, systems dynamics modeling, and the use of simulation in the Software Engineering. Section 2.1 presents Bloom's Taxonomy. Section 2.2 discusses the Software Engineering Body of Knowledge. Section 2.3 provides the basics of System Dynamics Modeling and the rationale for its application to the research. Section 2.4 presents the use of simulation in the Software Engineering classroom and in industry. Section 2.5 provides a summary of these topics as they relate to the research.

2.1 Bloom's Taxonomy

During the mid-twentieth century, Benjamin Bloom led a movement to develop a classification of educational objectives. Three domains were identified, the Cognitive Domain, the Affective Domain, and the Psychomotor Domain. The Cognitive Domain, dealing with the development of knowledge, intellectual attitudes and skills, is represented as a hierarchy of educational objectives known as Bloom's Taxonomy (Carmeson, Delpierre & Masters, 2001). The taxonomy partitions various objectives into a spectrum representing simple skills to complex knowledge. The levels of the taxonomy are shown in Table 2, where Knowledge is the lowest level and Evaluation is the highest level of skill competence.

Table 2

Bloom's Taxonomy (Counseling Services of the University of Victoria, n.d.)

Level of Competence	Skills Demonstrated
Knowledge	<p>Observation and recall of information.</p> <p>Knowledge of dates, events, places, major ideas.</p> <p>Mastery of subject matter</p>
Comprehension	<p>Understanding of information.</p> <p>Grasp meaning.</p> <p>Translate knowledge into new context.</p> <p>Interpret facts.</p> <p>Compare and contrast facts.</p> <p>Order, group, and infer causes.</p> <p>Predict consequences</p>
Application	<p>Use information.</p> <p>Use methods, concepts, theories in new situations.</p> <p>Solve problems using required skills or knowledge</p>
Analysis	<p>Seeing patterns.</p> <p>Organization of parts.</p> <p>Recognition of hidden meanings.</p> <p>Identification of components</p>
Synthesis	<p>Use old ideas to create new ones.</p> <p>Generalize from given facts.</p> <p>Relate knowledge from several areas.</p> <p>Predict, draw conclusions</p>
Evaluation	<p>Compare and discriminate between ideas.</p> <p>Assess value of theories and presentations.</p>

Make choices based on reasoned argument.

Verify value of evidence.

Recognize subjectivity

Each level in the hierarchy has characteristics that present the extent of knowledge or skill that is expected upon successful completion of the lesson. The Knowledge level represents the recollection or identification of facts or other material. At this level, no deep analysis is required. The Comprehension level also represents a basic level of knowledge, but this level requires some understanding of the material where some estimation or interpretation is required. The third level of competence, the Application level, represents the use of the learned material (i.e. rules, concepts) in new contexts. At this level, the learner is required to do more with the information than just understand it, but to apply it. The Analysis level represents the ability of the learner to not just apply the learned material as a whole, but to break it down and understand its parts and structure. This level of understanding goes beyond application, as the system, its parts and the relationships between its parts are understood. Synthesis is the ability to put components together to form a new system. Such construction is not always concrete, but can be in various forms including that of communication. Rather than understanding the system that is presented, the learner can use his or her deeper understanding of the parts and relationships to develop a new system to apply the understanding to a new situation or context. The highest level of competence is Evaluation, where the learner can assess the value of material for a given purpose. At this level, the skills developed in the previous levels are applied as well and are applied

consciously for a specific purpose. A summary of the skills demonstrated for each level in Bloom's Taxonomy is presented in Table 2.

Although work has continued in this area since its introduction, Bloom's Taxonomy is a popular means of developing instructional objectives and assessment devices to meet those objectives. As mentioned in Section 2.3, the SWEBOK uses Bloom's Taxonomy to represent the level of knowledge and skill expected in the Knowledge Areas. In addition, the educational goals for the different topics addressed in the model and simulation utilize the labels presented here.

2.2 The Software Engineering Body of Knowledge Project

The Software Engineering Body of Knowledge Project (SWEBOK) is an effort by the IEEE Computer Society to develop a guide to the subset of generally accepted knowledge that defines the Software Engineering profession (IEEE Computer Society, n.d.). The aim of the project is not to define the body of knowledge itself or to dictate the curricula for university programs. However such a guide can assist in the development of curricula, accreditation criteria, and in the licensing of software engineers. The goals of the Guide to the Software Engineering Body of Knowledge are to:

- Characterize the contents of the Software Engineering Body of Knowledge;
- Provide topical access to the Software Engineering Body of Knowledge;
- Promote a consistent view of software engineering worldwide;

- Clarify the place of and set the boundary of software engineering with respect to other disciplines such as Computer Science, Project Management, Computer Engineering, and Mathematics;
- Provide a foundation for curriculum development and individual certification and licensing material. (IEEE Computer Society, n.d.)

The project consists of three phases: Strawman, Stoneman, and Ironman. The overall timeline for the entire project is in Figure 1. The Strawman phase is completed and resulted in a guide presenting the Knowledge Areas and Related Disciplines. The purpose of this Strawman phase was also to bring together the discipline in order to move the project forward. The Stoneman phase is near completion. The Stoneman version of the guide organizes the body of knowledge into Knowledge Areas, a list of topics relevant to the materials for each Knowledge Area and a list of Related Disciplines, as shown in Table 3 (Bourque & Dupuis, 1999). The ten currently identified Knowledge Areas, and the topics that comprise them, are regarded as core knowledge. The knowledge that software engineers need to know from related disciplines is not specified in the Guide, but is left to the other working groups. The Ironman phase has enabled experimentation and trial usage of the guide, promotion of the guide, and development of "performance norms" for professionals (Abran & Moore, 2000). The effort required to carry out the project consists of individuals from industry, academia, and standard setting bodies from all over the world.

A Three-Phase Approach for Developing the Guide to the SWEBOK

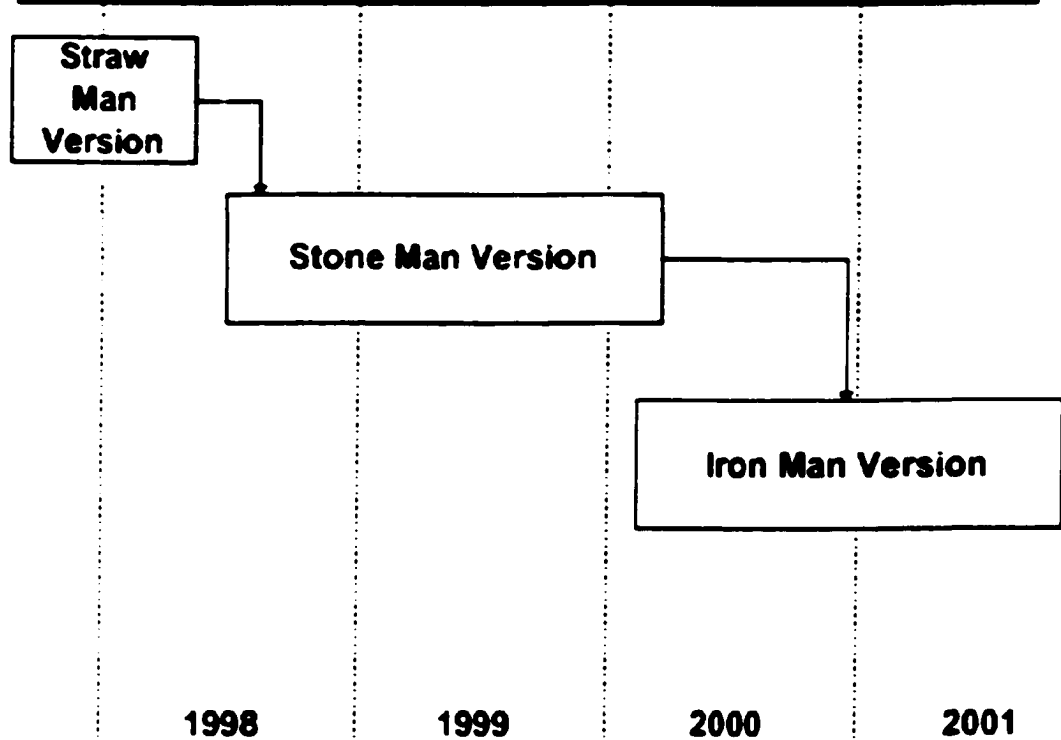


Figure 1. Timeline for SWEBOK (Dupuis, Bourque, Abran, Moore, & Tripp, 1999)

Table 3

Knowledge Areas and Related Disciplines

Knowledge Areas	Software Configuration Management
	Software Construction
	Software Design
	Software Engineering Infrastructure
	Software Engineering Management
	Software Engineering Process
	Software Evaluation and Maintenance
	Software Quality Analysis
	Software Requirements Analysis
	Software Testing
Related Disciplines	Cognitive sciences and human factors
	Computer engineering
	Computer science
	Management and management science
	Mathematics
	Project Management
	Systems engineering

The keys to the Guide are the Knowledge Areas and the mapping of topics within them. Each Knowledge Area is organized according to Figure 2, where it consists of a hierarchical breakdown of topics, reference topics, a matrix of the topics and the

reference materials. The topics for each Knowledge Area are decomposed and described, classified according to Vincenti's taxonomy, rated by Bloom's taxonomy, and referenced to related disciplines (Bourque & Dupuis, 1999).

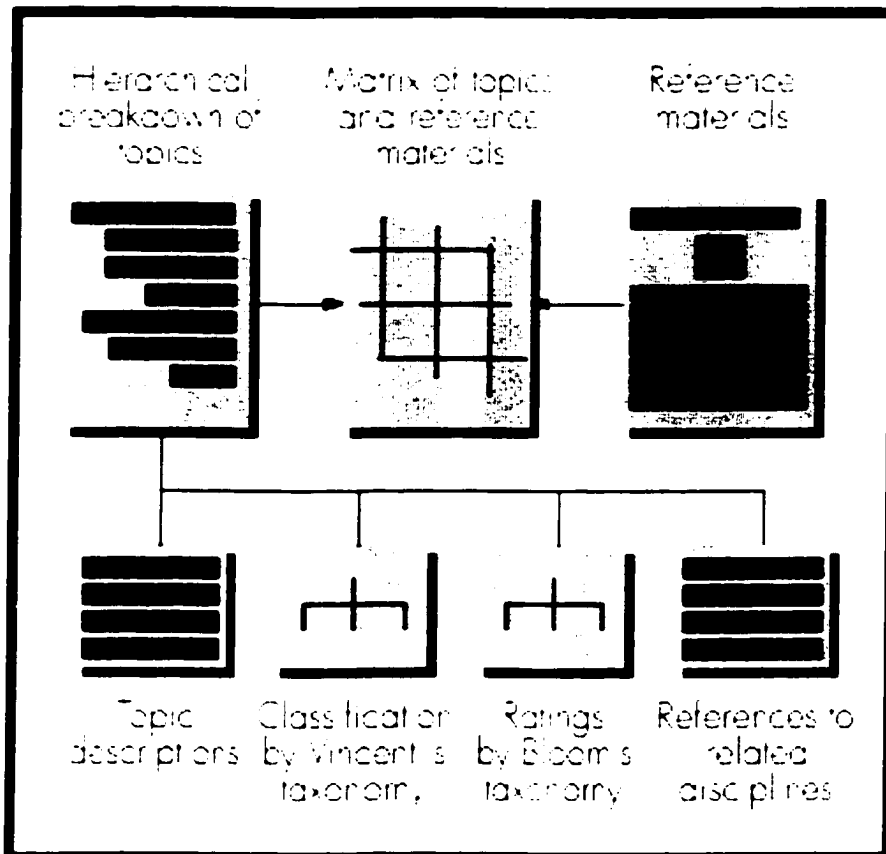


Figure 2 The Organization of a Knowledge Area Description (Bourque & Dupuis, 1999)

As of the current version of the Stoneman Guide (version 0.7), the mapping of topics to the Knowledge Areas is complete and shown in Appendix A.

While the purpose of the guide is not to dictate curricula, the guide does provide the topics and depth of knowledge for these topics based on Bloom's taxonomy for a graduate with four years of experience (Abran & Moore, 2000). The topics, organized by Knowledge Area, and the classification according to Bloom's taxonomy can be found in

Appendix B. This information can provide a base whereby a curriculum can be designed for an undergraduate software engineering program and undergraduate software engineering courses for computer science majors can be reevaluated. A required course such as Arizona State University's Computer Science Department's CSE 360 Introduction to Software Engineering course is an example of a course whose topics can be compared to the topics in SWEBOK's Knowledge Areas. This project takes the Guide to SWEBOK into consideration so that the course can best take advantage of the Guide and provides a more useful experience for students using the simulator as part of instruction.

2.3 System Dynamics Modeling

System Dynamics Modeling (SDM) was developed at the Massachusetts Institute of Technology in the late 1950's as a means to model the dynamic behavior of a system through the presentation of the cause-effect relationships and feedback loops that are observed in the system (Abdel-Hamid & Madnick, 1991). While SDM is a technique that has been applied to software, it is not uniquely applied to software. The model can portray the multiple layers of cause-effect relationships that exist in real systems, resulting in feedback loops where entities can observe the consequences to the affects that they cause. Systems dynamics models can present the people, processes, and products in the organization, enabling its application in a variety of systems.

The simplicity of the cause and effect relationships that exist in software development is evident at the macro and micro levels. A simple example of the relationships and feedback is evident in scheduling, documentation level, and system maintainability (see Figure 3). When a developer believes that he/she has fallen behind

schedule, he she revises his her strategy by decreasing the level of documentation.

This perception and revision in strategy is a cause and effect relationship, where the perception of falling behind is the cause and the decision to decrease the level of documentation is the effect. The decrease in the documentation detail and substance negatively impacts the maintainability of the system; this is another cause and effect relationship. Due to the short-term savings in time spent on the task of documenting by the developer, he she believes that their work is back on schedule. This shortsighted strategy is a side effect and is also a feedback loop.

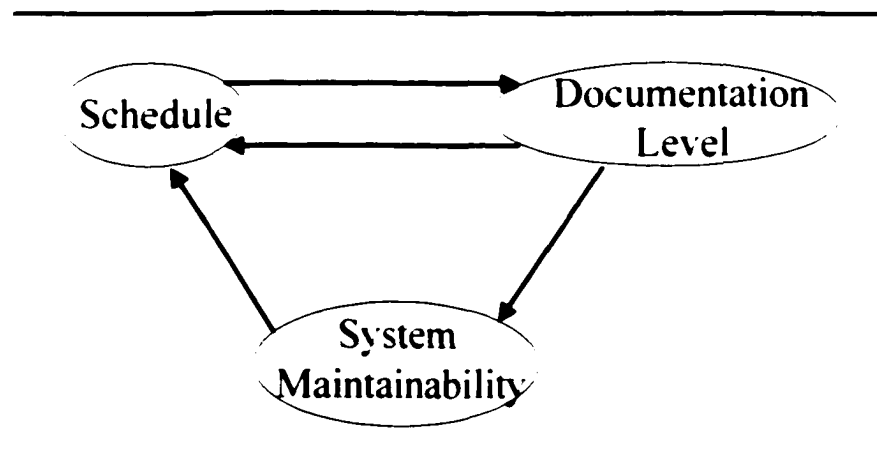


Figure 3 Example of SDM using schedule, documentation, and maintainability

In this example, the developer observed a schedule problem, took the action of decreasing the level of documentation, and observed that the project was back on schedule. In addition, another result emerges (system maintainability) that impacts the schedule and call for further action by the developer.

These cause and effect relationships and feedback loops enable developers to observe the consequences of their actions. Sometimes the consequences are immediate while others surface over a period of time. Even in this simple example, several cause-effect relationships work together to model the short and long-term repercussions of a change in documentation strategy.

Historically system dynamics modeling has been utilized in modeling different aspects of software engineering. Abdel-Hamid modeled the overall software development process (except for requirements engineering) (Abdel-Hamid & Madnick, 1991). The model was specific to the process utilized at NASA's Goddard Space Flight Center. Since then others have taken the model and focused on specific areas with software development. Thus over time, the effectiveness of utilizing system dynamics modeling to represent various aspects of software engineering is reinforced.

2.4 Educational and Training Simulations

A simulation can be a useful instructional model, allowing students to portray roles, face realistic conditions and develop realistic solutions (Joyce, Wel & Showers, 1992). While the situations are similar to the real world, the elements are simplified and presented in a controlled format. A positive aspect of simulations is the capacity for students to learn through the consequences of their actions that may not be able to occur in the real world. For example flight simulators and driving simulators allow pilots and drivers to experience their environments in order to learn and be tested for skill mastery, the alternative would be too expensive and dangerous.

Simulations can be useful in a variety of areas of study. Systems can be simulated and used to introduce a lesson, deepen students' understanding as a lesson progresses, or to provide a real world context for the culmination of a lesson. Simulations can be used at all levels of education, but in order for the success of the objectives to be verified the concepts and or skills that need to be mastered by the end of the simulation need to be identified. If specific roles are needed in order for the simulation to be conducted, they must also be clearly specified.

While the simulations appear to be self-contained, the instructor has a clearly defined role. The interactive environment of the simulation does not always produce clear-cut learning experiences for the student. In order to make the simulation more effective, the instructor needs to explain the fundamental rules, concepts, and issues raised in the simulation (Joyce, Weil & Showers, 1992). Simulations can take various forms, but for this project computer-based simulators will be discussed.

The field of Software Engineering has taken the concept of simulators and used them not merely as a cheap substitute for hardware, but instead as a means of providing opportunities to work with process in addition to the product. The use of simulators in Software Engineering assists both students and professionals. The Software Engineering Institute considers simulation to be a useful tool in various areas of development and training, but in the area of training specifically simulations can "play an important role" (Christie, 1999a). Much work has been done in the area of Project Management (Abdel-Hamid, 1993; Beagley, 1994; Chichakly, 1993; Smith, Nguyen & Vidale, 1993), where project leaders can use data to make educated decisions. Using System Dynamics Modeling, Collofello et al have been leading an effort to simulate various aspects of

Software Project Management (Collofello, 2000; Merrill & Collofello, 1997; Rus, Collofello & Lakey, 1998).

A Software Project Simulator was developed to provide hands-on experience that lecture-based instruction only presented (Merrill & Collofello, 1997). The simulator was developed in order to provide a learning environment to exercise skills where a simulator would be appropriate, specifically in the planning, tracking and control of a project and its development. Since the simulator was to supplement a course, lesson plans were developed in order to integrate the use of the simulator appropriately. The simulator was used by a graduate level project management course at Arizona State University. The students were surveyed in order to gather data on their management experience and understanding of project management concepts. Teams were formed and given the task of managing a simulated project. Data was provided to the students, including development process details, product details, personnel details, and historical metrics from past projects. The project had two increments, but the students were responsible for planning the second increment only. The teams were asked to plan the increment based on changes to the product requirements and were given guidelines in order to base their planning strategy. As a class exercise, some teams' plans were selected and run on the simulator so that the entire class could see how the different plans played out on the simulator in terms of discovered defects, productivity, and other attributes. Afterwards the results were discussed and compared to what the team's had expected. After the exercise, the students were surveyed to assess their learning and opinions. The students' learning was assessed in terms of how well the project was planned when compared to the default project plan data in the areas of cost, cycle time, and quality. All teams were

shown to have performed better than the default plan in at least one area. While the simulator is a work in progress, it has been shown to be a useful classroom tool for applying project management concepts.

Besides academia, simulators are also useful in industry to assist project managers in their training and in current projects. A collaborative project between Arizona State University and Motorola University was conceived to provide a Project Management training course that utilized a simulator (Collofello, 2000). As in the previously discussed simulator, System Dynamics Modeling was also the foundation for the simulator. The cause and effect relationships portrayed via the systems dynamics model, a variety of project attributes are presented using flight simulator instrumentation and data output displays. Project information is entered in to the simulator, including planned completion time, staffing, project complexity, and increment scheduling information. Once the simulator is seeded, output displays show the status of the project in terms of various attributes such as current staff load, elapsed person hours, and schedule pressure gauge. The student can pause the simulator in order to revise the input parameters. Once the simulator is resumed, the effects of the revised input data are observed. During the three-day training course students complete exercises in Life Cycle Model Comparison, Risk Management, Software Inspections, Critical Path Scheduling, and an exercise in overall project planning and tracking through project completion. The simulator was used in a class of 16 students at Motorola University. The students found that "the simulator added to the value of the course." (Collofello, 2000)

Another simulator can be used as part of a decision support system for specific attributes such as reliability. The simulator can assist project managers select the best

reliability engineering strategy for their projects (Rus, Collofello & Lakey, 1998). The simulator uses the System Dynamics Modeling and Discrete Event Modeling Paradigms as its foundation. The simulator has two parts, an expert system to suggest alternate scenarios and a simulator where the various scenarios can be evaluated and assessed.

The use of process modeling is used in several areas of professional training and also for use on the job. By simulating different scenarios, managers can make better decisions and improve their processes in such areas as resource allocation (Abdel-Hamid, Sengupta & Hardebeck, 1994), staffing (Sengupta, Abdel-Hamid & Bosley, 1999), and overall process improvement (Christie, 1999b; Robin, Johnson & Yourdon, 1994).

While much work has been done in using the dynamic modeling of process, the performance outcome does not always show improvement. Many managers continue to make poor decisions when they use the simulator (Sengupta & Abdel-Hamid, 1993; Sengupta, Abdel-Hamid & Bosley, 1999). Some issue has been taken with the type of feedback provided to the managers, specifically that the feedback centered on the outcome rather than cognitive feedback (Sengupta & Abdel-Hamid, 1993). With outcome feedback the manager does not have an adequate model of the system and does not see the relations among parts of the model. As such the managers cannot conceive of the shortcomings in their strategies and thus cannot improve the strategies. In other words the higher levels of comprehension in Bloom's Taxonomy are not being attained. Using cognitive feedback, the manager is provided with the task information and how the manager's cognitive system fits into the system. (Sengupta & Abdel-Hamid, 1993). The result is that the manager performed better than those provided with the outcome feedback. As these studies and projects are developed to support decision-making in a

professional environment, they do not directly relate to the classroom environment where students have less experience and resources. However the work done for the professional in terms of the importance for providing adequate feedback is useful in the design of the simulator.

Regarding the simulation of Requirements Engineering for any purpose, little work has been undertaken. A model has been developed that addresses the Joint Application Development (JAD) process utilized by the Computer Sciences Corporation for a specific project (Christie & Staley, 2000). This simulation was intended to show how both the organizational and social issues of requirements development affect the project in terms of quality and schedule. The model is an ongoing project to model social interaction in the context of development. The research was a proof-of-concept simulation based on a single project for a limited audience. The goal for the research was to see if the social of interaction could be modeled (using a real project as the base). The primary out of the simulation was the amount of time needed to complete the JAD sessions. While the premise behind the research is generally useful, it does not show the nuances of interaction that this research aims to show. Competence and other factors were simply numbers. In other words, the student would still be too separated from the activities involved in requirements engineering in such a simulation. This research endeavor is an instructional tool to enable the student to experiment with different techniques rather than just one as if he or she was actually participating in the project "in the trenches."

The use of simulation in requirements validation is also evident, however the simulation is not of the process but the simulation is of the actions of the system itself

(Christie, 1999a). Such simulation is intended for industry, especially for defense systems, rather than for training. This research focuses on the relationship between the requirements engineering process and the quality, cost, and schedule of the product. This simulation is intended for the classroom rather than industry.

Volatility is another area of simulation research (Pfahl & Lebsanft, 2000). The model was very specific in terms of scope. The purpose of the model was to present the impact of volatile requirements on a project's schedule and effort. Specifically the aim was to analyze the amount of money needs to be invested in order to stabilize the requirements in a cost-effective manner. The model was developed for a Siemens Business Unit using their data at the particular site. Like the previous model, it differs from this research in that it is more focused in scope and limited to a specific customer (one company).

After extensive research (including the past four years of work in the ACM Special Interest Group for Computer Science Education, Frontiers in Education, and the American Society for Engineering Education) no work has been found in the use of computer simulation for Requirements phase activities in the classroom.

2.5 Summary

The material presented in this chapter directs the focus of the research. The topics themselves are drawn from the Requirements Engineering key process area within SWEBOK. Bloom's Taxonomy is needed to ascertain the level of student understanding in the traditional course (using lecture, text, and project) and in the course with the simulator. Thus, the assessment questions can be developed at the appropriate levels of

complexity. The use of the simulator as a means of supplementing instruction is justified, and the application of the simulator to the undergraduate Software Engineering classroom is unique. In the other project management simulators discussed previously the student manager is often guided through scenarios and allowed to change a variety of parameters for use in forecasting that would not apply to a requirements situation. The result for the simulator is one that allows students to learn from the consequences of their decisions, in the context of Requirements tasks and activities, in order to judge the best decision based on several criteria and select the best course of action. The systems dynamics modeling technique is appropriate for the research as the technique has proven to be effective in modeling software development.

Chapter 3. Simulation and Model Requirements

Introduction

The objective of this chapter is to provide background information regarding the requirements for the simulation. The focus is to define the scope of the simulation system. Section 3.1 discusses the relevant best practices to be reflected in the system. Section 3.2 presents the course content that the simulator supplements. Section 3.3 discusses the topics covered in the simulation. Section 3.4 provides a summary of these topics.

The simulation is complex, consisting of several layers which represent the conceptual breakdown of the simulation. These layers are presented in Figure 4.

Simulation Interface	Interaction Layer
SDM Model	Model Layer
Simulation Assumptions	Assumptions
Simulation Topics	Topics

Figure 4 Layers of the Simulation

The simulation consists of an interface and a systems dynamics model. In order to better understand the contents of the model and the interface, the origin of the content needs to be explained. The content, which directs the purpose of the simulation, is guided by best practices, course content, and the relevant topics for the requirements engineering area. This chapter will discuss the bottom layer, the origin of the topics covered in the simulator. This layer serves as the foundation of the overall simulation.

3.1 Best Practices

While content is a significant feature of the simulator, a more general need for the simulator exists as well. In order for students to greater appreciate and support the value for the tasks involved in the Requirements phase, the simulator presents a large project in a manner allowing students to appreciate the lessons that are presented in lecture for an undergraduate Software Engineering course. These lessons are those pearls of wisdom that professors and experienced developers have shared over the years that are important enough to be mentioned during the course's lecture or reading. Such lessons that are often read about but cannot be appreciated by merely reading about them. Through the use of the simulator students have a deeper understanding. This understanding moves beyond the Knowledge level to the Comprehension level, the Application level, or the Analysis level, of these lessons:

- Spending time in gathering, analyzing, managing requirements is important for minimizing defects. (Basili & Boehm, 2001)
- Return on investment of time on requirement analysis and specification is significant. (Leffingwell, 1996)
- Following a change management process results in fewer changes in requirements being lost than if no process was followed. (Sommerville, 2001)
- Incompleteness and ambiguity in requirements is costly. (Gause & Weinberg, 1999)
- Requirements analysis and specification continues beyond the Requirements phase.(Sommerville, 2001)

- Dealing with the scope of the system early is important as software is complex. (Doll, 2001)
- Requirements need to be presented in different perspectives in order to minimize misinterpretation. (Sommerville, 2001)
- Non-functional requirements are just as important as functional requirements, and need to be presented in a precise manner. Non-functional requirements need to be held to the same standards as other requirements in terms of the need to be clear and complete (Sommerville, 2001).
- Understanding the existing application that is being replaced or extended is needed. (Ambler, 1999)
- Avoid scope feature creep by defining what will and will not be delivered. (Ambler, 1999)
- At some point you will need to restrict yourself to a realistic set of requirements that can be delivered. (Ambler, 1999)
- Invoke the real experts (domain and end users). (Ambler, 1999)
- Document the source of each requirement. (Ambler, 1999)
- The difficult part of requirements gathering is not the act of recording what the users want; it is the exploratory, developmental activity of helping users figure out what they want. (McConnell, 1998)
- Miscommunication can occur when the stakeholders and developers "speak different languages." Most end users and other stakeholders are not like the developers in terms of having business or technical backgrounds or other ways. Developers may need to change their mental model or perspective. (Leffingwell & Widrig, 2000)

While these lessons may not be the only ones that exist, they represent those that are represented in the simulator. Students are given the opportunity to appreciate these lessons as these “best practices” are demonstrated in a “real” project – a project that the students are a part of where real success and failure can happen and lessons can be learned.

These lessons are the umbrella for the topics and the project that the students traverse. The consequences of actions allow for students to appreciate the lessons more so than if they only read about them or heard about them in lecture.

3.2 Course Content

As the simulation is a supplement to the introductory Software Engineering course, the material covered in the course must be presented. The course is similar to many introductory Software Engineering courses that are required for Computer Science majors (and related programs) in terms of the topics covered. The list of topics, based on departmental objectives, reflecting the satisfaction of accreditation requirements, is listed in Table 4. These topics are listed in the general order at which they are presented in class since during the course of the semester some topics are rearranged in order to accommodate time or needs.

Table 4

Topics Covered in Introductory Software Engineering Course at Arizona State University

General Knowledge Area	Topic
Software Engineering Background	Overview of software life cycle models Overview of software development phases Software teams
Requirements Engineering	Importance of requirements Overview of requirements process Formal vs. Informal specifications Requirements elicitation techniques (e.g. meetings, interviewing, observation, use cases) Requirements documentation approaches (e.g. numbered paragraphs, data dictionaries, tables) Finite state machine specifications Non-functional requirements Attributes of a good requirements document Change Management Requirements reviews Human Computer Interaction

Design	<ul style="list-style-type: none"> Overview of the software design process Introduction to object-oriented design Design documentation Design quality measures and heuristics Design reviews
Testing	<ul style="list-style-type: none"> Overview of software verification and validation approaches Black box testing techniques White box testing techniques Integration testing System testing Regression testing
Project Management	<ul style="list-style-type: none"> Overview of software project management Cost estimation Project scheduling and tracking Configuration management Risk management
Metrics	
Overview of software maintenance	
CASE tools overview	
Professional Responsibility	

Each class meeting consists of lecture, with periodic collaborative, in-class activities to allow students to test their understanding of lecture material. During the course of the semester the students also work on the small team project in order to apply

some of the techniques discussed in class. The techniques used in the project are prescribed rather than determined by the students themselves. Experimentation and process improvement are not part of the "hands-on" experience due to time constraints in a course.

3.3 Topics Covered in the Simulation

The general scope of the simulator is the Requirements phase of project development. The goal of this research is not to produce a simulation for all aspects of a software project. Such a goal would not be realistic. The Software Engineering topics to be selected consist of a subset of areas that cannot be sufficiently covered in the Introduction to Software Engineering course but are included in the SWEBOK. The topics with the Software Requirements Knowledge Area in the SWEBOK are in Table 5. Using this Knowledge Area as a starting point, the topics covered in introductory Software Engineering course were analyzed.

Table 5

Requirements Engineering Topics and Sub-topics from SWEBOK

Requirements Engineering Topics	Subtopics
The requirement engineering process	Process models
	Process actors
	Process support and management
	Process quality and improvement
Requirements elicitation	Requirements sources

	Elicitation techniques
Requirements analysis	Requirements classification Conceptual modeling Architectural design and requirements allocation Requirements negotiation
Requirements specification	The requirements definition document The software requirements specification (SRS) Document structure and standards Document quality
Requirements validation	The conduct of requirements reviews Prototyping Model validation Acceptance tests
Requirements management	Change management Requirement attributes Requirements tracing

The basic curricula for the course, developed by the Undergraduate Curricula Committee at Arizona State University is more general than SWEBOK in terms of the topics covered. The curriculum does not use Bloom's Taxonomy in the presentation of topics. To address this shortcoming, a mapping of the course curricula to Bloom's Taxonomy and the SWEBOK has been drafted, and the mapping was subsequently verified by a professor who also teaches the course (James Collofello, personal

communication, 2001). The mapping was derived from the author's experience in teaching the course for three years (nine semesters), which followed the departmental curricula using Roger Pressman's *Software Engineering: a Practitioner's Approach*. The matrix containing the SWEBOK Requirements topics and the entire course curricula mapping using Bloom's Taxonomy are in Appendix B. The mapping of Bloom's Taxonomy to the SWEBOK topics and to the material currently covered in the Introduction to Software Engineering course is used to identify how the simulator can benefit instruction. Like the curricula mapping matrix in Appendix B, the determination of the most appropriate delivery method of material was first determined by the author, based on experience, including background knowledge in education. The author analyzed each software engineering topic while taking into consideration the level of Bloom's taxonomy that is the objective in the course. For topics where basic knowledge is needed in order to identify terms, lecture is suffice (Gleitman, 2000). When students need more concrete, hands-on experience to apply material the small group project is appropriate (Roland, 1997). Some topics are very complex, where different approaches need to be practiced and compared, the small class project is not capable of providing an appropriate instructional environment where the students can learn due to time constraints. If the concepts require the ongoing hands-on creation of a work product or a long-term (often repeated) process, then industry is most appropriate when learning to work in a large project since a course cannot accommodate the long-term investment of time and resources. The use of a simulator is appropriate when learning needs to be structured and conducted in a safe environment to allow for the building up of complex skills (Joyce, Weil & Showers, 1992). The simulator is feasible when techniques can be

applied or a process followed yet the need to interact with other individuals is not required and when the context of the work can be simulated (e.g. characteristics can be quantified, project material or feedback can be represented logically). At the conclusion of the classification, the matrix was verified by Dr. Collofello. Table 6 summarizes the topics that can be best accomplished via lecture, a small group project, industry experience, or a large project simulator, or.

Table 6

Summary of Software Requirements Topics in Various Instructional Contexts

Software Requirements Topic	Lecture only	Small Group Project (4-5 people)	Industry	Simulator
<i>Requirement Engineering Process</i>				
Process Models	x			
Process Actors	x			
Process Support	x			
Process Quality and Improvement	x			
<i>Requirement Elicitation</i>				
Requirement Sources	x			
Elicitation Techniques			x	x
<i>Requirement Analysis</i>				
Requirement Classification				x
Conceptual Modeling		x		x
Architectural Design and Requirement Allocation			x	
Requirement Negotiation			x	
<i>Requirements Specification</i>				
Requirement Definition Document				x
Software Requirement Specification			x	
Document Structure		x		
Document Quality				x

<i>Requirements Validation</i>	
The Conduct of Requirement Reviews	x x
Prototyping	x
Model Validation	x
Acceptance Tests	x
<i>Requirements Management</i>	
Change Management	x
Requirement Attributes	x
Requirements Tracing	x

While not all topics can best be conveyed through a simulator, a subset of topics was identified that can best be presented to students through the large project simulator. These topics are among those that the simulator consists of in order for the simulated project to have value in the course. The specific areas, the subtopics of those listed in Table 6, where the simulator can improve the course are noted in Table 7. These topics further limit the scope of the simulator for this research endeavor. These topics are defined further in Appendix C. In addition the Waterfall and Incremental process models are available to simulate the project.

Table 7

Summary of Software Requirements Topics in the Simulation and the Corresponding Mapping to Bloom's Taxonomy

<i>Software Requirements Analysis</i>	<i>Included in the Simulation</i>
I. Requirements Engineering Process	
A. Process models	
B. Process actors	
C. Process support	
D. Process quality and improvement	
II. Requirements Elicitation	
A. Requirements Sources	
B. Elicitation Techniques	
1. Interviews	x

2. Scenarios	
3. Facilitated Meetings	x
4. Observation	
III. Requirements Analysis	
A. Requirements classification	
1. Functional & Nonfunctional	x
2. Derived from 1+ high-level req. or imposed by a stakeholder other source	
3. Product or Process	
4. Prioritizing req. (mandatory, highly desirable, desirable, optional)	x
5. Scope	x
6. Volatility Stability	x
B. Conceptual modeling	
C. Architectural design & requirements allocation	
D. Requirements negotiation	
IV. Requirements Specification	
A. The requirements definition document	
1. For customer	
2. For other stakeholders	
B. The software requirements specification (SRS)	
C. Document Structure	
D. Document Quality	
1. Selecting appropriate indicators	
2. Gathering and Analyzing Metrics from reviews.	
V. Requirements Validation	
A. The conduct of requirements reviews	
1. Group composition is appropriate (may include customer)	x
2. Use of guiding documents like checklists to guide review and to doc findings	x
3. Review process is at specified checkpoints and redone as appropriate	x
B. Prototyping	
C. Model validation	
D. Acceptance tests	
VI. Requirements Management	
A. Change management	
1. Understanding the role of Change Management throughout lifecycle	x
2. Have procedure in place	x
3. Analyze proposed changes	x
B. Requirements activities	
C. Requirements tracing	

These topics allow students to apply the concepts learned in class to a simulated, large software project. In order to further place the research in context, the definition of a large project is needed. The definition of large project used in this research is a project that consists of a large software system. Such a software system contains more than 100,000 lines to code to millions of lines of code (Soukup, 1994). Also, a large system contains the scope and complexity that is beyond the capabilities of a single programmer. Instead, a large team (from dozens to hundreds of people) is needed to develop the system.

During the course of the simulation, the selected topics build upon one another as the simulation progresses in the context of a large project. Although the simulator concentrates on the Requirements phase, many problems are not apparent until later phases in the product's development. As a result, the subsequent phases of software development (Design, Implementation, Testing, and initial Maintenance activities) are presented as well. The flow of the topics will be discussed further in Chapter 4.

3.4 Summary

The material presented in this chapter presents the scope of the simulator itself (including the underlying model). The process of defining the specific concepts that the simulator embodies is presented in the rationale for the topics and lessons in relation to the traditional course. The topics themselves are a subset of topics from the Requirements Engineering key process area within SWEBOK. In addition, the assumptions for the simulator are presented in order to further scope the system and

provide context for the interaction. The next chapter provides further detail concerning the interaction and the model of the overall simulation experience.

Chapter 4. Model and Simulation Development and Methodology

Introduction

The objective of this chapter is to provide background information regarding the structure of the simulation interface software and the system dynamics model. The methodology used during the simulation development is also presented. Section 4.1 presents the overall structure of the simulation. Section 4.2 presents the assumptions associated with the simulation. Section 4.3 presents the structure of the simulation interface and the flow of topics. Section 4.4 outlines the methodology utilized in the simulator. Section 4.5 presents the scope of the system dynamics model. Section 4.6 discusses the development of the system dynamics model. The Section 4.7 presents the validation of the simulator.

4.1 The Simulation Structure

The project simulator presents students with a consistent project development experience in terms of the topics covered. Data relating to specific tasks is diversified in order to provide a more meaningful instructional exercise. The simulator presents each student with the tasks and topic information for the session. The student selects the factors that pertain to the topic at hand. The choices that the student selects are carried into subsequent topics so that the consequences of choices are seen and the students can react to the consequences. The system was developed with off-the-shelf products (Macromedia Director and HPS Ithink).

As presented in Chapter 3, the overall simulator consists of several layers (as shown in Figure 5). The Interaction Layer (the front end of the system) provides students with

an engaging interface to the simulator. Students are presented with options and make decisions that affect the project from the perspective of a Requirements engineer. This interface also guides the student through the sequence of events involved in the simulated project's development, from Requirements Elicitation through Maintenance (Change Management). At the end of the simulated project, the student is provided with a comprehensive report that details their choices, what the most appropriate choice is for the particular activity, and the impact that each choice has on the product and the impact of each choice on the project's quality, schedule and budget.

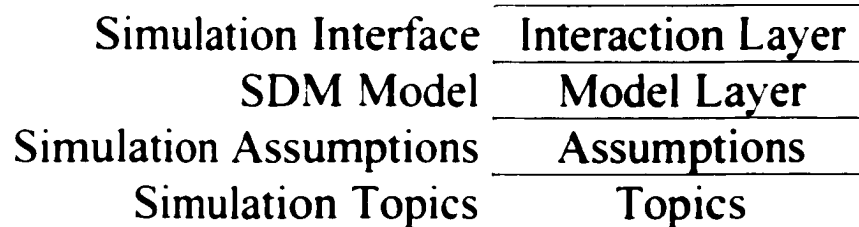


Figure 5 Layers of the Simulation Revisited

The advantage of this interface is that students can immerse themselves in the project and both the qualitative and quantitative details. Rather than merely entering values as parameters directly into a model and waiting for the output graphs (Beagley, 1994; Rubin, Johnson & Yourdan, 1994), students work in the context of the project and interact with stakeholders. Rather than entering the number of stakeholders to invite to a meeting, the student selects the two most appropriate stakeholders to invite to the meeting from a list. Rather than pressing the RUN button and watching the productivity graphs fluctuate on the screen, the student receives verbal feedback from the programmers based

on the quality of the requirements validation activity (in addition to the quantitative consequences on the project's schedule and cost). Further details regarding the specific flow of events and the types of choices and results are presented in Section 4.3.

In order to increase the replay value of the simulator, the requirements that are analyzed, inspected, and managed during the simulation are drawn from large lists that are organized in order to provide varied information, within the bounds of the quality of previous interaction. Most feedback from stakeholders and developers is drawn from a large pool of statements so that the quality is dependent upon previous interaction. In addition, the agenda topics used during elicitation activities are randomly selected in order to vary the presentation of information that the student needs to assess.

The underlying SDM model (e.g. the Model Layer or the back end of the system) receives inputs from the Interaction Layer, processes selected inputs from the student and simulates the impacts of the parameters on the project's schedule and cost. The results of the simulation model are then sent back to the simulator to display and utilize in subsequent activities. The ongoing output from the simulator is displayed alongside the quantitative or qualitative output from the Interaction Layer (the interface program).

4.2 Assumptions of the Simulation

In order to further define the scope of the simulator and to provide context for the student, several assumptions are made in the team, the project, and the process. Since these assumptions help set the stage for the simulation, the assumptions have been given their own layer in the overall simulation overview diagram in Figure 5. In the model, most of the assumptions are represented as constants. The assumptions are:

The Organization and Team.

- The members of the team developing the system generally have experience working on large projects using the technology needed to complete the project, but some inexperienced developers are present. The overall experience level is considered to be average.
- All employees work full-time.
- Morale and management support is high.
- Turnover and other staffing problems are not present.

The implications of the Organization and Team assumptions are that the underlying model will not be further complicated and need further modeling in order to accommodate various staffing, personnel, and administrative details. In order to test the hypothesis, the model is streamlined to concentrate on the Requirements Engineering (and subsequent) activities. Rather than dismiss the need to incorporate such details into the overall development model, the task is left for future versions of the simulator as described in Chapter 6.

The Project

- The project domain and details, the course registration and grade display system for the entire California State University system, are of sufficient complexity for a large project simulator.
- The given estimation of effort and cost are appropriate for the large project.

The hypothetical project is assumed to be large enough in terms of the size, complexity, and effort required. No one developer could develop and maintain such a large system. Such a large system gives the student the opportunity to participate in the type of large project that is undertaken in industry – offering added value to the simulator. Other systems in this domain that exist are student information systems or course registration systems for a university. Such projects are developed by several developers over the course of a couple of years, from initial planning to delivery. The hypothetical project scaled up the course registration system and added complexity so that it is easily classified as a large project.

The Development Process.

- In regard to communicating with stakeholders, the assumption is made that the customer is in the same geographical region as the developer. As a result, travel time is not an issue in terms of the customer or other stakeholders attending meetings or interviews.
- Neither elicitation technique is better than the other in terms of efficiently gathering requirements. Either technique is appropriate.
- Only one elicitation technique is used per project. In other words students do not use both the facilitated meeting technique and interviews in the same project (session).
- When the facilitated meeting technique is used to elicit requirements, all of the requirement engineers' questions need to be addressed by the appropriate stakeholders before the subsequent activities can proceed.

- During the requirement validation activity, all selected stakeholders are prepared for the inspection meetings. Even stakeholders who are not appropriate to include in the inspection process have looked over the requirements and have comments for the meetings.
- Once requirements are identified for rework, the necessary corrections are assumed to have been conducted.

The Development Process assumptions streamline the underlying model so that it will not be further complicated and need further modeling in order to accommodate various development-related details. In order to test the hypothesis, the model is streamlined to concentrate on a straightforward sequence of Requirements Engineering (and subsequent) activities. During each topic (e.g. Requirement Elicitation) the student concentrates on the current tasks rather than dividing his/her attention between the current topic and periphery topics (e.g. staffing). The simplified sequence of events allows the student to concentrate on the tasks at hand rather than be confused with details that will distract the student from the primary objectives of the system. Future versions of the simulator will incorporate such details into the overall development model as it evolves. See Chapter 6 for more information.

4.3 The Simulation Interface and Flow of Topics

The Interaction Layer of the simulator presents the sequence of activities that the student traverses. The flow of interaction is presented in Figure 6. Two general paths are possible in the simulator, with each path represented by the elicitation technique selected

in the onset of the project (either Facilitated Meetings or Interviews). The following presents the order and details of the activities involved in the simulated project.

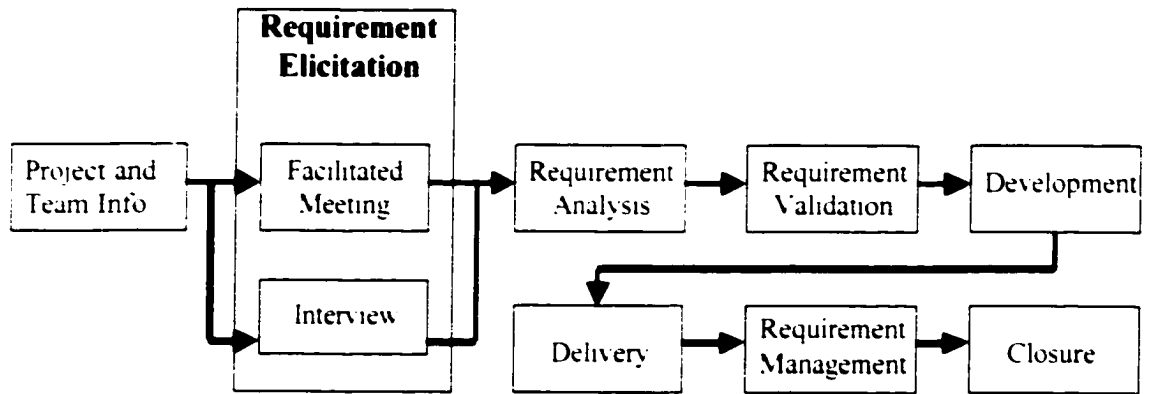


Figure 6. Flow of Interaction

4.3.1 Initial Details

Before the student can start the project, he or she must know what the project is about and what the development team is like. The author spent considerable time developing the requirements for a hypothetical, large project. The project, a course registration and student information system for all of the campuses in the California State University system. The project was selected since the student is familiar with course registration, grades, and general university life. Thus, the student would not need considerable time to learn about the domain. At the same time, the project is different from the course registration and student information system at Arizona State University, and so the student is compelled to read the details carefully. Since the primary task of the student is to gather and work with requirements, the initial project description does not list all of the project requirements. Instead the description provides an overview of the

project and lists some requirements. The description can be referred to at any point in the project.

In addition to the project description, a description of the team that the student participates in is presented. Overall team details are provided so that the student can feel more that he/she is part of a team rather than merely making selections with a mouse. Like the project description, the team description is viewable at any time in the project.

4.3.2 Requirement Elicitation

The first decision that the student needs to make is to choose which elicitation technique he or she wishes to use for the project. Either the Facilitated Meeting technique or the Interview technique can be selected. For the hypothetical project, either technique is appropriate. With the simulator, the student compares their strategies with the two techniques as he or she can experiment with each technique at each use of the simulator.

4.3.2.1 Facilitated meetings.

When the student selects the Facilitated Meeting technique, he or she needs to invite the most appropriate stakeholders to the meeting. The student is engaged in several meetings, one at a time, where each meeting has three agenda topics. The three agenda topics are randomly selected from a set of pre-defined topics. When the meeting agenda topics are displayed, the student selects the two stakeholders (from a list) that are most likely to answer all of the requirements engineers' questions regarding the project's requirements.

Once the student sets up the meeting, a meeting status screen displays a variety of information. The status screen informs the student of how well the meeting went in terms of what percentage of the developers questions were answered by each selected stakeholder and what percentage of all questions for all topics have been answered at the time. The quality of the answers, based on the average of the percentages of answered questions from all meetings conducted, are analyzed in the Requirement Analysis section. In addition two counters are displayed. These counters show the impact of the current meeting on the overall project schedule and cost, in terms of the number of days that the schedule will be exceeded and the number of dollars that the project budget will be exceeded. The layout of the status information is displayed in Figure 7.

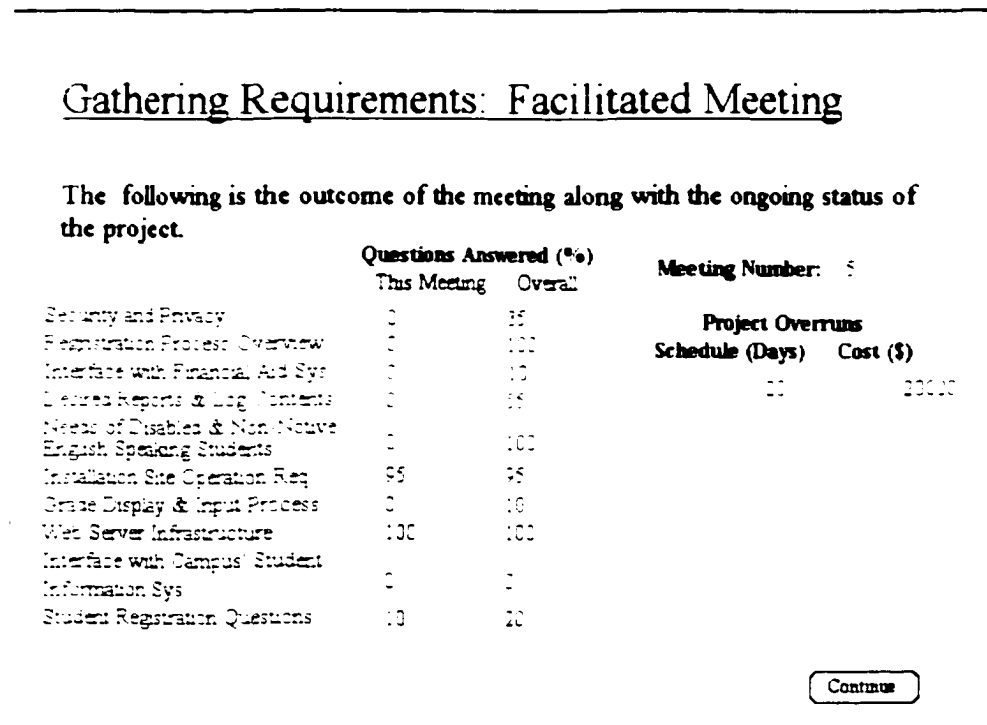


Figure 7. Sample Facilitated Meeting Status Information Screen.

After each meeting, the student decides whether to include the current topics in future meetings or to remove the topics from future meetings. If all questions are answered for a topic (100%), then it is automatically removed from future agenda meetings. Next, the student is presented with the agenda topics for the next meeting. The simulator asks the student if he or she wishes to keep the current stakeholders or choose new ones. The cycle continues until all of the topics have been addressed, and hopefully all of the questions for all of the topics have been addressed.

4.3.2.2 Interviews.

When the student selects the Interview technique, he or she needs to invite the most appropriate stakeholder for each interview. The student is engaged in several interviews, one at a time, where each interview has two agenda topics. The two interview topics are randomly selected from a set of pre-defined topics. When the interview agenda topics are displayed, the student selects the stakeholder (from a list) that is most likely to answer all of the requirements engineers' questions regarding the project's requirements.

Next, the student selects five questions from a randomly-generated list for each topic included in the interview. The selected questions should be the most clear and complete questions from the list that are appropriate for the stakeholder (questions that he/she can answer). Rather than quantitative results, the student sees the stakeholder's responses in a Questions-and-Answer format for each interview topic. As in the Facilitated Meeting technique, the quality of the answers are analyzed in the Requirement Analysis section. Using the Interview technique, the quality of answers is based on the average of the percentages of selected questions that are appropriate to ask the

stakeholder from all conducted interviews. An appropriate interview question (for the selected stakeholder) will yield a response that is useful to the requirement elicitation activity. The quality of the answers factor (combined for all topics) is used to select the quality of requirements that are examined in the Requirement Analysis activity. After the interview, the student selects whether to conduct another interview with another stakeholder (if the student is not satisfied with the results with the interview) or to conduct another interview with a new set of topics. If the student moves on to another set of topics, then the interview cycle continues until all of the topics are covered.

At the interview status screen, two counters are displayed that show the impact of the current interview on the overall project schedule and cost. The layout of the status information is displayed in Figure 8.

Gathering Requirements: Interview

Below is a sampling of the outcome of the interview. Read through each question and the stakeholder's answer. The responses will represent how knowledgeable the stakeholder is in the topic areas you selected.

Topic: Grade Display and Input Process (into the system).

Q: Is it possible to have a course without a grade assigned to it at the end of the term?

A: Yes, an incomplete will have an "I" instead of a grade. Also if the instructor does not get the grades in on time it will be blank.

Interview Number:	Project Overruns
:	Schedule (Days) Cost (\$)
:	:
:	:

Continue

Figure 8. Sample Interview Status Information Screen

4.3.3 Requirement Analysis

During the Requirements Analysis activity, students need to assess subsets of requirements based on four different perspectives: Scope, Type, Priority, and Volatility. For each type of analysis, seven requirements are randomly selected from a larger set. The student must analyze each requirement and categorize it. According to Scope a requirement can be in or out of the scope of the system. According to Type, a requirement is either functional or nonfunctional. According to Priority, a requirement is either Mandatory, Desirable, or Optional. According to Volatility, a requirement is either High, Medium, or Low probability of changing during the course of development.

The student does not receive immediate feedback from the simulator. Instead the Interaction layer averages the percentages of the requirements that were identified correctly for all of the perspectives. This result is sent to the model for use in the derivation of the number of days that the schedule is overrun and the number of dollars that the budget is exceeded. In addition, the result is also used to select the quality of requirements used during the subsequent Requirements Validation activity. The more accurate the analysis, the better (in terms of completeness, clarity, and accuracy) the requirements set to be inspected.

4.3.4 Requirement Validation

The quality of the requirements from the analysis activity affects the quality of the requirements used in the validation activity. At this stage, the selection of topics is narrowed to three topics. Three topics are chosen in order to allow students to

concentrate on the quality of inspection rather than quantity during this instructional activity. More than three topics would render the inspection process tedious in respect to the other project tasks in the simulator. Within each topic a large pool of requirements are available from which random selections are made. For nearly all requirements, different versions exist for good, fair, and poor quality requirements. Good requirements are clear and complete. Fair requirements have moderate problems with clarity or completeness. Poor quality requirements have serious issues with clarity and or completeness.

For each topic, the student is presented with a partially filled team needed to inspect the requirements. The student is given the opportunity to either select an appropriate stakeholder from a given list or not select a stakeholder at all if none is appropriate. Then, a list of randomly generated requirements for each topic is presented. The student inspects each requirement, based on a provided inspection checklist, and selects that each requirement is acceptable as-is, needs revision, or that the student is not qualified to inspect the requirement (the requirement is outside of his/her domain).

The inspection status screens display the results of the inspection. The feedback from the different members of the validation team are presented for the group of requirements. Besides the requirements themselves, the feedback also is presented from the teammates for the quality of your inspection. Based on the quality of the feedback, the student can either hold another inspection for that topic or move to the next topic until all topics have been addressed.

The average of the correctly identified requirements for the current inspection is sent to the model for use in the cost and schedule penalty calculations that are displayed

in the last status screen for each inspection. The average of all percentages for the correctly inspected requirements during all inspections is calculated by the Interaction Layer (the interface program). In addition, the result is used to select the quality of the developers' and customers' feedback during the subsequent development phases and delivery, respectively. The more accurate the validation, the better (in terms of completeness, clarity, and accuracy) the requirements set to be used during development. The better the requirements set, the better the product that is delivered to the customers.

4.3.5 Development Feedback

Based upon the quality of the requirements set from the validation activity, the student received feedback from designers, programmers, and testers from his or her team. Statements are displayed from designers at first, and the impact of the requirements set and their feedback (which describes the quality of the requirements and any rework that needs to be done) is presented in terms of any overall schedule and cost penalties. Next the programmers present their feedback along with any overall schedule and cost penalties. Lastly, the testers present their feedback, and any penalties are also displayed for the student. The statements originate from the Interaction layer, while the penalties are derived from the underlying model.

4.3.6 Delivery

After the student receives feedback from his or her simulated teammates, the product is delivered to the simulated customer. The simulated teammates enable the student to feel more a part of the team rather than merely a user of a program. In addition, the student learns to accept qualitative feedback from (simulated) peers rather than only quantitative feedback from the simulation. The delivery feedback consists of verbal feedback from the customer as to whether the product meets their needs (in terms of quality), and the display of the schedule and cost overruns for the project to that point in time. The feedback is drawn from possible statements stored in the system, while the penalties are drawn from the model. The quality of the requirements set produced during the validation activity (the percent identified correctly) drives the customer feedback.

4.3.7 Maintenance as Change Management

As a product's development does not end with its delivery, the simulator provides the means for the student to participate in the product's maintenance in terms of change management. The quality of the requirement set from the product affects the change submissions that are used in this phase. Since the product has been delivered, the schedule and cost counters have been reset.

Four change submissions are presented to the student, one at a time. For each change submission, information is provided concerning the nature of the change, who has asked for the change, the need for the change, and an estimation of the time needed to complete the change. The student analyzes the change, using a supplied list of heuristics, and determines the priority of the change within the scheme of the overall maintenance cycle. The student can chose to implement the change immediately (High Priority), wait

to implement the change until the next release (Medium Priority), or delay the change until a possible, yet undetermined time in the future (Low Priority). After the student prioritizes a change submission, the student receives verbal feedback from the developers on the project and the customer. The feedback refers to how the decision either positively or negatively has affected the project's quality, schedule, and cost. The short-term and long-term effects are shared with the student by those whom the decision affect. For example if the student chooses to have a change implemented immediately, when the change should have been delayed until the next release, the customer may thank the student but the developers complain about losing time that needed to be allocated to more pressing issues.

4.3.8 Closure

The last stage of the simulator is the presentation of the report. The student sees a detailed history of his or her decisions in the system. Besides the selections themselves, the report also displays any cost or schedule penalties for each decision. For the requirements elicitation activity and the requirement validation, the report also presents the stakeholders who were most appropriate for the topics covered during the elicitation session inspection.

4.4 Methodology

The design of the simulator is based on the research hypothesis, whereby the student will be able to better understand the Requirements Analysis and Specification activities in a long-term project. The simulator serves as an instructional tool that enables

the student to increase his/her level of understanding from one level to another. This learning is based on Constructivist Learning, the learning paradigm whereby the student builds knowledge through active learning and reflection (New York Institute of Technology, n.d.). Constructivist theory is based primarily on the work of Jean Piaget. The main principles of Constructivism, that support the use of the simulator as an instructional aid, are that:

- Constructivist knowing assumes the active and proactive nature of learning, and knowledge.
- Students use prior knowledge and experience as a starting point for useful, personal knowledge construction.
- Constructivist learning experiences include reflective thinking and productivity. (New York Institute of Technology, n.d.)

The simulator is not computer-based training, where the program serves an electronic tutorial and quiz. Instead, the student is actively involved in his/her learning. The student needs to use prior knowledge and utilize it within the context of a large project. During the project and after the project has ended (and the report is presented), the student reflects on his/her decisions and adjusts them as needed in order to produce a better product in the future (simulated or otherwise).

For both elicitation techniques, the goal is for the student to ascend to the Application level of understanding, based on Bloom's Taxonomy. At the Application level, the student uses previously acquired information in new and concrete situations to

solve problems (Krumme, 1995). The new situation is the meeting or interview needed to elicit requirements using either the facilitated meeting or interview technique, respectively. The knowledge of what the techniques consist of is the previously acquired information. In order to provide some structure, the topics are provided for the meeting or interview. Based on this information, the student selects the most appropriate stakeholders, those who will be able to satisfactorily answer questions, for the meeting or interview. A list of stakeholders is provided in order to keep the student focused on the list of stakeholders, and to facilitate the computer simulation. In the case of the Interview technique, the student also selects the five most appropriate questions from a list for each of three topics on the interview agenda. Allowing the student to select the appropriate questions, allows the student to show his/her understanding of what a clear and complete question is – a question that would be useful in obtaining an answer from the stakeholder. During the class project, the student always asks the instructor any project-related questions since the instructor represents all project stakeholders. The simulation allows the student to ask multiple stakeholders questions over time. The output provides the student with minimal information needed to assess the quality of the meeting or interview. The feedback is then used to direct the future actions, either to continue the quality of work or to counteract inferior decisions. For the Facilitated Meeting technique, the student is presented with quantitative information consisting of what percentage of all questions were answered by each stakeholder for the current meeting and ongoing meetings. Having verbal feedback from the stakeholders for the different topics would be overwhelming. However, knowing how effective the meeting was, in terms of the amount of questions answered, is the main issue. By contrast, each interview presents the

student with the answers to the questions that were selected. Such detail is more appropriate in the Interview technique since only one stakeholder is being interviewed, and the student deserves such feedback in order to ascertain the correctness of the choice of the questions selected. The crux of conducting interviews is to ask the right questions, and this activity carries that through to the student.

For both elicitation techniques, a pair of data points is provided to the student – the counters for the schedule and budget overruns. These counters are provided during the meeting and interview status screens, in addition to nearly all other topic status screens. The purpose of the counters is to present the status of the project in a succinct, meaningful manner. After all the goal of software engineering is to deliver a product that is 'on time and under budget.' These counters provide the student with this product information so that the student can see how his/her decision affected the project's schedule and budget. When the student sees that his/her choices are negatively affecting the schedule or budget, then he/she can use the opportunity to make different and better choices.

During the Requirement Analysis activity, the goal is for the student to ascend to the Comprehension level of understanding, based on Bloom's Taxonomy. At the Comprehension level, the student can understand the meaning of informational material, and can demonstrate the understanding through classification, description, and providing examples (Krumme, 1995). The simulation provides an environment where the student exercises his/her understanding of each type of requirement (e.g. by priority, scope, volatility, and functional/nonfunctional). In order to provide some structure, each type of analysis is conducted separately and the requirements are presented in a list. Within each

list, the student classified each requirement using criteria appropriate for each requirement type. The classification schemes are minimal in order to provide concrete gradations and avoid confusion. After the classification activity is completed the student progresses to the next activity, requirement validation. Unlike the other topics, no immediate feedback is provided that informs the student how many of his/her responses are correct. The immediate feedback is presented in the quality of the requirements that are inspected in the subsequent activity. However the student receives feedback in the final report, where each requirement is presented with the student's response and the correct classification for comparison. Since the intent for the simulation is to show the consequences of the correct and incorrect choices, the design decision was made to not allow the student to correct the classification. If the simulation merely informed the student whether the requirements were classified correctly (immediately after the classification occurred), then the student does not see the relationship between the choice and the consequences at the project level.

During the Requirement Validation activity, the general goal is for the student to ascend to the Application level of understanding, based on Bloom's Taxonomy. As in the case of the Requirement Elicitation activity, the student uses previously acquired information in new and concrete situations to solve problems (Krumme, 1995). The new situation is conducting the inspection. The knowledge of what an inspection consists of is the previously acquired information. Within the specifics of Requirement Validation, the additional goals of understanding the use of guiding documents and of the need to review requirements at various points in the development process at the Analysis level exist. Understanding at the Analysis level involves the usage of prior knowledge in the

recognition of patterns, the recognition of hidden meanings, and the organization of parts. The prior knowledge consists of the understanding of checklists and at the conduct of reviews throughout the lifecycle. In order to provide some structure, the topics are provided for the inspections, and most of the members of the inspection teams are provided. Based on this information, the student selects the appropriate stakeholder for each inspection team. It is possible that no other stakeholder is needed besides those already in the team. This possibility exists in order for the student to examine the structure of the team in regard to the requirements that need to be inspected. For each topic, a list of requirements is provided for inspection. A checklist is provided that the student uses during the inspection process. The inspection process allows the student to apply their understanding of the project requirements and of the inspection process. However, the Analysis level of understanding is apparent as the student breaks the inspection process down into steps and examines the structure of the requirements. The requirements are classified as either being valid, needing revision, or out of the student's domain. The possibility of a requirement being out of the student's domain of expertise, introduces the need for the student to distinguish his/her role in the process. The result of the inspection is presentation of the verbal feedback from the other reviewers. In addition the budget and schedule counters are presented, showing the impact of the inspection on the overall project budget and schedule. This information provides evidence of the behavioral outcome that the student reflects on as the inspection process progresses. As the student has the opportunity to reflect of the quality of the inspection, and can elect to hold inspection again or move to next topic's inspection. The

recognition of the need to re-inspect the requirements, based on the performance of the inspection, further supports the acquisition of the Analysis level of understanding.

The subsequent development phases, design, implementation, and testing, are not shown in detail since those phases are outside of the scope of the research. However some representation of these phases is needed in order to provide a feeling of continuation for the project. In addition the student's choices during the requirements engineering activities (represented in the simulation) do have consequences in these subsequent phases that need to be represented. The feedback is verbal, from the student's simulated teammates who "work" on the design, implementation, or testing. The verbal feedback is meant to remind the student that he/she is part of a team and that each decision affects the team. Rather than displaying a set of numbers, the student reads the comments and questions from his/her teammates. To remind the student that the previous choices can have an additional impact, the schedule and budget counters are displayed. These counters may increase at this stage of the simulation. For example, an inaccurate requirements set will result in questions and complaints from the teammates and rework is needed.

Before the maintenance phase is portrayed, the product must be delivered to the customer – even if the customer is simulated. The delivery is realized as the point where the customer offers verbal feedback to the student. The feedback represents the customer's sentiment about the product's quality. Quantitative feedback is presented, in the form of the budget and schedule counters. These counters show the student the state of the project schedule and budget when the project is delivered. The presentation of

these counters offers closure, whereby the student can reflect on how his/her decisions resulted in the extent of schedule or budget issues.

The final activity is the Requirement Management activity. During this activity, the goal is for the student to ascend to the Application level of understanding, based on Bloom's Taxonomy. In this activity, the new situation is conducting the inspection since the student does not manage requirement changes during the class project. The knowledge of change submissions is the previously acquired information. In this activity, the specific topics involved are the role of requirements management throughout the lifecycle, the need to have a procedure in place, and the analysis of proposed changes. In order to provide some parameters for the simulation, the application of the requirements management process is limited to the maintenance. Applying the requirements management process to maintenance is intended to demonstrate the process at a point in the lifecycle other than the requirements phase. The structure for this process is provided by the four change submissions. Each change submission describes the desired change, the origin of the submission, and the time estimated to implement the change. In addition, the student is provided with heuristics to use in the analysis of the change submissions. The student solves the problem of classifying the change by selecting the priority of the change. The choices are to implement the change immediately, implement the change in the next version, or to delay the change until a future version (if at all). The resulting feedback consists of verbal feedback from customer and developers. This feedback allows the student to reflect upon his/her choice as the next change submission is examined. As is the case of other verbal feedback, the student hears from those individuals who are affected by his/her decision. Even if the class project entailed

requirement management. The feedback would be limited to instructor feedback, which is not as meaningful or realistic as the feedback from the customer or teammates.

4.5 Scope of the Model

Just as the overall simulation has boundaries, so does the underlying model. Like the interaction portion (the front end) of the simulator, the entire development lifecycle is represented. The emphasis of the model is the requirements phase, but the design, implementation, and testing phases for the core increment (or the initial version if the Waterfall model is being utilized) are included in order to allow the choices from the requirements phase to filter through the lifecycle.

The model is at a high level of abstraction. Such abstraction is contrary to the detailed models used in other simulators (Tvedt, 1996) that were tailored to mimic reality in as much detail as possible. While general accuracy is desired, the intent for the model is to reflect the more general patterns of interaction between requirements engineering and development (as reflected by the overall schedule, cost and quality) in order to assist students in the learning process. As a result, students can observe how their interactions affect the overall project rather than minute details that are not addressed by the instructional topics (e.g. staffing requests).

4.6 The Development of the System Dynamics Model

The main purpose of the underlying System Dynamics Model is to represent the affects of the quality of requirement elicitation, analysis, and inspection on a project's

overall schedule and cost. The model is used in conjunction with the interface (the Interaction Layer).

During the initial, analysis phase of this research project the processes involved in the Requirements phase were studied and modeled. The relationships between the various tasks, activities, products, and people are modeled according to System Dynamics Modeling. The model itself was developed using a standard technique using the depiction of cause-effect and feedback loops according to standard iThink notation. The model is primarily straightforward in nature, as it reacts to the student's input as the project progresses. The model is in Appendix D.

Variables in the model that are input by the Interaction Layer are named *SIMname*, in order to provide easy recognition to the developer and to the reader. These variables are sent to the model after the Interaction Layer completes any necessary calculations. In order to calculate values that represent the impact of the decisions made up till the current activity, default values are used in the simulator until the Interaction Layer submits new values that represent the student's selections.

At the beginning of the simulation, the student selects an elicitation technique to use in the project. In order to accommodate the need for the student to choose an elicitation technique, the model includes switches for both the Facilitated Meeting and Interview techniques that are set to 1 when the technique has been selected and to 0 when the technique has not been selected. These switches are set by the Interaction Layer. Both switches are set to the appropriate value. The rate of the requirement elicitation differs depending on which elicitation technique is selected.

During the Facilitated Meeting technique, the rate of eliciting requirements is influenced by the effectiveness of the facilitator (default is 100), stakeholder buyin (default is 100), the factor representing that the correct stakeholders were selected (SIM Representation of Stakeholders), and the number of meeting sessions conducted (SIM Number of FM Sessions Multiplier). The number of meeting sessions multiplier is used, along with the number of planned meeting sessions, to determine the ongoing meeting schedule and cost overruns. The schedule and cost overrun values are used in the ongoing calculation of schedule and cost overruns, calculations handled in another part of the model. The rate of requirement elicitation determines the set of raw requirements that will be analyzed in the subsequent activity.

During the Interview technique, the rate of eliciting requirements is influenced by the quality of questions (SIM INT Quality of Questions Factor), the effectiveness of the requirements engineer (default value is 100), and the buyin of the stakeholder (default value is 100). The quality of the questions (an ongoing average for the interview sessions conducted) is along with the number of planned interview sessions, to determine the cost overrun value. The schedule and cost overrun values are used in the ongoing calculation of schedule and cost overruns, calculations handled in another part of the model. The rate of requirement elicitation determines the set of raw requirements that will be analyzed in the subsequent activity.

At the conclusion of the requirement analysis activity, the Interaction Layer calculates the percent of requirements that were correctly identified by the student for each analysis activity (e.g. scope, priority). These values are sent to the model to populate the variables, SIM Req Scope Efficiency Factor, SIM Func Nonfunc Req

Analysis Efficiency Factor, SIM Req Prioritization Efficiency Factor, and SIM Req Volatility Efficiency Factor. These values are used to extrapolate the quality of the requirement analysis for all requirements in the system, and participate in the subsequent calculations of anticipated budget and cost overruns. The average of the efficiency factors (the percent of correctly classified requirements for the requirement types) affect the Set of Analyzed Requirements that will be developed into the product. The average also determines the extent of any schedule and cost overruns.

The rate of Requirement Validation is affected by SIM Percent Assessed Correctly VAL, the percent of requirements that are assessed correctly in the validation activity. This value, sent by the Interaction Layer, is the average for all inspections. The quality of the inspection is reflected by this value. As each inspection is conducted, the Interaction Layer resends the new value in order to reflect all inspections through the current one. The quality of the inspection is used to determine any schedule or cost overruns (e.g. Actual Cost VAL, Actual Validation Result Time Usage VAL) that are derived from the Requirements Validation activity.

The rate of implementing the requirements is influenced by the overall quality of the requirements (Overall Quality of Requirements Factor) and any additional work that needs to be done (Add'l Work). The overall quality of the requirements is an average of the average percentage of the quality factor for the selected elicitation technique, the average percentage of correctly classified requirements during the Requirements Analysis activity, and the percentage of requirements assessed correctly during the Requirements Validation activity. The Additional Work is comprised of the rate of new requirements introduced to the system. If the requirements volatility flag is set to 1 (as it is by

default), then the rate of new requirements (the Additional Work) is set by the Interaction Layer (SIM percent of new requirements work).

The schedule and cost overruns are calculated separately by the model. Cost overruns are calculated using the overruns from the selected elicitation technique, the analysis activity, and the validation activity. The overrun is the difference between the estimated budget and the overruns from all traversed activities. The schedule overruns are calculated using the overruns from the selected elicitation technique, the analysis activity, and the validation activity. The overrun is the difference (in days) between the estimated schedule and the overruns from all traversed activities.

The outputs from the model are the ongoing schedule and cost overruns calculations. The Interaction Layer retrieves these values and displays them, along with any other feedback, on the status screen. The student uses this feedback to assist him or her in determining how to proceed in the simulator.

4.7 The Validation of the Simulator

Model validation is discussed in a variety of research applications of simulation models (Sterman, 1992; Sycamore, 1996; Gilbert & Troitzsch, 1998). Several tests exist including the ability of a model to replicate past system behavior, historical fit, and sensitivity analysis. In order to accomplish this, tests can be conducted through comparison against survey results or even expert opinion. The goals of validation are to show that the model is suitable for the intended purposes and that it is consistent with reality (Madachy, 1996). To accomplish these goals, multiple tests are often conducted on the model structure and behavior, in order to filter out ineffective models.

When research is undertaken to realistically model a real system as closely as possible, great care is taken to validate the model. For example, John Tvedt conducted multiple tests when validating his model, simulating the impact of process improvements on software development cycle time (Tvedt, 1996). First, the model's structure was validated after consultation with experts (experienced software engineers) of the real system. A parameter values test was also conducted in order to demonstrate that the model's calibration input values consistently portray the information gathered about the real system. In this case, the information was gathered from research literature and a local software development company. The behavior of the model was tested via model execution and output comparison with the expected or observed values from reality. The degree of accuracy is critical during the behavior tests. Four types of tests were undertaken, where the model was tested in its ability to replicate reference behaviors, under extreme conditions, with surprise behaviors, and through observed system behaviors using an actual project as a statistical comparison. The first three tests were conducted simultaneously through a series of scenarios.

The model (and the simulation in general) in this research endeavor plays a key role in the student's learning experience. This learning experience, involving the recognition of the value and need of Requirements Analysis and Specification activities in a long-term project, is the foundation of this research. The primary objective of this simulator (and model) is not to model the realism of the development phases of a large project. Instead the objective is to define and evaluate a model to provide students with the knowledge and skills normally acquired while working on a large project. The model portrays the activities needed to gather, analyze, validate, and manage the requirements

involved in a large software project. This portrayal of these activities does not include the realistic depiction of the nuances involved in development. The model is more abstract than the other models that have been presented previously (see Chapter 2 for examples).

The Interaction Layer interacts with the model in order to provide the student with the illusion of interaction in a project. The model's primary focus is to portray the consequences of poor requirements in the schedule and budget as the student progresses through the exercise. Unlike industry, the student does not experiment with the model directly and the student does not interpret the model's output directly (in the form of graphs, charts, or instrument displays). Instead the model, in conjunction with the Interaction Layer, provides an experience for the student that allows him/her to learn about the components involved in development. As such, the validation activities described previously were not employed here. The tests did not correspond to the objectives of the simulator or the model. If the model (and simulator) was intended to realistically model a long-term project, then such rigorous validation is called for. However the abstract, instructional nature of the simulator makes the gathering of data from a real system difficult at best. In addition, the objectives of the simulation clash with the parameter tests and behavioral tests that are used to validate other models.

Instead the simulator was checked to make sure that the results are reasonable. If the simulator does not produce results that are sensible, then the objectives cannot be assessed. The simulator and model are intended to represent the general correlations that exist in industry between choices and their impacts on a project's quality, cost, and schedule. In order to test the validity of the simulator and model, a variety of inputs were

entered and the interface's and model's output were assessed in order to ascertain whether the results were reasonable. Even though some aspects of the simulator are randomized, the types of inputs do not change. The only variant is whether the Facilitated Meeting or Interview technique is selected as the requirement elicitation technique.

Each type of scenario was tested as a pair, once for the Facilitated Meeting technique and once for the Interview technique. The first pair of scenarios tested were projects that were on-time and within budget (no overruns) for each type of requirement elicitation technique. For each type of elicitation technique, the appropriate inputs were entered that would result in a perfect requirement set. The results from the interface were highly favorable feedback from stakeholders, teammates and the customer (upon delivery). The results from the model, displayed in the interface, were no schedule or cost overruns – just as predicted.

The second pair of scenarios tested was for projects that had a poor set of requirements and should result in serious schedule overruns, cost overruns, and appropriate feedback. For each elicitation technique, the appropriate inputs were entered that would result in a very poor quality requirement set, where the quality factor is below 40 percent. The results from the interface were very negative feedback from stakeholders, teammates and the customer (upon delivery). The results from the model, displayed in the interface, were extremely high schedule or cost overruns – just as predicted. The schedule overrun was 620 days and 714 days (over the 700-day schedule) while the cost overrun was \$1,375,000 and \$1,571,166 (over the \$1.5 million dollar budget).

The third pair of scenarios tested were for projects that have a good requirements set and should result in a minor schedule delay (within 15% of the 700 day schedule), cost overruns, and appropriate feedback. For each elicitation technique, the appropriate inputs were entered that would result in a good quality requirement set (with a factor between 80 and 90). The results from the interface were appropriate feedback from stakeholders, teammates and the customer (upon delivery). The feedback reinforced the high quality of relevant requirements while pointing out relevant problems. The results from the model, displayed in the interface, were moderate schedule or cost overruns – just as predicted. The requirements set quality factors were about 81 and 85.7 respectively (out of 100). The schedule delays were about 90 days and 70 days (of the estimated 700-day schedule) while the cost overruns were about \$216,000 and \$135,000 (of the \$1.5 million dollar budget).

These tests show that the simulator yields reasonable results for the extreme requirement quality and for a project that the researcher considers to be a reasonable effort by a prospective user. The results provide a basis for assessment of the simulator.

4.8 Summary

This chapter presents the design and implementation details of the front-end (the interaction layer) and the back-end (the model layer) of the simulator. The flow of interaction is presented in detail, as is the interaction between the interaction layer and the model. The methodology of the design of the simulator is also presented. In addition, the need for the simulator's validation is described. The next chapter presents the assessment of the research in terms of a case study.

Chapter 5. Case Study

Introduction

The objective of this chapter is to provide background information regarding the design and execution of the case study for the simulation and the hypothesis. Section 5.1 presents the hypothesis. Section 5.2 describes the design of the assessment used in the pilot case study and the case study. Section 5.3 presents the procedure used to execute the pilot case study. Section 5.4 presents additional considerations taken into account during the design of the pilot case study study. Section 5.5 presents the analysis of the pilot case study results. Section 5.6 presents the procedure used to execute the case study. Section 5.7 describes additional considerations taken into account for the case study. Section 5.8 presents the analysis of the case study. Section 5.9 provides a summary of analysis.

5.1 Hypothesis Revisited

In order to assess the impact of the simulator in terms of instructional effectiveness, the following hypothesis is asserted:

H1: Undergraduate students using the simulator will increase their level of understanding, based on Bloom's Taxonomy, of the Requirements Analysis and Specification activities utilized in a long-term project to a greater extent than with a traditional course without a simulator.

This hypothesis is intended to address the assertion that students who use the simulator will benefit in their increased ability to understand a variety of Requirements Analysis and Specification tasks in a large project. As the simulator is a supplement to the course,

rather than a substitute to the course, a group of students utilized the simulator in conjunction with regular class lectures and the course project. In order to test the hypothesis, the group of students was administered a pretest and a posttest. The assessments enabled data to be collected showing the students' ability to recognize and judge the value and need of Requirements Elicitation, Requirements Analysis, Requirements Validation, and Requirement Management. The collected data was analyzed and is presented in Section 5.5.

5.2 Design of the Assessment

The pretest and posttest are the same test. The questions are multiple-choice in order to expedite grading and data analysis. The questions were developed in order to test whether the student understands the SWEBOK topic at a specific level in Bloom's taxonomy. The anticipated levels of student understanding for the topics before and after the use of the simulation are shown in Table 8. These levels reflect the extent of understanding for each area that each student is expected to have before the pretest (without use of the simulator) and after the posttest (after use of the simulator). The mapping of student understanding to the SWEBOK topics was discussed previously in Chapter 3.

Table 8

Simulation Topic Mapping to Bloom's Taxonomy

SOFTWARE REQUIREMENTS ANALYSIS TOPICS FROM SWEBOK	ASSUMED LEVEL OF UNDERSTANDING	TARGET LEVEL OF UNDERSTANDING
--	---	--

	(BEFORE)	(AFTER)
II. Requirements Elicitation		
B. Elicitation Techniques		
1. Interviews	Application, only with the instructor	Application
3. Facilitated Meetings	Comprehension	Application
III. Requirements Analysis		
A. Requirements classification		
1. Functional & Nonfunctional	Comprehension, for development of Nonfunctional list	Comprehension
4. Priority	Knowledge	Comprehension
5. Scope	Knowledge	Comprehension
6. Volatility	NA	Comprehension
V. Requirements Validation		
A. The conduct of requirements reviews		
1. Group composition is appropriate (may include customer)	Comprehension	Application
2. Use of guiding documents like checklists to guide review and to doc findings	Comprehension, as docs used for recording only	Analysis
3. Review process is at specified checkpoints and redone as appropriate	Application, but done once	Analysis
VI. Requirements Management		
A. Change management		
1. Understanding the role of Change Management throughout lifecycle	Comprehension	Application
2. Have procedure in place	Comprehension	Application
3. Analyze proposed changes	Comprehension	Application

These anticipated levels of understanding correspond to the assessment design.

The individual assessment questions are traceable to the topics covered in the simulator.

The full assessment is in Appendix F, and the assessment's traceability matrix is presented in Appendix G. Nearly all of the assumed levels of understanding and all of the target levels of understanding (in Table 8) correspond to the assessment questions. Some additional assessment was included. In order to assess the extent of student understanding of the various topics, some questions were added, beyond the assumed and target level of understanding, to assess understanding at the Knowledge and Application levels for most of the Requirements Analysis classification types. Minor discrepancies exist between the table depicting the anticipated levels of understanding (table 8) and the traceability matrix in Appendix G due to the limited undertaking of some topics in lecture or in the project. In order to compensate for the partial topic understanding, the assessment tests for understanding at one level below (as evident in the traceability matrix). The areas of discrepancy are shaded in Table 8.

Before the full case study was conducted, the assessment was piloted with recent CSE 360 students. These three students provided feedback, including the quality of writing on the instructions and content. The result of all feedback resulted in the final version of the assessment that was used to carry out the case study.

5.3 Pilot Case Study Procedure

The simulation was tested with students enrolled in CSE 360, Introduction to Software Engineering, at Arizona State University. The experimental design was a One-Group Pretest-Posttest Design (Gall, Borg & Gall, 1996). The experimental group participated in course lectures and used the simulator. As the intent for the study is to

assess how effective the addition of the simulator is to the course. this experimental design is appropriate (Gall, Borg & Gall, 1996).

Students enrolled in the researcher's section during the Spring 2002 semester participated in the case study. In accordance with Human Subjects procedures, participation in the study was voluntary. The experimental group contained 17 students (14 of whom completed the entire case study), and were identified with a codeword of their choosing. The students did not receive feedback on their pretest performance. After the administration of the pretest, the experimental group utilized the simulator ten days. Students were asked to use the simulator once using each Requirements Elicitation technique, allowing them to explore the entire simulator. Proof of completion (besides the assessments) was the submission of two reports generated by the simulator, one report for each elicitation technique type. In order to assess the effectiveness of the simulation, the participants were administered the pretest and posttest on the topics covered by the simulation. Since the simulator is intended as a supplement to the course, the case study was conducted near the end of the course. The case study was conducted well after lecture (and project) was conducted on the topics portrayed in the simulator. Also, this timing was chosen since the use of the simulator during the critical periods, the requirements and design phases, of the team project would have been overwhelming to the students.

Since the students volunteered to participate in the study, attrition was not considered to be a significant threat. Since the course is quite large, the 17 participants was acceptable to test this proof-of-concept and withstood the 3 students who withdrew

from the case study. In addition a raffle of gift certificates, computer books and other small prizes was used to compensate students who participated.

Even with such compensation, the recruitment of participants was difficult. Volunteers were required and anonymity was paramount, in order to comply with Human Subjects regulations. As such, no extra credit for participation was allowed. Such motivation may have encouraged more participation.

5.4 Additional Pilot Case Study Design Considerations

Since the students were from the same class, the possibility that students discussed the simulator was a real one. The content of the simulator was designed to provide several different versions in order to provide different experiences. As such, very few students viewed the same content in the simulator thus minimizing the possibility of sharing information. In terms of external validity participants cannot be fully generalized to the general population of undergraduate students in an introductory Software Engineering course. Due to the fact that the participants were volunteers whose diversity could not be controlled, the 14 participants serve more as a group testing the potential of the simulator and hypothesis rather than an absolute answer. Further research with a larger, broader set of students is needed in order to generalize the results to all students.

In terms of ecological validity, the experiment is repeatable. The experimental details and the assessment content is presented in enough detail so that it can be repeated in another instructional setting. Since the students are only exposed to the single type of treatment (the simulator), multiple-treatment interference is not an issue in this case study

(Gall, Borg & Gall, 1996). Students in the experimental group received no additional instruction (other than that derived from the lecture and the text), thus addressing the Hawthorne Effect. Also, the use of the codewords and any submission of materials was completely anonymous. As such, each student participated with complete confidence that his/her privacy was intact, without any intentional or unintentional influence by the researcher. Unlike instances of testing student attitudes, this experiment addressed Pretest sensitization due to the testing concepts and their application. In order to address the interaction of the time of the posttest and the treatment effects, the posttest was administered within four days following the end of the experiment.

The duration of the experimental treatment was brief, 10 days, thus History and Maturation factors were not an issue in this study. All lecture and course work related to requirements engineering was conducted before the case study. The pretest and posttests were the same, although the questions were in a different order. The intent for placing the questions in a different order was to decrease the possibility that questions were recognized. The students were instructed to answer to the best of their ability, and to not guess answers to questions. While the use of the same test can cause problems with internal validity, potential problems were minimized by resorting the questions and by the fact that the answers were not supplied to the students.

5.5 Pilot Case Study Analysis

The results were analyzed from the 14 students who completed both the pretest and the posttest. Each main topic area (requirements elicitation, requirements analysis, requirements validation, and requirements management) is presented separately. Overall,

students increased their knowledge in requirements engineering. In order to ascertain whether the amount of improvement is statistically significant for this small population, the Wilcoxon test was used. The Wilcoxon test is used to see whether the median in two populations differ in size, especially when the sample population is small and in an unknown distribution (Hollander, 1999). In this case study, the two samples, are the pretest and posttest score for each participant in the group. The key to this test is to work with the difference in the posttest score from the pretest score (where 1 means correct and 0 means incorrect). If no change (no improvement) has occurred, then the median difference between the pretest responses and the posttest responses for a group is 0. If change (improvement) has occurred, then the median difference is less than 0 between the pretest responses and the posttest responses for a group. The results have a 94.8% confidence level. The statistics were calculated using Minitab for Windows version 13.

The extents of the gains vary from topic to topic, as the subsequent sections will present. In the tables where results are summarized, the question numbers used are those from the Pretest in order to provide consistency.

5.5.1 Requirements Elicitation

The students improved their understanding of both the facilitated meeting and interview techniques. Moderate gains in knowledge resided in the Comprehension level rather than at the Application level of Bloom's Taxonomy, as presented in Table 9 and Table 10.

Table 9

Summary of results for the facilitated meeting technique

Assessment	Facilitated Meeting - Comprehension Level % Correct (#1)	Number of Students who Answered Correctly	Facilitated Meeting - Application Level % Correct (#23)	Number of Students who Answered Correctly
Pretest	42.9	6	14.3	2
Posttest	67.3	9	21.4	3

The increases in the facilitated meeting technique was 24.4%, with 3 more students answering the question correctly, and the increases in the interview meeting technique was 28.6%, with 4 more students answering the question correctly. The increases in the facilitated meeting technique was 24.4%, with 3 more students answering the question correctly, and the increases in the interview meeting technique was 28.6%, with 4 more students answering the question correctly.

Table 10

Summary of results for interview technique

Assessment	Interview - Comprehension Level Average % Correct (#2)	Number of Students who Answered Correctly	Interview - Application Level Average % Correct (#12)	Number of Students who Answered Correctly
Pretest	57.1	8	21.4	3
Posttest	85.7	12	14.3	2

While the number of students who answered the Application level questions was consistently quite low, a possible reason is in the nature of the questions. The Application-level questions required students to select four items to create the correct

answer. With both questions, only 2 to 3 students selected all of the correct portions of the answer. The number of students who improved their scores by selecting more correct items in the posttest than in the pretest was significant. In the facilitated meeting question, 9 of the 14 students improved the number of correct items selected. Of the remaining 5 students, 4 had no change in the number of correct answers and 1 showed a decrease in the number of correct answers. In the interview question, 9 of the 14 students improved the number of correct items selected in their answers. Of the remaining 5 students, 3 had no change in the number of correct answers and 2 showed a decrease in the number of correct answers. Out of all student responses in this topic area, a 50% percent increase in correct answers exists between the pretest and posttest.

Further analysis was conducted in order to assess whether the improvement was statistically significant. For the Facilitated Meeting technique, the results were mixed. The improvement at the Comprehension level was not statistically significant ($p = .176$), but the improvement at the Application level was statistically significant ($p = .006$). Similarly, the results for the Interview technique were mixed. At the Comprehension level, the results were not statistically significant ($p = .05$), but the improvement at the Application level was statistically significant ($p = .015$).

The improvement, especially at the Application level, can be attributed to the simulator. The detailed simulated process of eliciting requirements allows students to participate in the lengthy process of either the facilitated meeting or interview technique. Students are able to explore the ongoing process, including learning from failure to ask the right questions or to interact with the right people. Without the simulator, students

rely on asking the instructor questions and reading the project description. The simulator has added value in this topic area.

5.5.2 Requirements Analysis

In the past, the researcher noticed that students had difficulty with basic understanding of nonfunctional requirements. As such, the questions regarding requirements analysis included understanding at the Knowledge level of Bloom's Taxonomy, in addition to questions at the Comprehension and Application levels. While the intent is for students to increase their level of understanding to the Comprehension level, Application level questions were added in order to further gauge student understanding of the material. These Application level questions coincided with the interactive nature of the simulation, where students applied their knowledge of the various types of requirements analysis.

The students demonstrated greater understanding of requirements analysis at all three levels tested, as shown in Table 11. The largest gains in knowledge resided in the Comprehension and Application level of Bloom's Taxonomy. The gains at the Knowledge level are more modest since an average of 10 students answered these questions correctly during the pretest and an average of 11.6 students answered the questions correctly during the posttest. In regard to the Knowledge question for nonfunctional requirements, only 8 students answered the question correctly during the pretest but 11 students selected the correct answer during the posttest. None of the results are statistically significant, and the small margin for improvement was a factor (see Table 12).

Table 11

Summary of results for requirements analysis

Assessment	Knowledge Level Average % Correct (=3-7)	Average Number of Students who Answered Correctly	Comprehension Level Average % Correct (=13-16)	Average Number of Students who Answered Correctly	Application Level Average % Correct (=17-19)	Average Number of Students who Answered Correctly
Pretest	71.4	10	55.4	7.75	31	4.3
Posttest	82.9	11.6	71.4	10	47.6	6.7

At both the Comprehension and Application levels of Bloom's Taxonomy, students improved their understanding of requirements analysis. More students answered more questions correctly in both levels of understanding. In some topics, Scope and Priority, the number of students who answered the Comprehension level questions correctly was nearly the same (between a 0 and 2 student difference). Volatility and Nonfunctional questions showed a 3-student increase in correct responses. The understanding of Volatility and Scope was also increased at the Application level, with 4 more students and 3 more students (respectively) correctly answering the relevant questions on the posttest (from 4 students during the pretest for both topics).

Out of all student responses in this topic area, a 21.4% percent increase in correct answers exists between the pretest and posttest. Of the remaining students, 66.1% had no change in the number of correct answers and 12.5% showed a decrease in the number of correct answers. The increase in the number of students who understand the concept of Volatility is made possible by the potential for increase, since so few students understood Volatility beyond a basic level. This increase is not statistically significant ($p = .086$).

but it is the closes of all results to being statistically significant. In using the simulator, the students were able to see the dynamic nature of project requirements and the changing needs of the customer (even on the limited scale). During the course, requirements are rarely changed after the requirements phase is completed so that the students have a fair chance to complete the project by the end of the semester.

Table 12

Summary of statistical significant results for requirements analysis

Assessment	p-value (94.8% Confidence interval)	Estimated Median
Knowledge Level (#3)	.977	0
Knowledge Level (#4)	.233	0
Knowledge Level (#5)	.186	0
Knowledge Level (#6)	.186	0
Knowledge Level (#7)	.176	0
Comprehension Level (#13)	.140	0
Comprehension Level (#14)	.394	0
Comprehension Level (#15)	.186	0
Comprehension Level (#16)	.176	0
Application Level (#17)	.091	0
Application Level (#18)	.572	0
Application Level (#19)	.086	-.5

The increase in the number of students who understand Scope at a higher level (Application) is due the fact that the simulator allows the student to explore the concept of scope and make mistakes. During the course, the instructor corrects student project documentation when requirements are added that are out of the project's scope. As a result, the student does not see the consequences of developing requirements that are out-of-scope.

These two topics illustrate how the nature of the large project simulator, allowing students to participate in a seemingly realistic project by interacting with a variety of

people, offers more opportunities for learning. The large project enables students to work on a project where interaction and analysis is needed. The student has to put effort into making the project successful.

5.5.3 Requirements Validation

Although the objective was for students to improve their understanding from the Comprehension and Application levels (for different sub-topics) to the Analysis level of Bloom's Taxonomy, all sub-topics were tested at the Comprehension, Application, and Analysis levels. Overall, the students had virtually no improvement in their understanding of requirements validation. The largest gains seemed to be at the Comprehension level with the virtually no improvement at the Application and Analysis levels of Bloom's Taxonomy. The summary of the results are summarized in Table 12.

Table 13

Summary of results for requirements validation

Assessment	Comprehension Level Questions Average % Correct (#8, 9)	Average Number of Students who Answered Correctly	Application Level Questions Average % Correct (#10,20)	Average Number of Students who Answered Correctly	Analysis Level Questions Average % Correct (#21,22)	Average Number of Students who Answered Correctly
Pretest	53.6	7.5	39.3	5.5	14.3	2
Posttest	67.9	9.5	46.4	6.5	17.9	2.5

The number of students who answered questions correctly was virtually unchanged, although an average of 2 students answered the Comprehension level questions correctly. The number of correct responses in the Comprehension, Application,

and Analysis levels was unchanged, with the exception of the Comprehension level question where 2 more students answered the question correctly.

Out of all student responses in this topic area, a 25% percent increase in correct answers exists between the pretest and posttest. The improvements are derived from the Comprehension level questions and the stakeholder-related question at the Application level. These improvements, most notably with the stakeholder-related questions at the Comprehension and Application level, can be attributed to the emphasis on selecting stakeholders in the simulated inspection process. The improvement at the Application was statistically significant ($p = .023$), while the improvement at the Comprehension level was not statistically significant ($p = .265$) as shown in Table 14.. In using the simulator, the students were able to participate in the inspection process and receive both immediate and delayed feedback from teammates and project stakeholders. During the course, the inspection process is conducted solely within the team and within during the course of one class meeting. Afterwards, the instructor advises the student on issues that are not caught during the inspection. The student relies on such feedback during the project – which is not realistic in industry. A gap still remains in the process whereby analysis is not possible, with only 3 students answering the analysis question correctly. In nearly all cases the results are not statistically significant, as many students' answers did not change at all (correct or incorrect).

Table 14

Summary of statistical significant results for requirements analysis

Assessment	p-value (94.8% Confidence interval)	Estimated Median
Comprehension Level (=8)	.265	0
Comprehension Level (=9)	.233	0
Application Level (=10)	.673	0
Application Level (=20)	.023	=.5
Analysis Level (=21)	.673	0
Analysis Level (=22)	.394	0

Issues do remain in the validation process. Although student improvement is noted at the Application level, more than half of the students still do not understand requirement validation at the Application level of Bloom's Taxonomy. Since the objective was to achieve the Analysis level, work still remains. Such potential for improvement will be addressed in Chapter 6.

5.5.4 Requirements Management

Several students improved their understanding of requirements management. While gains are evident at both the Comprehension level and the Application level, the larger gains are at the Application level of Bloom's Taxonomy. The summary of the correct response results are summarized in Table 15.

Table 15

Summary of results for requirements management.

Assessment	Comprehension Level Questions Average % Correct (=11, 24)	Average Number of Students who Answered Correctly	Application Level Questions Average % Correct (=25, 26)	Average Number of Students who Answered Correctly
Pretest	42.9	6	25	3.5
Posttest	67.9	9.5	53.6	7.5

At both the Comprehension and Application levels of Bloom's Taxonomy, students improved their understanding of requirement management. More students answered more questions correctly in both levels of understanding. At the Comprehension level, 9.5 students answered the questions correctly at the posttest while 6 students answered the questions correct during the pretest. At the Application level, 4 more students correctly answered the relevant questions on the posttest (from 3.5 students during the pretest for both topics).

The noted increase in the number of students who understand requirement management at the Application level is made possible by the potential for increase, since so few students understood the topic beyond a basic level. In using the simulator, the students were able to participate in the change management process and receive feedback from teammates and the customer. During the course, no change management tasks are completed after project delivery. The student relies on course lecture and reading for this material. The simulator offers students the opportunity to truly understand the purpose and process of managing requirements. Out of all student responses in this topic area, a 28.6% percent increase in correct answers exists between the pretest and posttest.

In terms of statistical significance, the results are mixed though promising. In terms of understanding the role of requirements management (question #11), there results were not statistically significant at $p = .233$ with an estimated median of 0. However in terms of understanding the evaluation (at the Comprehension level) and analyzing requirements changes (at the Application level), the improvements for these questions (24 and 25, respectively) are statistically significant at $p = .03$ and $p = .011$, respectively, with an estimated mean of -0.5 in both instances. The results for understanding auditing in the requirements management activity (question #26) is not statistically significant at $p = .5$ with an estimated mean of 0.

Issues do remain with the requirement management process. Although student improvement is noted, many students still do not understand requirement management at the Application level of Bloom's Taxonomy. Potential for improvement exists and will be addressed in Chapter 6.

5.6 Case Study Procedure

In order to compare the impact of the use of the simulator to understanding, a different study was undertaken. During the case study, one group of students used the simulator while the other group did not use the simulator. The environment also differed. The simulation was tested with students enrolled in the undergraduate Software Requirements and Specification course at the Rochester Institute of Technology. The course was a required, upper-division course for students in the Software Engineering program. All of the students were enrolled in the Software Engineering program. The prerequisites for the course are the Introduction to Software Engineering course, the

Design course, and the Formal Methods course. While the Introduction to Software Engineering course provides an overview of Software Engineering, it does not have as much breadth and depth (in some areas) in the topics as the Introduction to Software Engineering course at Arizona State University contains. The subsequent courses at RIT, such as the Requirements course, provide opportunity to learn about the various Software Engineering topics. In addition, RIT follows a 10-week quarter system while ASU follows the semester system.

The experimental design consists of the Pretest-Posttest Control-Group Design (Gall, Borg & Gall, 1996). The members of both the control group and the experimental group were all enrolled in the Requirements and Specification course. The control group participated in the course lectures and assignments, but did not use the simulator. The experimental group participated in course lectures and assignments, while using the simulator. Both groups took the pretest and the posttest. The students were randomly assigned to either the control group or the experimental group. Since the student population is traditional in terms of age and is extremely male-dominated, no attempts were made to equalize the group assignments.

Students enrolled in the course during the Fall 2002 quarter participated in the study. The course was taught by the author. The experimental group contained 12 students (50% of enrollment), and the control group contains 12 students (50% of enrollment). The remaining 3 students did not volunteer for the study. The students were identified with a codeword of their choosing. The pretest was administered at the beginning of the course. The students did not receive feedback on their pretest performance. Upon receipt of the pretest, the students were asked not to guess the

answers to the questions. Instead if the student did not know the answer, then he or she was instructed to select the "I do not know" response.

At the beginning of the fourth week of class, the experimental group utilized the simulation for ten days. Students were asked to use the simulator once using each Requirements Elicitation technique, allowing them to explore the entire simulator. Proof of completion (besides the assessments) were the submission of two reports generated by the simulator, one report for each elicitation technique type. The posttest was administered after the 10-day period allotted for student use of the simulator – this will be at the beginning of the sixth week of class. The experiment was conducted during the lecture on the topics portrayed in the simulator.

As with the pilot case study, the pretest and posttests were the same, although the questions were in a different order. In addition, the assessment was the same one used for the pilot case study. The primary data collected consisted of the assessment responses. Additional data collection included the extent of industry experience, gender, and the extent of English fluency (native, English as a second language non-native). Due to the fact that the additional data presented a picture of a homogeneous group, the additional data was not used to further analyze the assessments results. Specifically, out of the 24 participants 2 students were women, 2 students were not native English speakers, and all of them had at least 6 months of industry experience (due to required co-op experience).

5.7 Additional Case Study Design Considerations

Many of the design considerations addressed for the case study were identical to the those addressed during the pilot case study. The content of the simulator still provided different experiences for the students. As such, few students viewed the same content in the simulator thus minimizing the possibility of sharing information. In terms of external validity participants cannot be fully generalized to the general population of undergraduate students in a Requirements and Specification course as the academic program's co-op requirement is not representative of all universities. This industry experience may influence the results. Also, the participants were not diverse in demographics though they are representative of students at RIT. Further research with a larger, broader set of students is needed in order to generalize the results to all students.

Like the pilot case study, the experiment is repeatable. Thus, ecological validity is addressed. The experimental details and the assessment content is still presented in enough detail so that it can be repeated in another instructional setting. Multiple-treatment interference is still not an issue in this case study since the students are only exposed to the single type of treatment (the simulator), (Gall, Borg & Gall, 1996). Students in either group received no additional instruction (other than that derived from the lecture and the text), thus addressing the Hawthorne Effect. Also, the use of the codewords and any submission of materials was completely anonymous. As such, each student participated with complete confidence that his/her privacy was in tact, without any intentional or unintentional influence by the researcher. The case study also addressed Pretest sensitization through the testing of concepts and their application. In

order to address the interaction of the time of the posttest and the treatment effects, the posttest was again administered within four days following the end of the experiment.

Like the pilot case study, the duration of the experimental treatment was brief, 2 weeks, thus History and Maturation factors was not an issue. All lecture and course work related to requirements engineering topics contained in the simulator was conducted before the case study.

5.8 Case Study Analysis

The results were analyzed from the 24 students who completed both the pretest and the posttest. The questions with each main topic area (requirements elicitation, requirements analysis, requirements validation, and requirements management) is presented separately. The students from this case study scored much higher than the students from the pilot case study. This fact allowed for only small room for improvement in many cases. Despite this circumstance, students increased their knowledge in some areas of requirements engineering.

As with the pilot case study, the Wilcoxon test was used in order to ascertain whether the amount of improvement is statistically significant for this small population. In this case study, the two samples, are the pretest and posttest score for each participant in the group. The key to this test is to work with the difference in the posttest score from the pretest score (where 1 means correct and 0 means incorrect). If no change (no improvement) has occurred, then the median difference between the pretest responses and the posttest responses for a group is 0. If change (improvement) has occurred, then the median difference is less than 0 between the pretest responses and the posttest responses

for a group. The results have a 94.5% confidence level. The statistics were calculated using Minitab for Windows version 13.

The subsequent sections present the extent of the gains by topic. In the tables where results are summarized, the question numbers used are those from the Pretest in order to provide consistency.

5.5.1 Requirements Elicitation

Both the experimental group and the control group improved their understanding of the elicitation techniques at the Comprehension level. The increase is nearly identical in the case of the Facilitated Meeting (Table 16), although according to the Wilcoxon test the experimental groups increase is statistically significant ($p < .05$). The increase is identical in the case of the Interview technique (Table 17). For both groups, the increase from 10 to 12 students having answered the question correctly is not statistically significant ($p > .05$).

Table 16

Summary of results for the facilitated meeting question #1

Assessment	Facilitated Meeting – Comprehension Level # People Correct (#1) Pretest	Facilitated Meeting – Comprehension Level # People Correct (#1) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	6	11	.03	-.5
Control Group	7	10	.091	0

Table 17

Summary of results for interview question #2

Assessment	Interview – Comprehension Level = People Correct (=2) Pretest	Interview – Comprehension Level = People Correct (=2) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	10	12	.186	0
Control Group	10	12	.186	0

As with the pilot case study, the number of students who answered the Application level questions correctly was quite low for both groups (see Table 18 and Table 19). The Application-level questions required students to select four items to create the correct answer.

The number of students who improved their scores by selecting more correct items in the posttest than in the pretest was not significant in the overall quantity. However in terms of the understanding of the facilitated meeting technique at the application level, the control group had a larger increase in the number of correct responses. The amount of increase is statistically significant for the control group, while the increase by the experimental group is not. The values recording the number of students who correctly answered the questions is misleading as the question required four items to be selected. Very few people in either group correctly identified all four items, but many participants in both groups increased the number of items correctly identified.. In terms of understanding of the interview technique at the application level, the results were the opposite as the experimental group had a statistically significant amount of

improvement. The control group also had some improvement, though it was not statistically significant. The interview question at the application level was similar in format to the facilitated meeting question in that four items need to be selected in order to form a correct answer.

In terms of an effect with the facilitated meeting technique (at the application level), the simulator did not make a difference. In terms of an effect with the interview technique (at the application level), the simulator did make a difference. The difference in result between the two techniques, is worth noting although the difference in the number of students who answered the questions correctly is not very high.

However the control group did not perform any better than the experimental group. Since the class project gave all students the opportunity to elicit questions from stakeholders, the lack of any significant is disappointing even in a general sense. More assessment may be needed in order to further pinpoint strengths and weaknesses in this area.

Table 18

Summary of results for the facilitated meeting question #23

Assessment	Facilitated Meeting – Application Level = People Correct (#23)	Facilitated Meeting – Application Level = People Correct (#23)	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	2	4	.071	-.5
Control Group	1	2	.007	-1

Table 19

Summary of results for interview question #12

Assessment	Interview – Application Level = People Correct (=12) Pretest	Interview – Application Level = People Correct (=12) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	1	4	.011	-.5
Control Group	1	2	.071	-.5

5.5.2 Requirements Analysis

Both the experimental and control groups demonstrated an increase of understanding at the Knowledge and Comprehension levels, though the increases are not statistically significant (with minor exception). At both the Pretest and the Posttest, the scores were nearly identical for both groups (see Tables 20-28). In nearly all cases, except for Question 16, the pretest scores were very high and not much room for improvement existed.

In only two questions, was there improvement by a group. Question 4, dealing with scope at the Knowledge level, the experimental group improved their understanding while the control group did not (see Table 21). The simulator had some impact in this area, though slightly more than the control group's improvement. . Question 16, addressing volatility at the Comprehension level, the control had statistically significant improvement over the experimental group (see Table 28). In both cases, the overall difference between the groups is 1 person. In this area, the simulator had no impact.

Table 20

Summary of results for requirements analysis question #3 - priority

Assessment	Knowledge Level = People Correct (=3) Pretest	Knowledge Level = People Correct (=3) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	11	12	.05	0
Control Group	11	12	.5	0

Table 21

Summary of results for requirements analysis question #4 - scope

Assessment	Knowledge Level = People Correct (=4) Pretest	Knowledge Level = People Correct (=4) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	8	12	.05	-.5
Control Group	8	11	.091	0

Table 22

Summary of results for requirements analysis question #5 - volatility

Assessment	Knowledge Level = People Correct (=5) Pretest	Knowledge Level = People Correct (=5) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	12	12	Not applicable, since all responses were correct	Not applicable, since all responses were correct
Control Group	9	12	.091	0

Table 23

Summary of results for requirements analysis question #6 – requirement type

Assessment	Knowledge Level = People Correct (=6) Pretest	Knowledge Level = People Correct (=6) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	10	11	.5	0
Control Group	11	12	.5	0

Table 24

Summary of results for requirements analysis question #7 – requirement type

Assessment	Knowledge Level = People Correct (=7) Pretest	Knowledge Level = People Correct (=7) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	9	12	.091	0
Control Group	10	12	.186	0

Table 25

Summary of results for requirements analysis question #13 – requirement type

Assessment	Comprehension Level = People Correct (=13) Pretest	Comprehension Level = People Correct (=13) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	11	12	.5	0
Control Group	9	11	.186	0

Table 26

Summary of results for requirements analysis question #14 - scope

Assessment	Comprehension Level = People Correct (=14) Pretest	Comprehension Level = People Correct (=14) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	8	10	.233	0
Control Group	9	10	.395	0

Table 27

Summary of results for requirements analysis question #15 - priority

Assessment	Comprehension Level = People Correct (=15) Pretest	Comprehension Level = People Correct (=15) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	9	11	.186	0
Control Group	9	11	.233	0

Table 28

Summary of results for requirements analysis question #16 - volatility

Assessment	Comprehension Level = People Correct (=16) Pretest	Comprehension Level = People Correct (=16) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	6	9	.091	0
Control Group	6	10	.05	-.5

The positive impact of the simulator for the Application level is limited. In Table 29, the experimental group's improvement is shown as being statistically significant while the control group had no improvement. In this area, and the other two questions at the Application level, both groups had an increase in the number of correct responses

though the overall increase was not significant (see Table 30 and Table 31). The simulator only had an impact in the area of Scope, as happened at the Knowledge level.

Table 29

Summary of results for requirements analysis question #17 - scope

Assessment	Application Level = People Correct (=17) Pretest	Application Level = People Correct (=17) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	3	8	.03	-.5
Control Group	4	6	.233	0

Table 30

Summary of results for requirements analysis question #18 - priority

Assessment	Application Level = People Correct (=18) Pretest	Application Level = People Correct (=18) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	4	7	.140	0
Control Group	8	8	.572	0

Table 31

Summary of results for requirements analysis question #19 - volatility

Assessment	Application Level = People Correct (=19) Pretest	Application Level = People Correct (=19) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	8	11	.091	0
Control Group	8	10	.186	0

The common improvements made by both groups may be attributed to the course project rather than the simulator. During the course project, the students applied the

concepts and techniques used in class on a quarter-long project that solely concentrated on the requirements engineering process. As such, the students had hands-on experience that provided more opportunity than the students in the pilot case study received.

5.5.3 Requirements Validation

In the area of Requirements Validation, both groups of students had statistically significant improvement at the Comprehension level (see Table 32 and Table 33) and the Application level (see Table 34 and Table 35). The experimental group did have a larger improvement than the control group at the Application level, as shown in Table 35. The simulator did not have an impact in this area. While the concepts addressed in these questions were represented in the simulator but not in the class project, though they were presented in lecture. The improvement is most likely attributed to the lecture, since it is the common element.

Table 32

Summary of results for requirements validation question #8

Assessment	Comprehension Level Question # People Correct (=8) Pretest	Comprehension Level Question # People Correct (=8) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	5	12	.011	-.5
Control Group	6	12	.018	-.5

Table 33

Summary of results for requirements validation question #9

Assessment	Comprehension Level Question = People Correct (=9) Pretest	Comprehension Level Question = People Correct (=9) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	7	11	.05	-.5
Control Group	5	10	.03	-.5

Table 34

Summary of results for requirements validation question #10

Assessment	Application Level Question = People Correct (=10) Pretest	Application Level Question = People Correct (=10) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	4	11	.011	-.5
Control Group	2	8	.018	-.5

Table 35

Summary of results for requirements validation question #20

Assessment	Application Level Question = People Correct (=20) Pretest	Application Level Question = People Correct (=20) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	1	9	.005	-1
Control Group	2	7	.011	-.5

While both groups generally made the same progress at the Analysis level, as shown in Tables 36 and 37. The experimental group did make statistically significant progress in for question 21 (see Table 36), while the control group did not have such improvement (they had no change whatsoever). An anomaly existed in the pretest results

for this question as the control group had a far greater number of students who answered the question correctly than existed in the experimental group. While an impact from the simulator may have occurred, more study is needed due to the anomaly.

Both groups had no overall improvement in their understanding for question 22 (see Table 37). It is worth noting that both groups had a very high number of students who answered the question correctly initially, so there was not much room for improvement.

Table 36

Summary of results for requirements validation question #21

Assessment	Analysis Level Question = People Correct (=21) Pretest	Analysis Level Question = People Correct (=21) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	2	8	.018	-.5
Control Group	7	7	.572	0

Table 37

Summary of results for requirements validation question #22

Assessment	Analysis Level Question = People Correct (=22) Pretest	Analysis Level Question = People Correct (=22) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	9	10	.395	0
Control Group	10	11	.395	0

Both groups achieve the Analysis level, basically due to the fact that they already achieved it at the pretest. The difference between the pilot case study population and the case study population is staggering. Further study is needed to rectify this in terms of generalizing to the general population. Such potential will be addressed in Chapter 6.

5.5.4 Requirements Management

The results from the pretest and posttest were interesting, not so much in the any significant improvement of the students but rather in the lack of any need for improvement. The levels of understanding, at the Comprehension and Application levels, were very high, as shown in Tables 38 to 41. Not much room for improvement remained for either group, and the calculated significance (or lack thereof) reflects that.

Table 38

Summary of results for requirements management question #11

Assessment	Comprehension Level Question = People Correct (=11) Pretest	Comprehension Level Question = People Correct (=11) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	11	12	.5	0
Control Group	10	11	.395	0

Table 39

Summary of results for requirements management question #24

Assessment	Comprehension Level Question = People Correct (=24) Pretest	Comprehension Level Question = People Correct (= 24) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	11	12	.5	0
Control Group	11	11	.673	0

Table 40

Summary of results for requirements management question #25

Assessment	Application Level Question = People Correct (=25) Pretest	Application Level Question = People Correct (=25) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	10	11	.395	0
Control Group	10	11	.395	0

Table 41

Summary of results for requirements management question #26

Assessment	Application Level Question = People Correct (=26) Pretest	Application Level Question = People Correct (=26) Posttest	p-value (94.5% confidence interval)	Estimated Median
Experimental Group	11	11	Not applicable. since all responses were correct	Not applicable. since all responses were correct
Control Group	8	11	.091	0

The difference in the pilot case study population and the case study population is again evident. While the course project did include aspects of change management, even this was not a factor in the results as the students started the course with a high level of understanding. While the students are representative of their university, they are not representative of the general population of students studying Software Engineering.

5.9 Summary

This chapter presents the design and execution of the simulator assessment. The details of the experimental method utilized, along with additional methodological considerations, are described for both a pilot case study and the experimental case study. In addition, the results of both case studies are presented. In the pilot case study, the students' overall understanding of requirements engineering increased in the level of Bloom's Taxonomy. The most notable areas of improvement are in requirements elicitation and requirements management, as both areas had improvement that is statistically significant. Improvement in the experimental case study was more illusive as the students started with such a high level of understanding. Some areas of improvement by the experimental group are Requirements Analysis (Scope) and Requirements Elicitation (Facilitated Meeting). The next chapter presents the future work for this research, including those alluded to in the requirements validation (section 5.5.3) and requirements management sections (section 5.5.4).

Chapter 6. Conclusions and Future Work

6.1 Conclusions

This research endeavor arose from the need to enhance the educational experience for undergraduate Software Engineering students in order to prepare them for industry. The research objective is to define and evaluate a model that provides undergraduate Software Engineering students with the knowledge and skills normally acquired while working on a large project. The development of a model, based on System Dynamics Modeling, and a large project simulator has been tested for use in the undergraduate Software Engineering classroom. In conjunction with the model, a hypothesis was tested whereby students who use the simulator can better understand the Requirements Analysis and Specification activities in a long-term project than students who did not use the simulator. The results of this research has contributed to the field of Software Engineering and Computer Science Education in the three ways outlined in Chapter 1.

First, the effectiveness of the model and simulator (the hypothesis) was tested in the form of the two case studies. The pilot study showed a general success of the hypothesis, showing a potential for increased student learning of Requirements Analysis and Specification activities in a long-term project. This recognition was demonstrated by the increase in the level of understanding in various requirements engineering topics using Bloom's Taxonomy as the measure of understanding. Nearly all four areas directly addressed in the simulation showed improvement. The areas of Requirements Elicitation and Requirement Management showed the most improvement in terms of student movement up Bloom's Taxonomy. Improvement in these areas is statistically significant. Students also showed improvement in their level of understanding for Requirements

Analysis, though the improvement was between the Knowledge and Comprehension levels rather than the Comprehension and Application levels. Improvement in these areas was not statistically significant however

Success was not across all topics, as student understanding of Requirement Validation was nearly unchanged resulting in no statistically significant improvement. Possible reasons for the lack of success range from learning style differences to motivation. Students have different learning styles. As a result, the Requirement Validation portion of the simulator did not address a wide variety of learning styles. Students who may need auditory reinforcement (or other perception-related issues affecting learning) or have emotional or sociological needs that affect learning that the simulator does not address. The anonymous student volunteers could not have their learning styles assessed by the researcher and the students themselves may not be sufficiently aware of their own learning styles to share the information in the form of a survey. Also, some students may simply not have put effort into all topics. While proof of completion (in terms of the reports) was requested, the amount of time or effort allocated to the case study participation was not measured. The application and analysis of concepts often requires the student to take notes and refer to them throughout an activity. If care was not taken to write or refer to notes, then the activity would be more difficult to accomplish. Students were not asked to submit their notes in order to simplify case study participation. Regardless, further analysis is needed in order to address the learning style issues – see Section 6.2 for further details. Although student understanding did not increase at some of the higher levels of Bloom's Taxonomy, increased understanding was measured at the lower levels of understanding for those topics. Thus

while issues in learning styles (or otherwise) impacted higher-level learning more so than the lower levels of understanding (e.g. Knowledge and Comprehension). As the pilot case study did not utilize control group, a second case study was conducted in order to fully test the hypothesis with both an experimental group and a control group.

The additional case study revealed how two highly skilled populations compare, with one population utilizing the simulation and the other population not having utilized the simulation. Unlike the pilot case study, the areas where the simulator had an impact were few. Specifically the areas of impact consisted of Requirements Elicitation (Facilitated Meeting), Requirements Analysis (Scope), and Requirements Validation. The simulation had an impact at the Application level of Requirements Validation and the Application level of Requirements Analysis (Scope). Some potential for impact does exist. The control group did not improve in these areas. Generally, both the experimental group and the control group made some improvement in all topics. In some areas, such as Requirements Validation, the scores for the pretest started out so high that there was not much room for improvement. Such a high skill, though impressive, made data analysis and conclusions difficult to conduct. The simulator seems to be less useful for students who are advanced or enrolled in a Requirements Engineering course.

Second the Requirements phase of a project was modeled using System Dynamics Modeling. Such a model, though simple, enabled a different area of development to be modeled and provided a foundation for the simulator (as outlined in Chapter 4). The Abdel-Hamid model has been modified over the years (Rubin, Johnson & Yourdon, 1994; Tvedt, 1996) but the model the additional revisions do not address the requirements phase. The primary modeling focus has concerned Project Management (Collofello,

2000; Rus, Collofello & Lakey, 1998). One of the few undertakings in modeling the requirements engineering phase, Joint Application Development (JAD) process model, models the impact of the social interaction with the project's quality and schedule (Christie & Staley, 2000). The model generated in this research provides an overview of the activities involved in the general requirements engineering phase. Such a general model can serve as a foundation for further elaboration and specification, including the use of specific process models. Further work is discussed in Section 6.2.

Third, the project simulator itself is a unique product to be used in the classroom or in training environments. While an apparent by-product, as the simulator is not an abstract theory or algorithm, it is a unique research outcome that can be used in the classroom. Students increase their understanding, even if at a basic level, of Requirements Elicitation and Requirements Validation that are not covered adequately in the course. The role of Requirements Analysis is put into context for students, who only classify requirements in a limited fashion during the project and exams. In addition, students learn more about the need for and activities in Requirements Management since it is not conducted in the course. The details of the simulator requirements and flow of interaction are presented in detail in Chapters 3 and 4. The highly interactive simulation engages the students with the various activities involved in the requirements phase and the subsequent phases of development in the context of a large project. Students participate in the project, not by inputting values and reading graphs, but by making selections and responding to quantitative and qualitative feedback from developers and stakeholders. Students are immersed in the project rather than participating on the sidelines.

These contributions provide a foundation for the use of simulation in the undergraduate, Software Engineering classroom. The potential exists for increased student understanding in an area of Software Engineering that is often undervalued. The simulator, and the underlying model, are tools to assist in this endeavor. Instructors can utilize the simulator to supplement course lecture and the class project, and researchers can build upon the model to further requirements engineering research.

6.2 Future Work

With these contributions, further study into the use of simulation in the undergraduate, Software Engineering classroom exists. The fourteen pilot case study participants enabled a successful study that tested the concept of a simulation in the Introductory Software Engineering classroom. The results were promising, and were worth further study. The additional case study, consisting of 24 students, provided the opportunity to truly ascertain whether the simulation had an impact on understanding. However the population in the second case study scored well in both groups. Some areas had impact, though a small one. Still in order to generalize the results and expand the application of the simulation to the greater population of undergraduate Software Engineering students. To facilitate the ability to generalize the results to the general undergraduate student population enrolled in the Introductory Software Engineering course (or a Requirements Specification course), further study could be designed to include several course sections, including courses at among several universities. This time-consuming task, due to the coordination required, could not be undertaken in this current endeavor. However the researcher intends to do so as the next step in the

research. Expanding the number of courses addresses the overall issue of the difficulty in participant recruitment experienced in the pilot case study and the skill imbalance would be better distributed than was evident in the experimental case study population (further described in Chapter 5).

A larger participant group would also enable different experimental approaches to be conducted that were beyond the scope of this research. Given that the simulator increases student understanding, further research would allow comparison between the benefits of simulation and other techniques (e.g. case studies). The identification and required validation of a comparable technique required resources that detracted from the primary research contribution. The undertaking of such an expanded study, the comparison of the simulator with a (validated) case study, can be accomplished through the use of a Pretest-Posttest Control-Group Design (Gall, Borg, & Gall, 1996), the same technique used to conduct the main (experimental) case study outlined in Chapter 5. A large pool of participants would be randomly divided into two groups to compare the impact of the simulator and the case study. The control group would use the case study, and the experimental group would use the simulator. Such a study would require careful planning as extra instruction by the course or lab instructor in order to ensure that each group utilizes the relevant tool without sharing the knowledge with the other group. The undertaking is beyond the scope of this initial research, but it is an appropriate follow-up study.

In addition, a longitudinal study would enable researchers to examine the long-term effects of simulator use. By tracking both students who have and have not utilized the simulator, the meaningful feedback would provide researchers valuable insight as to

the simulator's effectiveness on students' long-term habits. After all, the purpose of introductory Software Engineering instruction is to prepare future Software Engineers for industry.

Another focus of revision is to enhance areas of the simulator where topic understanding fell short of the target learning levels. The Requirements Validation and area is a topic area that should be examined further. While the stakeholder concept was better understood, achievement did not reach the higher levels of understanding for many students. Careful attention to different learning style may address this issue. The inclusion of electronic notetaking tools or of a real-time collaborate environment may address the different learning styles and motivation issues. In particular the Requirement Validation topic is complex enough that it could be its own simulator. The process of validating requirements and the timing of validation are examples of Requirements Validation topics that need to be modeled and simulated at a lower level of granularity.

Both the model and the simulation can be expanded to include other topics. For example, the Observation elicitation technique can be added. To better address the nuances involved in observation, video clips can be included to provide a more meaningful interaction than the sole use of text that is sufficient for the Interview and Facilitated Meeting techniques. Also, other knowledge areas such as Design, Testing, and Project Management can be added to provide more breadth to the experience. In addition, the model itself can be enhanced to include behavior that more accurately reflects industry. Partnership with local industry can help develop a more realistic representation. Such an overhaul would need to be skewed to reflect specific techniques or domains, such as extreme programming or telecommunications respectively. These

revisions are well beyond the scope of this research, but the attempt is well worth the effort.

In order to accomplish these goals, new tools need to be found that can support the interaction between the model and the interface. While the current interface tool (Director) supports multimedia, the data exchange technology (Microsoft's Dynamic Data Exchange) used to share the data between the interface (Interaction Layer) and the model is crude and obsolete.

The research contributed to the field of Computer Science Education and Software Engineering through the development and instructional application of the model and simulator. Students can use the simulator to increase their level of understanding of the Requirements Engineering activities required in a long-term project. The simulator's utilization supplements the topics presented in lecture and applied in the course project. While the current version demonstrates the usefulness of the concept, the model's extension can allow it to grow in new directions. Further experimentation will reinforce the benefits of using simulators as instructional tools in the Software Engineering curriculum.

REFERENCES

- Abdel-Hamid, T. (1993, May). Thinking in circles. *American Programmer*, 3-9.
- Abdel-Hamid, T., & Madnick, S. (1991). *Software project dynamics: An integrated approach*. Englewood Cliffs, NJ: Prentice Hall.
- Abdel-Hamid, T., Sengupta, K., & Hardebeck, M. (1994, May). The effect of reward structures on allocating shared staff resources among interdependent software projects: An experimental investigation. *IEEE Transactions on Engineering Management*, 41(2), 115-125.
- Abran, A., & Moore, J. (Eds.) (2001, May). *Guide to the software engineering body of knowledge. a stone man trial version 1.00*. Retrived May, 2, 2002, from Guide to the SWEBOK Web site: <http://www.swebok.org>
- Abran, A., & Moore, J. (Eds.) (2000). *Guide to the software engineering body of knowledge. a stone man version (version 6)*. Retrieved November, 2000, from Guide to the SWEBOK Web site: <http://www.swebok.org> ironman Guide to SWEBOK
- Ambler, S. (1999). *Process patterns*. Cambridge, UK: Cambridge University Press.
- Basilis, S., & Boehm, B. (2001, January). CeBASE software defect reduction top 10-list. *IEEE Computer*, 34(1), 135-137.
- Beagley, S. (1994, March). Staying the course with the project control panel. *American Programmer*, 29-34.
- Bourque, P., & Dupuis, R. (1999, November December) The Guide to the software engineering body of knowledge. *IEEE Software*, 16(6), 35-44.

- Carneson, J., Delpierre, G., & Masters, K. (2001). *Designing and managing multiple choice quizzes*. Retrieved February 2001, from University of Cape Town, South Africa Web site: <http://www.uct.ac.za/projects/cbe/mcqman/mcqappc.html>
- Chichakly, K. (1993, May). The bifocal vantage point: Managing software projects from a systems thinking perspective. *American Programmer*, 18-25.
- Christie, A. (1999, April). *Simulation - an enabling technology in software engineering*. Retrieved November 2000, from Software Engineering Institute's Web site: <http://www.sei.cmu.edu/publications/articles/christie-apr1999/christie-apr1999.html>
- Christie, A. (1999). Simulation in support of CMM-based process improvement. *The Journal of Systems and Software*, 46, 107-112.
- Christie, A., & Staley, M. (2000). Organizational and social simulation of a software requirements development process. *Software Process Improvement and Practice*, 5, 103-110.
- Collofello, J. (2000, March). University Industry collaboration in developing a simulation based software project management course. *Proceedings of the CSEE&T 2000 13th Conference on Software Engineering Education and Training*, USA. Retrieved November, 2000, from <http://dlib.computer.org/conferen/cseet/0421/pdf/04210161.pdf>
- Counseling Services (n.d.). *Learning skills program*. Retrieved January 2001, from University of Victoria Web site: <http://www.coun.uvic.ca/learn/program/hndouts/bloom.html>
- Doll, S. (2001, March 13). *Seven steps for avoiding scope creep*. Retrieved from builder.com: <http://builder.com/article.jhtml?id=u00820010313gcn01.htm>.

- Dupuis, R., Bourque, P., Abran, A., Moore, W., & Tripp L. (1999, November). *The SWEBOK project. Guide to the software engineering body of knowledge*. Retrieved November 2000, from Guide to the Software Engineering Body of Knowledge Web site: <http://www.swebok.org/documents>
- Gall, M., Borg, W., & Gall, J. (1996). *Educational research: An introduction*. White Plains, NY: Longman Publishers.
- Gause, D., & Weinberg, G. (1999). *Exploring requirements: Quality before design*. New York, NY: Dorset House Publishing.
- Gilbert, N., & Troitzsch, K. (1998). *Simulation for the social scientist*. Retrieved May 2002, from Universitat Koblenz-Landau Web site: <http://www.uni-koblenz-landau.de/~kgt/Learn/Textbook/NewBook.html>
- Gleitman, H. (2001). *Lecturing: using a much maligned method of teaching*. Retrieved January 5, 2003, from the University of Chicago Web site: <http://teaching.chicago.edu/handbook/tac06.html>
- Hollander, W. (1999). *Nonparametric statistical methods*. New York, NY: Wiley.
- IEEE Computer Society. (n.d.). *Guide to the SWEBOK web site*. Retrieved November 2001, from Guide to the Software Engineering Body of Knowledge Web site: <http://www.swebok.org/overview>
- Joyce, B., Wel, M., & Showers, B. (1992). *Models of teaching*. Boston, MA: Allyn and Bacon.
- Kellnet, M., Madachy, R., & Raffo, D. (1999). Software process simulation modeling: Why? what? how? *The Journal of Systems and Software*, 46, 91-105.

- Krumme, G. (1995). *Major Categories in the Taxonomy of Educational Objectives*. - Bloom 1956. Retrieved June 2001, from University of Washington. Seattle Web site: <http://faculty.washington.edu/krumme/guides/bloom.html>
- Leffingwell, D. (1996). *Calculating your return on investment from more effective requirements management*. Retrieved December 2002 from Rational Web site: <http://www.rational.com/media/whitepapers/roi.pdf>
- Leffingwell, D., & Widrig, D. (2000). *Managing software requirements*. Reading, MA: Addison-Wesley.
- Madachy, R. (1996). *Process modeling with system dynamics*. Retrieved November 2000, from University of Southern California Web site: <http://www-ref.usc.edu/~madachy/sd/sepg.htm>
- McConnell, S. (1998). *Software project survival guide*. Redmond, WA: Microsoft Press.
- Merril, D., & Collofello, J. (1997, November). Improving software project management skills using a software project simulator. *Proceedings of the Frontiers In Education Conference*, USA, 1361-1366.
- New York Institute of Technology (n.d.). *Teaching strategies: Constructivist learning*. Retrieved May 2002, from New York Institute of Technology Educational Enterprise Zone Web site: <http://www.nyiteez.org/EDIN777/strategies.htm>
- Paananen, J. (1995). *Introduction to and comparison of formalisms*. Retrieved November 2000, from Helsinki University of Technology Web site: <http://www.tml.hut.fi/Opinnot/Tik-110.501/1995/intfo.html#HEADING2>
- Pfahl D. & Lebsanft, K. (2000). *Using simulation to analyse the impact of software requirement volatility on project performance*. Retrieved November 2000, from

European Software Control and Metrics Web site:

<http://www.escom.co.uk/conference2000/pfahl-lebsanft.pdf>

Roland, L. (1997, April). *Benefits of collaborative learning*. Retrieved December 2002

from http://www.wou.edu/las/natsci_math/math/class/couplist.htm

Rubin, H., Johnson, M., & Yourdon, E. (1994, September). With the SEI as my copilot:

Using software process "flight simulation" to predict the impact of improvements in process maturity. *American Programmer*, 50-57.

Rus, I. (1997). Modeling the impact on project cost and schedule of software engineering

practices for achieving and assessing software quality factors. Retrieved November 2000, from Arizona State University's College of Engineering Web site:

<http://www.eas.asu.edu/~sdm/papers.html>

Rus, I., Collofello, J., & Lakey, P. (1998, June). Software process simulation for

reliability strategy assessment. International Workshop on Software Process Simulation Modeling, USA. Retrieved November 2000, from <http://enuxsa.eas.asu.edu/~rus>

Sengupta, K., & Abdel-Hamid, T. (1993, April). Alternative conceptions of feedback in

dynamic decision environments: An experimental investigation. *Management Science*, 39(4), 411-428.

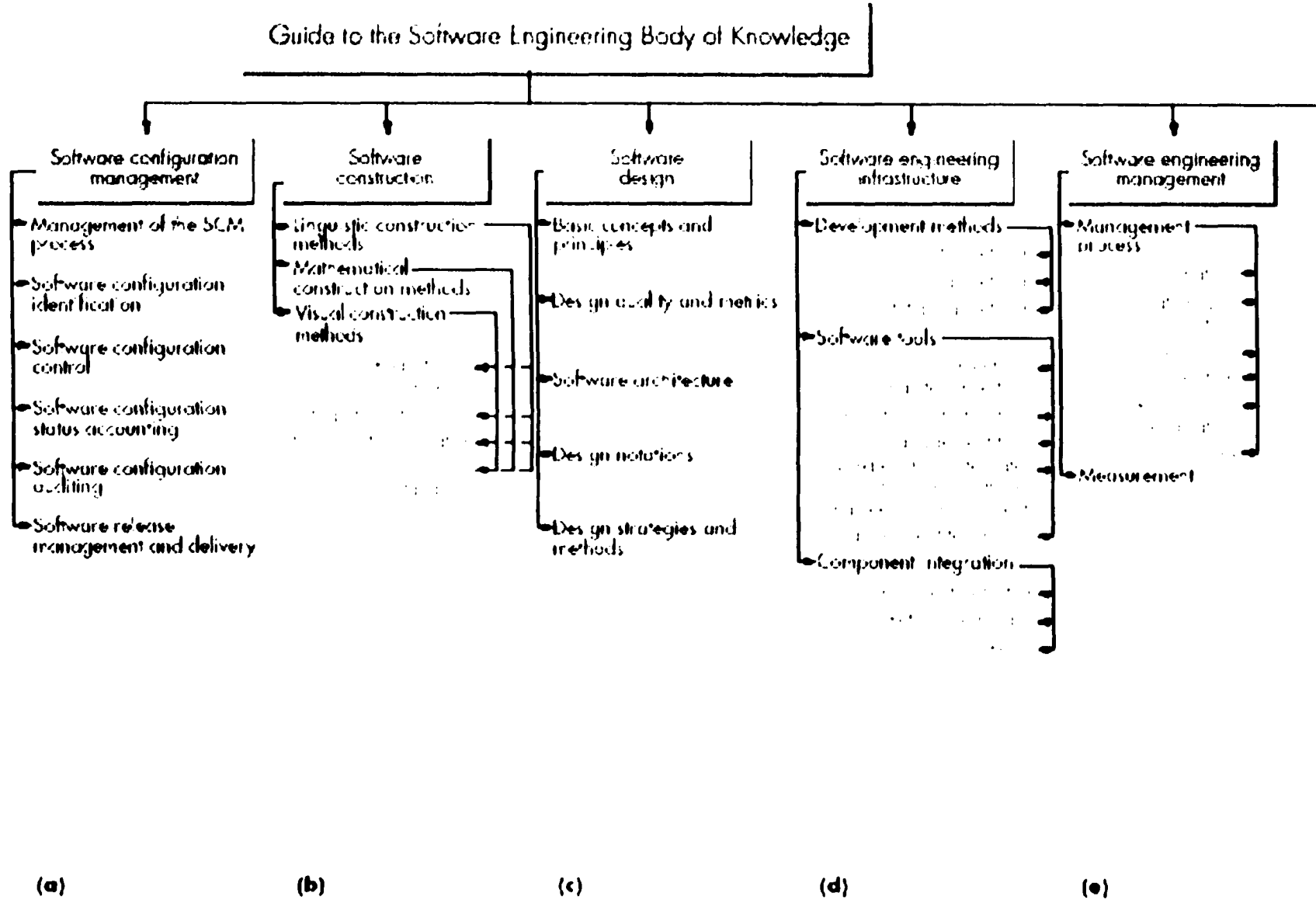
Sengupta, K., Abdel-Hamid, T., & Bosley, M. (1999, January). Coping with staffing

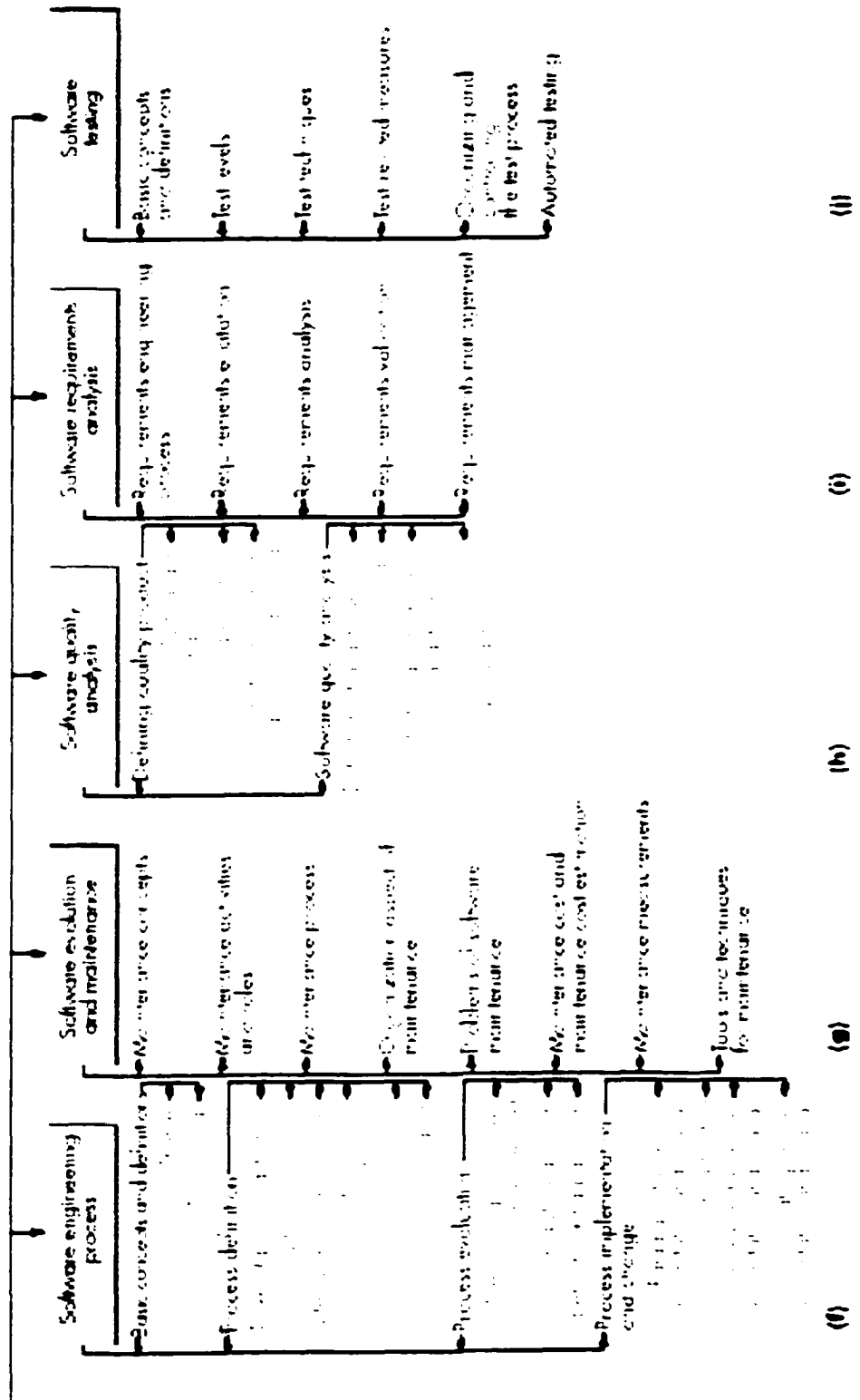
delays in software project management: An experimental investigation. *IEEE Transactions on Systems, Man, and Cybernetics- Part A Systems and Humans*, 29(1), 77-91.

- Smith, B., Nguyen, N., & Vidale, R. (1993, May). Death of a software manager: How to avoid career suicide through dynamic software process modeling. *American Programmer*, 10-17.
- Soukup, J. (1994). *Taming C++: Pattern classes and persistence for large projects*. Reading, MA: Addison-Wesley.
- Sommerville, I. (2001). *Software engineering*. New York, NY: Addison-Wesley.
- Sterman, J. (1992). *System dynamics modeling for project management*. Retrieved April 2002, from Massachusetts Institute of Technology Web site: <http://web.mit.edu/jsterman/www/SDG/project.pdf>
- Sycamore, D. (1996). *Improving software project management through system dynamics modeling*. Unpublished master's thesis. Arizona State University.
- Tvedt, J. (1996). *An extensible model for evaluating the impact of process improvements on software development cycle time*. Unpublished doctoral dissertation. Arizona State University.

APPENDIX A

SWEBOK KNOWLEDGE AREA MAPPINGS (Bourque & Dupuis, 1999)





(a) (b) (c) (d) (e)

APPENDIX B

SWEBOK TOPICS ORGANIZED BY KNOWLEDGE AREA.
AND CLASSIFIED ACCORDING TO BLOOM'S TAXONOMY

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
Software Configuration Management (SCM)		
I. Management of the SCM Process	Knowledge	Knowledge
A. Organizational Context for SCM	Knowledge	Knowledge
B. Constraints and Guidance for SCM	Knowledge	Knowledge
C. Planning for SCM	Knowledge	Knowledge
1. SCM Organization & Responsibilities	Knowledge	Knowledge
2. SCM Resources & Schedules	Comprehension	Knowledge
3. Tool Selection & Implementation	Knowledge	Knowledge
4. Vendor Subcontractor Control	Knowledge	NA
5. Interface Control	Comprehension	? NA
D. SCM Plan	Knowledge	Knowledge
E. Surveillance of SCM	Comprehension	NA
1. SCM Metrics & Measurements	Comprehension	NA
2. In-Process Audits of SCM	Knowledge	NA
II. Software Configuration Identification	Comprehension	Comprehension
A. Identifying items to be controlled	Comprehension	Comprehension
1. Software Configuration	Comprehension	Comprehension
2. Software Config. Item (SCI)	Comprehension	Comprehension
3. Software Config. Item relationships	Comprehension	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
4. Software Versions	Comprehension	Comprehension
5. Baselines	Comprehension	Comprehension
6. Acquiring SCIs	Knowledge	? Knowledge
B. SCM Library	Comprehension	Knowledge
III. Software Configuration Control	Application	Application
A. Requesting, Evaluating & Approving Software	Application	Comprehension
1. Software Configuration Control Board	Application	Comprehension
2. Software Change Request Process	Application	Application
B. Implementing Software Changes	Application	Application
C. Deviations & Waivers	Comprehension	NA
IV SW Configuration Status Accounting	Comprehension	Knowledge
A. SW Configuration Status Information	Comprehension	Knowledge
B. SW Configuration Status Reporting	Comprehension	Knowledge
V. Software Configuration Auditing	Knowledge	Knowledge
A. Software Functional Config. Audit	Knowledge	? Knowledge
B. Software Physical Config. Audit	Knowledge	? Knowledge
C. In-process audits of a software baseline	Knowledge	? Knowledge
VI. Software Release Management &	Comprehension	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
Delivery		
A. Software Building	Comprehension	Comprehension
B. Software Release Management	Comprehension	Knowledge
Software Construction (Prior to the course in reality)		
I. Linguistic Construction Methods	Not in ver. 0.6	Synthesis
A. Reduction in Complexity	Not in ver. 0.6	Synthesis
B. Anticipation of Diversity	Not in ver. 0.6	Synthesis
C. Structuring for Validation	Not in ver. 0.6	Synthesis
D. Use of External Standards	Not in ver. 0.6	Synthesis
II. Mathematical Construction Methods	Not in ver. 0.6	Analysis
A. Reduction in Complexity	Not in ver. 0.6	Analysis
B. Anticipation of Diversity	Not in ver. 0.6	Analysis
C. Structuring for Validation	Not in ver. 0.6	Analysis
D. Use of External Standards	Not in ver. 0.6	Analysis
III. Visual Construction Methods	Not in ver. 0.6	Knowledge
A. Reduction in Complexity	Not in ver. 0.6	Knowledge
B. Anticipation of Diversity	Not in ver. 0.6	Knowledge
C. Structuring for Validation	Not in ver. 0.6	Knowledge
D. Use of External Standards	Not in ver. 0.6	Knowledge
Software Design		

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
I. Software Design Basic Concepts		
A. General Design Concepts	Comprehension	Comprehension
B. The Context of Software Design	Comprehension	Comprehension
C. The Software Design Process	Analysis, Evaluation	Analysis, Evaluation
D. Basic Software Design Concepts	Analysis	Analysis
E. Key Issues in Software Design	Comprehension, Application	Comprehension, Application
II. Software Architecture		
A. Architectural Structures & Viewpoints	Application	Comprehension
B. Architectural Styles & Patterns (Macro-Arch.)	Analysis, Evaluation	Synthesis
C. Design Patterns (Micro-Arch.)	Analysis, Evaluation	Knowledge
D. Design of Families of Programs & Frameworks	Application	NA
III. Software Design Quality Analysis & Evaluation		
A. Quality Attributes	Analysis	Comprehension
B. Quality Analysis & Evaluation Tools	Application,	Application

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
	Analysis	
C. Metrics	Application, Analysis	Knowledge
IV. Software Design Notations		
A. Structural Descriptions (static view)	Application, Analysis	Application, Analysis
B. Behavioral Descriptions (dynamic view)	Application, Analysis	Application
V. Software Design Strategies & Methods		
A. General Strategies	Application	Application
B. Function-oriented Design	Application	NA
C. Object-oriented Design	Analysis, Evaluation	Application, Analysis
D. Data-structure-centered Design	Comprehension	NA
E. Other Methods	Comprehension, Application	NA
VI. Software Design Tools		
A. Mathematical Tools	Application	Knowledge
B. CASE Tools	Application	Knowledge
C. Tools for Metrics	Application	Knowledge
VII. Standards relevant to Software Design	Comprehension	Application

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
Software Engineering Infrastructure		
I. Development Methods		
A. Heuristic Methods		
1. Structured Methods	Application	Application
2. Data-oriented Methods	Application	Application
3. Object-oriented Methods	Application	Application
4. Domain-specific Methods	Knowledge	Knowledge
B. Formal Methods		
1. Specification Languages	Comprehension	NA
2. Refinement	Knowledge	NA
3. Verification Proving Properties	Comprehension	NA
C. Prototyping Methods		
1. Styles	Comprehension	Knowledge
2. Prototyping targets	Comprehension	Knowledge
3. Evaluation Techniques	Comprehension	Knowledge
II. Software Tools		
A. Development & Maintenance Tools		
1. Creation & Editing	Application	Application
2. Translation Tools	Application	Knowledge
3. Analysis Tools	Application	Knowledge
4. Comprehension Tools	Application	Knowledge

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
5. Testing Tools	Application	Knowledge
6. Integrated CASE tools and SE Environments	Application	Knowledge
7. Reverse & re-engineering Tools	Comprehension	Knowledge
B. Management Tools		*= not CASE
1. Project planning & tracking tools	Application	Application *
2. Risk analysis & management tools	Knowledge	NA
3. Measurement tools	Application	Knowledge
4. Defect, Enhancement, Issue	Application	Application *
5. Configuration management tools	Application	Application *
C. Infrastructure Support tools		
1. Interpersonal Communication	Application	Application
2. Information retrieval	Application	Knowledge
3. System administration & support tools	Application	NA
4. Tool integration techniques	Knowledge	NA
5. Meta-tools	Comprehension	NA
III. Component Integration		
A. Component Definition		
1. Interface specifications	Knowledge	Knowledge

Knowledge Area (K.A)	SWEBOK Bloom Level	Course's Bloom Level
2. Protocol specifications	Knowledge	Knowledge
3. Off-the-shelf components	Application	Knowledge
B. Reference Models		
1. Open systems	Comprehension	Knowledge
2. Standard architectures	Comprehension	Knowledge
3. Frameworks	Application	NA
4. Patterns	Application	NA
C. Reuse		
1. Types of reuse	Comprehension	Comprehension
2. Re-engineering	Comprehension	NA
3. Reuse repositories	Comprehension	NA
4. Cost benefit analysis	Comprehension	NA
Software Engineering Management		
I. Archival activities	Application	Application
II. Acquisition Decisions & Management	?	NA
III. Collection of Data	Analysis	Analysis
IV. Collection & Negotiation of Requirements	Analysis	Analysis
V. Communication	Synthesis (Evaluation)	Synthesis
VI. Control Process	Evaluation	Evaluation

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
VII. Determine Deliverables	?	NA
VIII. Determining Closure	Application	NA
IX. Determining Satisfaction of Requirements	Analysis	Analysis
X. Determining the Goals of Measurement	Analysis	Comprehension
XI. Feasibility Analysis	Synthesis	Synthesis
XII. Feedback	Synthesis	Synthesis
XIII. Implementation of Plan	Synthesis	Synthesis
XIV. Implementing a metrics process	?	NA
XV. Iterative development	?	Knowledge
XVI. Maintenance	?	NA
XVII. Measuring software & its development	?	NA
XVIII. Monitor process	Analysis	Comprehension
XIX. Personnel management	Synthesis	Analysis
XX. Policy management	Synthesis	Comprehension
XXI. Portfolio management	Analysis	NA
XXII. Process for the revision of requirements	Analysis	Analysis
XXIII. Process planning	?	Application
XXIV. Proposal construction	?	Synthesis

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
XXV. Quality management	Evaluation	Application
XXVI. Resource allocation	Application	Application
XXVII. Reviewing & evaluating performance	Synthesis	Synthesis
XXVIII. Risk management	Synthesis	Application
XXIX. Schedule & cost estimation	Evaluation	Application
XXX. Selection of measurements	Analysis	Analysis
XXXI. Software metric models	Analysis	Comprehension
XXXII. System retirement	?	NA
XXXIII. Task & responsibility allocation	Analysis	Application
Software Engineering Process		
I. Basic Concepts & Definitions		
A. Themes	Not in ver. 0.6	Comprehension
B. Terminology	Not in ver. 0.6	Comprehension
II. Process Infrastructure	Not in ver. 0.6	Comprehension
III. Process Measurement		
A. Methodology in process measurement	Not in ver. 0.6	Comprehension
B. Process measurement paradigms	Not in ver. 0.6	Comprehension
IV. Process Definition		
A. Types of Process Definitions	Not in ver. 0.6	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
B. Life Cycle Models	Not in ver. 0.6	Comprehension
C. Software Life Cycle Process Models	Not in ver. 0.6	Comprehension
D. Notations for Process Definitions	Not in ver. 0.6	NA
E. Process Definition Methods	Not in ver. 0.6	NA
F. Automation	Not in ver. 0.6	NA
V. Qualitative Process Analysis	Not in ver. 0.6	Knowledge
VI. Process Implementation & Change		
A. Paradigms for process implementation & change	Not in ver. 0.6	Comprehension
B. Guidelines for process implementation & change	Not in ver. 0.6	Comprehension
C. Evaluating the outcome of process implementation & change	Not in ver. 0.6	Comprehension
Software Evolution & Maintenance		
I. Introduction to software evolution & maintenance	Comprehension	Comprehension
A. Need for evolution & maintenance	Comprehension	Comprehension
B. Categories of maintenance	Comprehension	NA
II. Evolution & Maintenance activities	Comprehension	Knowledge
A. Unique Activities	Comprehension	Knowledge
B. Supporting activities	Comprehension	Knowledge

Knowledge Area (K.A)	SWEBOK Bloom Level	Course's Bloom Level
1. Configuration Management	Comprehension	Knowledge
2. Quality	Comprehension	Knowledge
C. Evolution & Maintenance planning activity	Comprehension	Knowledge
III. Maintenance process	Synthesis	Comprehension
A. Standards	Comprehension	NA
B. Maintenance Process models	Synthesis	Comprehension
IV. Organization aspect of maintenance	Comprehension	NA
A. the maintainer	Comprehension	NA
B. outsourcing	Comprehension	Knowledge
C. Organizational structure	Comprehension	NA
V. Problems of software maintenance	Comprehension	Knowledge
A. Technical	Comprehension	NA
1. Limited Understanding	Comprehension	NA
2. Testing	Comprehension	NA
3. Impact Analysis	Comprehension	NA
4. Maintainability	Comprehension	Comprehension
B. Management	Comprehension	NA
1. Alignment with organization issues	Comprehension	NA
2. Staffing	Comprehension	NA

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
3. Process issues	Synthesis	NA
VI. Maintenance cost & Maintenance cost estimation	Comprehension	NA
A. Cost	Comprehension	NA
B. Cost Estimation	Comprehension	NA
C. Parametric models	Comprehension	NA
D. Experience	Comprehension	NA
VII. Maintenance Measurements	Synthesis	Comprehension
A. Establishing a Metrics program	Comprehension	Comprehension
B. Specific Measures	Synthesis	Comprehension
VIII. Tools & Techniques for maintenance	Synthesis	NA
A. Maintenance tools	Synthesis	Knowledge
B. Program Comprehension	Synthesis	Knowledge
C. Re-engineering	Synthesis	NA
D. Reverse Engineering	Synthesis	NA
E. Impact Analysis	Synthesis	NA
IX. Resources	Comprehension	NA
Software Quality Analysis		
I. Definition of Quality	Programmer: Synthesis: SQA VV:	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
	Synthesis: Project Manager: Analysis	
II. Definition of SQA, VV process	Programmer: App: SQA VV: Synthesis: Project Manager: Analysis	Application
III. Plans	Programmer: App: SQA VV: Synthesis: Project Manager: Analysis	Application
A. Activities & techniques		
1. Static - People intensive	Programmer: Eval: SQA VV: Evaluation: Project Manager: Analysis	Analysis
2.. Static - Analysis	Programmer: Eval: SQA VV:	Analysis

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
	Evaluation: Project Manager: Analysis	
3.. Dynamic	Programmer: Synth Eval: SQA VV: Eval: Project Manager: Analysis	Analysis
IV. Measurement		
A. Fundamentals	Programmer: Application : SQA VV: Eval: Project Manager: Analysis	Application
B. Metrics	Programmer: Application : SQA VV: Evaluation: Project Manager: Analysis	Comprehension
C. Techniques	Programmer:	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
	Application :	
	SQA VV: Eval:	
	Project Manager:	
	Analysis	
D. Defect Characterization	Programmer:	Comprehension, Application
	Application :	
	SQA VV: Eval:	
	Project Manager:	
	Analysis	
E. Additional Concerns	Programmer:	Comprehension
	Application :	
	SQA VV: Eval:	
	Project Manager:	
	Analysis	
Software Requirements Analysis		
I. Requirements Engineering Process		
A. Process models	Knowledge	Knowledge
B. Process actors	Knowledge	Knowledge
C. Process support	Knowledge	Knowledge
D. Process quality and improvement	Knowledge	NA
II. Requirements Elicitation		

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
A. Requirements Sources	Comprehension	Comprehension
B. Elicitation Techniques	Application	Application, but limited
III. Requirements Analysis		
A. Requirements classification	Comprehension	Comprehension, but limited
B. Conceptual modeling	Comprehension	Application
C. Architectural design & requirements allocation	Analysis	Application
D. Requirements negotiation	Analysis	NA
IV. Requirements Specification		
A. The requirements definition document	Application	Application
B. The software requirements specification (SRS)	Application	Application
C. Document Structure	Application	Application
D. Document Quality	Analysis	Analysis
V. Requirements Validation		
A. The conduct of requirements reviews	Analysis	Application, but limited
B. Prototyping	Application	
C. Model validation	Analysis	Comprehension
D. Acceptance tests	Application	Application
VI. Requirements Management		

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
A. Change management	Analysis	Comprehension
B. Requirements activities	Comprehension	Comprehension
C. Requirements tracing	Comprehension	Comprehension
Software Testing		
I. Testing Basic Concepts & Definitions		
A. Definitions of testing & related terminology	Analysis	Comprehension
B. Faults v. Failures	Analysis	Comprehension
C. Test selection criteria test adequacy criteria (or stopping rules)	Application	Application
D. Testing effectiveness Objectives for testing	Comprehension	Comprehension
E. Testing for defect removal	Comprehension	Comprehension
F. The oracle problem	Comprehension	? NA
G. Theoretical & practical limitations of testing	Comprehension	Comprehension
H. The problem of infeasible paths	Application	Comprehension
I. Software testability	Application	Comprehension, Application
J. Relationships of testing to other activities	Application	Application
II. Test Levels		

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
A. Unit Testing	Synthesis	Application
B. Integration Testing	Synthesis	Application
C. System Testing	Synthesis	Application
D. Acceptance qualification testing	Synthesis	Application
E. Installation testing	Application	Knowledge
F. Alpha & Beta testing	Application	Knowledge
G. Conformance testing functional testing correctness testing	Application	
H. Reliability achievement & evaluation by testing	Comprehension	Comprehension
I. Regression testing	Application	Application
J. Performance testing	Application	Comprehension
K. Stress testing	Application	Application
L. Back-to-back testing	Knowledge	? NA
M. Recovery testing	Comprehension	NA
N. Configuration testing	Comprehension	Comprehension
O. Usability testing	Application	NA or Comprehension
III. Test Techniques		
A. Equivalence partitioning	Application	Application
B. Boundary-value analysis	Application	Application
C. Decision table	Knowledge	? Knowledge or NA

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
D. Finite-state machine-based	Knowledge	? Knowledge or NA
E. Testing from formal specifications	Knowledge	NA
F. Reference models for code-based testing (flow graph, call graph)	Evaluation	Comprehension
G. Control flow-based criteria	Evaluation	Comprehension
H. Data flow-based criteria	Comprehension	Comprehension
I. Error guessing	Application	? NA
J. Mutation testing	Knowledge	NA
K. Operational profile	Comprehension	NA
L. SRET	Knowledge	?
M. Object-oriented testing	Comprehension	Comprehension
N. Component-based testing	Comprehension	Comprehension
O. GUI testing	Knowledge	Knowledge
P. Testing of concurrent programs	Knowledge	NA
Q. Protocol conformance testing	Knowledge	NA
R. Testing of distributed systems	Application	NA
S. Testing of real-time systems	Comprehension	NA
T. Testing of scientific software	Knowledge	NA
U. Functional & structural	Synthesis	Application
V. Coverage & operational Saturation effect	Knowledge	Knowledge

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
IV. Test-related measures		
A. Program measurements to aid in planning & designing testing	Synthesis	Comprehension
B. Types, classification & statistics of faults	Application	Comprehension
C. Remaining number of defects fault density	Application	Comprehension
D. Life test, reliability evaluation	Comprehension	NA
E. Reliability growth models	Knowledge	NA
F. Coverage thoroughness measures	Evaluation	Comprehension
G. Fault seeding	Knowledge	Knowledge
H. Mutation	Knowledge	NA
I. Comparison & relative effectiveness of different techniques	Comprehension	Comprehension
V. Managing the Test Process		
A. Attributes Egoless programming	Application	Comprehension ?
B. Test process	Synthesis	Synthesis
C. Test documentation	Synthesis	Application
D. Internal v. independent test team	Comprehension	Comprehension
E. Cost effort estimation & other process metrics	Application	Comprehension

Knowledge Area (KA)	SWEBOK Bloom Level	Course's Bloom Level
F. Test reuse	Application	Application
G. Test activities	Application	Application
VI. Test Tools		
A. Selecting Tools	Application	NA
B. Use of Automated Tools	Application	NA
C. Surveys of Existing Support Tools	Application	Knowledge

APPENDIX C
SIMULATOR SWEBOK TOPIC DEFINITIONS

The following descriptions are taken from SWEBOK (Abran & Moore, 2001). The order and names of the topics are identical to those used in SWEBOK.

Software Requirements Topic Definitions	Definition
II. Requirements Elicitation	The first stage in understanding the problem that the software is intended to solve. Stakeholders are identified and relationships are established between the development team and the customer.
B. Elicitation Techniques	The process of gathering requirements from stakeholders by getting the stakeholders to communicate their requirements.
1. Interviews	The traditional means of acquiring information through structured question and answer sessions between the developer and stakeholder.
3. Facilitated Meetings	A means of gathering requirements by enabling a group of people to bring insight to their requirements through brainstorming, discussion, and refinement. Careful facilitation is needed in order to keep the meetings productive.
III. Requirements Analysis	The process of analyzing requirements to detect and resolve conflicts between requirements, discover the bounds of the system (and how it interacts with its environment), and elaborate system requirements to

	software requirements.
A. Requirements classification	The classification of requirements based on a variety of attributes.
1. Functional & Nonfunctional	The classification as to whether a requirement is functional or nonfunctional.
4. Priority	The priority of a requirement represents how essential the requirement is to meeting the system's goals. Classification is often fixed on a fixed point scale. Balance is needed between priority and the cost of development and implementation.
5. Scope	Scope is the extent to which a requirement affects the system and system components. Some requirements affect more than one component.
6. Volatility	The stability of requirements during the lifecycle. Some estimate as to the likelihood that a requirement will change is useful in order to alert developers during design. The result can be a design that is more tolerant of change.
V. Requirements Validation	The process of inspecting the requirements document to make sure that it defines the system correctly.
A. The conduct of requirements reviews	The inspection or formal review of the requirement document(s). A group of reviewers looks for errors.

	mistaken assumptions, lack of clarity, and deviations from standard practice.
1. Group composition is appropriate	The group participants, often including at least one representative for the customer. Also, the group members often help in the creation of the checklists.
2. Use of guiding documents like checklists to guide review and to document findings	Such documents are used to ensure that all needed perspectives and issues are addressed and their results recorded during inspection.
3. Review process is at specified checkpoints and redone as appropriate	Reviews are conducted at the completion of the system requirements definition document, the software requirements specification document, the baseline specification for a new release, and at other needed checkpoints in the lifecycle.
VI. Requirements Management	An activity that spans the entire lifecycle. This area is primarily concerned about change management and the maintenance of the requirements so that they reflect the (soon-to-be) system.
A. Change management	The process of managing the revisions, addition, or deletions to the system.
1. Understanding the role	The understanding of the need to manage change

of Change Management throughout lifecycle	throughout the product lifecycle.
2. Have procedure in place	The activities, with strong links to configuration management, required to manage change in the system.
3. Analyze proposed changes	The analysis required to accept, deny, or defer change submissions. Multiple considerations are required such as the affect of the change on the resources and the existing requirements of the system.

APPENDIX D
SIMULATOR MODEL

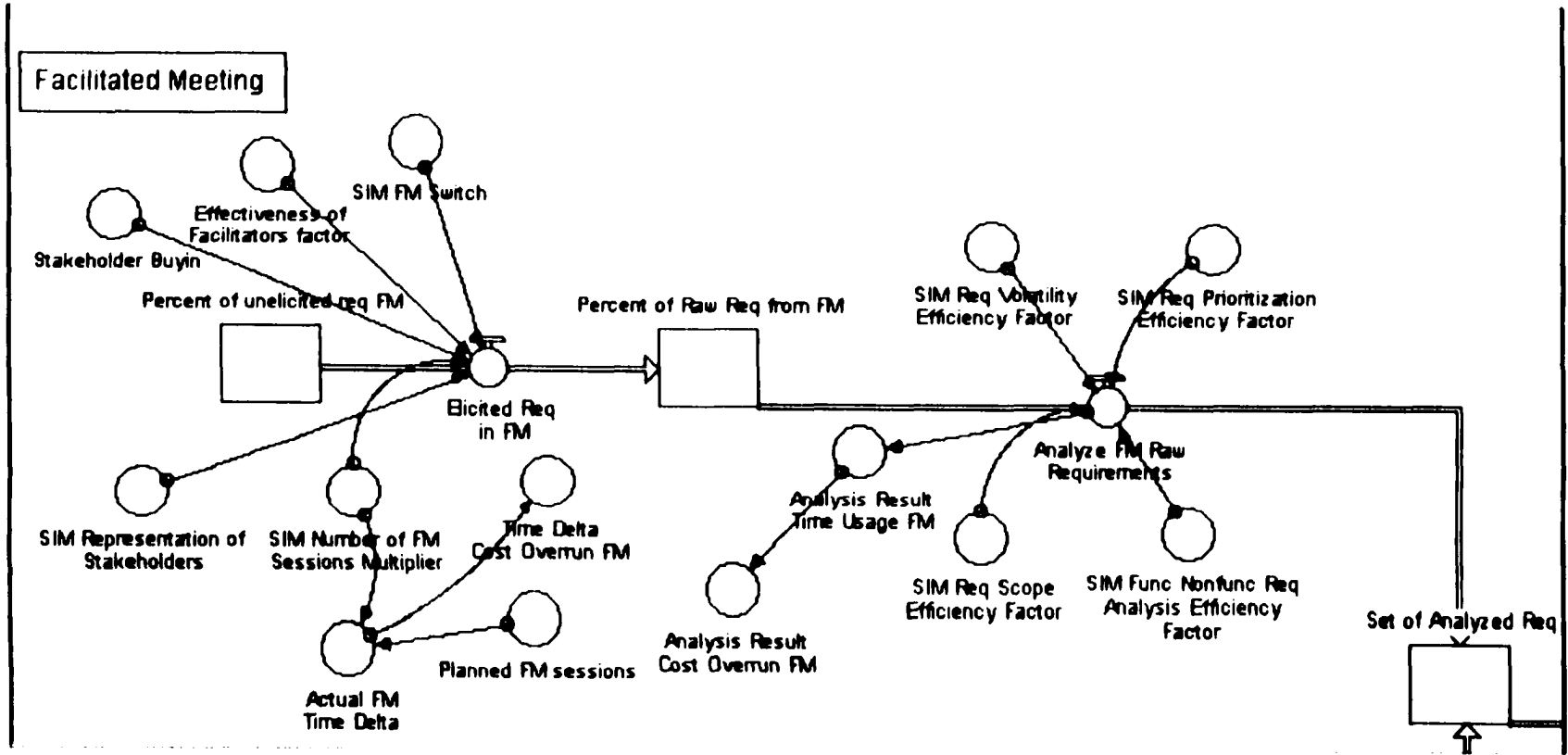


Figure D1. Facilitated Meeting and Requirements Analysis Segment of Model

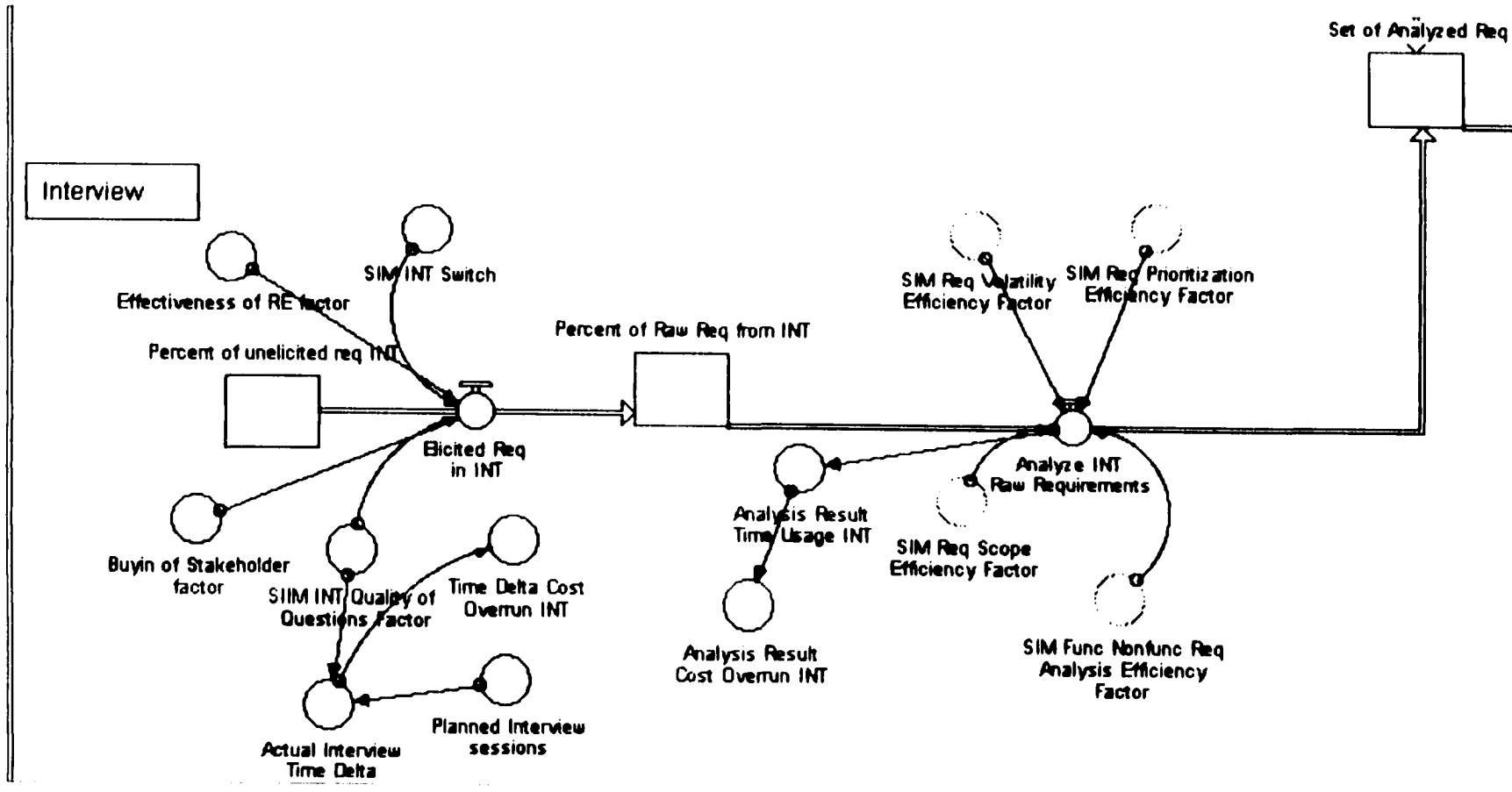


Figure D2. Interview and Requirements Analysis Segment of Model

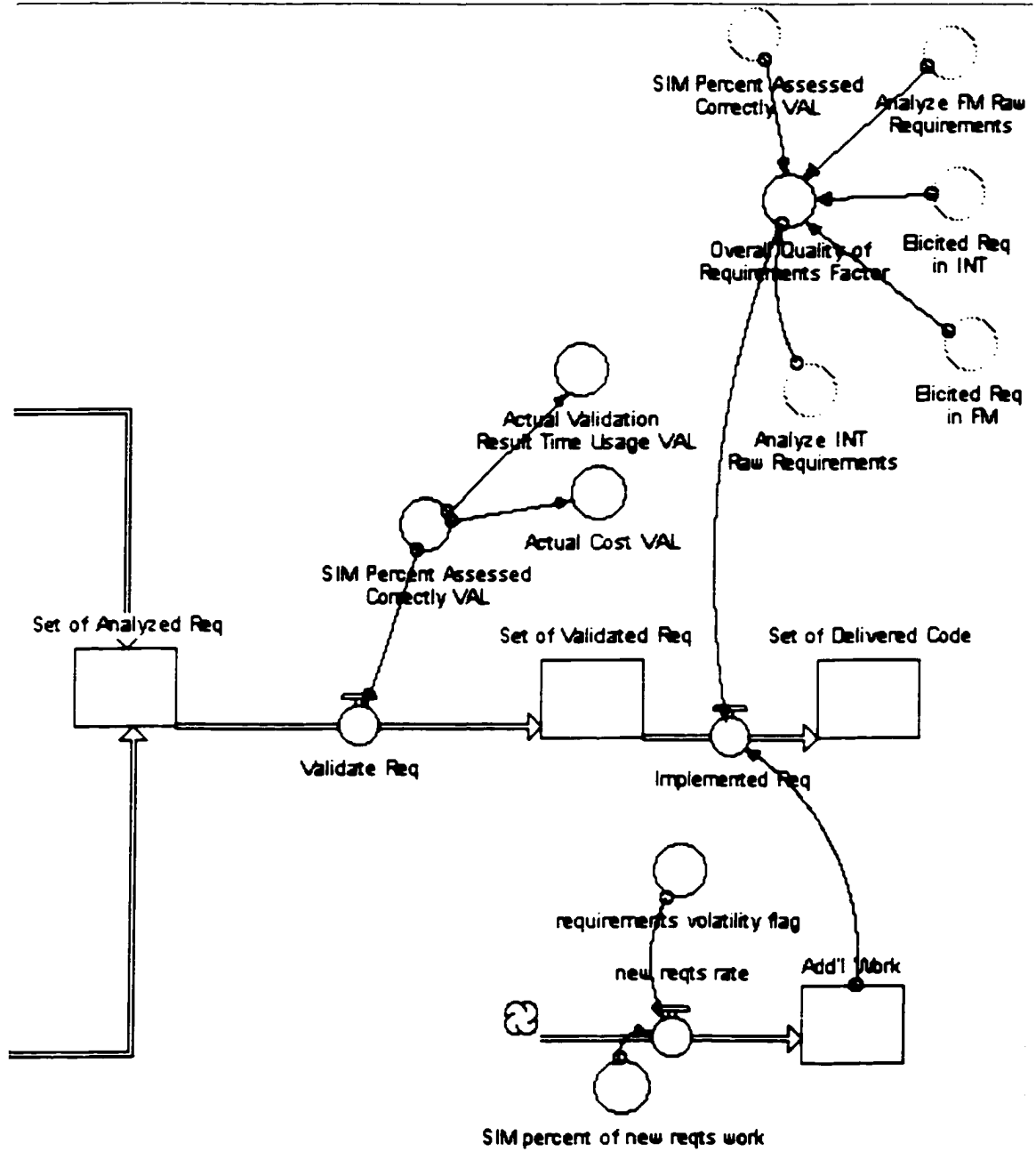


Figure D3. Requirement Validation and Implementation Phase of the Model

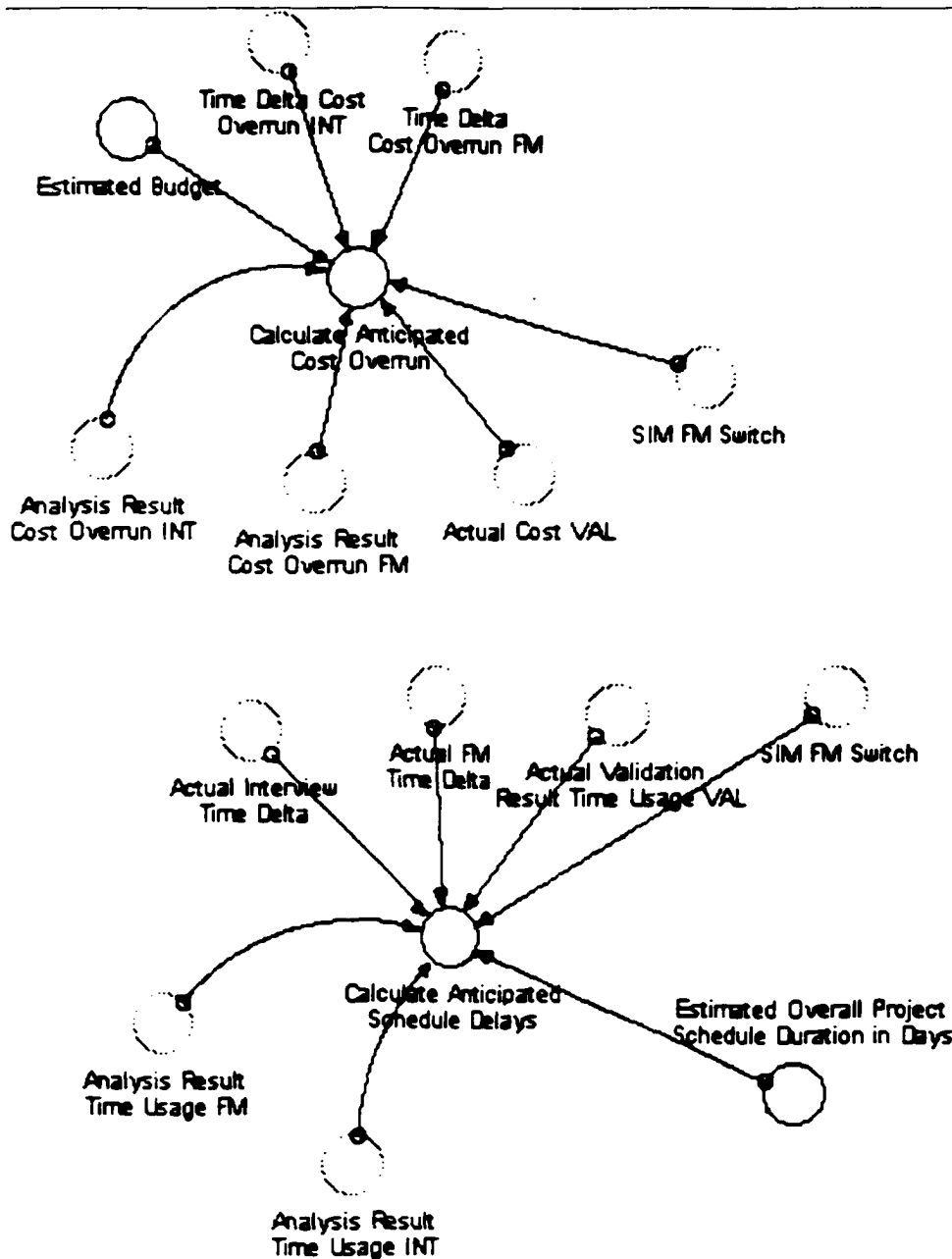


Figure D4. Overrun Calculation Segment of the Model

Table D1

Model Definitions

Model Item	Item Type	Definition
Facilitated Meeting		
Percent of unelicited req FM	Store	Percent of Requirements that have not been elicited yet. (starts at 100)
SIM FM Switch	Converter	Represents the use of the facilitated meeting technique (1=true, 0=false)
Effectiveness of Facilitators Factor	Converter	A factor (0-100) that represents the quality of the facilitator in terms of his her effectiveness during meetings. Effectiveness is in terms of asking questions that related to the topics on the agenda and in documenting the responses. The default will be 100
Stakeholder buyin	Converter	A factor (0-100) that represents the extent that the stakeholders agree with the project and its goals. The default will be 100 - that they totally agree with the project and its goals.
SIM Representation of Stakeholders	Converter	The factor (0 - 100) that the correct stakeholders were invited to the meetings.

		This is based on the selection made in the simulator (= of appropriate stakeholders selected / total stakeholders invited)
SIM Number of FM Sessions Multiplier	Converter	Number of facilitated meeting sessions actually conducted, as determined by the simulator
Time Delta Cost Overrun FM	Converter	The cost overrun incurred for the difference between the planned number of meetings and the number actually needed. The overrun is the product of the difference.
Actual FM Time Delta	Converter	Converting the number of any additional facilitated meetings (than those planned for) to calendar days by multiplying the number of meetings by 5 (days), due to a 5 days per workweek. If the number of sessions is less than or equal to the number planned for, then there is no time penalty.
Planned FM Sessions	Converter	The number of meeting sessions that were planned for.
Percent of Raw Req from FM	Store	The set, based on percentage, of elicited requirements from the facilitated meeting technique

Interviews		
Percent of unelicited req INT	Store	Percent of Requirements that have not been elicited yet. (starts at 100)
SIM INT Switch	Converter	Represents the use of the interview technique (1=true, 0=false)
Effectiveness of RE factor	Converter	A factor (0-100) representing how effective the RE is in conducting the interview - this is in terms of getting through the question set and documenting the responses. The default is 100.
Buyin of Stakeholder factor	Converter	A factor (0-100) representing the buyin of the stakeholder in the project and its goals. The default is 100 - complete buyin
SIM INT Quality of Questions Factor	Converter	An overall factor (0-100) that represents the overall quality of questions in the interviews. This factor is generated by (= appropriate questions / total questions asked). Appropriate questions are seen as clear and complete questions that are appropriate to ask the interviewee.
Actual Interview Time Delta	Converter	Converting the number of any additional interviews (than those planned for) to

		calendar days by multiplying the number of meetings by 5 (days), due to a 5 days per workweek. If the number of sessions is less than or equal to the number planned for, then there is no time penalty.
Time Delta Cost Overrun INT	Converter	The cost overrun incurred for the difference between the planned number of interviews and the number actually needed. The overrun is the product of the difference.
Planned Interview Sessions	Converter	The number of interviews that were planned for.
Requirements Analysis		
SIM Req Scope Efficiency Factor	Converter	Factor (0-100) of the user's ability to correctly identify the scope of a set of requirements.
SIM Func Nonfunc Analysis Efficiency Factor	Converter	Factor (0-100) of the user's ability to correctly identify whether a set of functions are of the type functional or nonfunctional.
SIM Req Prioritization Efficiency Factor	Converter	Factor (0-100) representing the effectiveness of the user in correctly identifying the priority of the requirements.
SIM Req Volatility	Converter	Factor (0-100) representing the

Efficiency Factor		effectiveness of the user in correctly identifying the volatility of the requirements.
Analyze FM Raw Requirements	Rate	The rate that requirements are correctly analyzed. per session (meeting) – a percentage.
Analysis Result Time Usage FM	Converter	This shows how the analysis selections affect the overall schedule (in days) in the time needed to complete the task. Based on assigning a preset overrun value to the <i>Analyze FM Raw Requirements</i> value.
Analysis Result Cost Overrun FM	Converter	The cost overrun incurred correlates to the schedule overrun for the analysis activity.
Analyze Result Time Usage INT	Converter	This shows how the overall analysis affects the overall schedule in days. If done well, the task will have no delay. If not done well, then there will be some delay depending on how "bad" the analysis was done. Based on assigning a preset overrun value to the <i>Analyze INT Raw Requirements</i> value.
Analyze Result Cost	Converter	The cost overrun incurred correlates to the

Overrun INT		schedule overrun for the analysis activity.
Analyze INT Raw Requirements	Rate	The rate that requirements are correctly analyzed. per session (meeting) – a percentage.
Set of Analyzed Requirements	Store	The set of requirements that has been analyzed.
Requirements Validation		
Validate Req	Rate	The rate of inspecting the requirements is based on the percent validated correctly per inspection session
SIM Percent Assessed Correctly VAL	Converter	The percent of correctly inspected (by classification) requirements for the ongoing inspections (an ongoing average).
Actual Validation Result Time Usage VAL	Converter	Based on how well the validation was done. an overrun will result. If the validation was well done, then there is no overrun incurred in the overall schedule. If the validation was not done well, then there will be an overrun (depending on how bad the validation was) to the overall schedule - in days.
Actual Cost VAL	Converter	The actual cost for the requirement

		validation activity, including overruns.
Set of Validated Req	Store	The set of requirements that have been validated
Implementation		
Implemented Req	Rate	The rate of implementing the requirements per day. This rate is assigned from a preset value selection, based on the difference of the overall quality of the requirements and any additional work that results from volatile requirements.
Add'l Work	Store	Work that is new work due to requirements changes (as a result of volatility).
New reqts rate	Rate	New requirements rate is formulated as a percentage of new requirements work that arises. If the volatility flag is off (0), then no new work is present.
Requirements volatility flag	Converter	A switch representing if there are volatile requirements that exist. (0 = requirements volatility effects are off. 1 = requirements volatility effects are on)
SIM percent of new reqts work	Converter	Percent of new requirements work, as determined by the simulator.

Overall Quality of Requirements factor	Converter	Factor (0-100) representing the quality of the overall requirements, based on the quality of requirements from previous activities.
Set of Delivered Code	Store	The set of code, the developed requirements, that has been delivered as the "finished" product.
Calculations of Cost and Schedule Overruns		
Calculate Anticipated Cost Overrun	Converter	The ongoing cost overrun calculation subtracting all ongoing overruns from the planned budget.
Estimated Budget	Converter	The planned budget for the project, in dollars.
Calculate Anticipated Schedule Delays	Converter	The ongoing schedule overrun calculation subtracting all ongoing overruns from the planned schedule.
Estimated Overall Project Schedule Duration in Days	Converter	The planned schedule for the project, in days.

APPENDIX E
SIMULATOR SCREENSHOTS

Project Description

Title University Course Registration System (Web-based)

Overview

The purpose of the project is to provide a consistent course enrollment system for all California State Universities. The system will be accessed via the World Wide Web. Besides course enrollment, students can also check their schedule and grades. In addition, administrative features are included to provide system data.

Audience

Students from diverse backgrounds attend courses in the California State University (CSU) System, a system of public universities throughout California. The CSU system consists of 23 campuses, 370,000 students, 40,000 faculty and staff. Most universities are on the semester calendar, but some campuses are on the quarter system. Summer school is available at all campuses.

Continue

Figure E1. Project Description

Gathering Requirements: Facilitated Meeting

Select up to 2 stakeholders to participate in the meeting.

- Student Representative
- Class Schedule Office Administrator from CSU Northridge
- Registrar's Office Representative from San Diego State University
- Information Technology representative from Cal Poly SLO
- Usability consultant specializing in Disabled & International Interfaces
- CSU Regents Office Representative
- CSU Financial Aid Office Representative from San Jose State

Continue

Figure E2. Facilitated Meeting: Selecting the Stakeholder

Gathering Requirements: Interview

Given each topic to be addressed in the interview, select the 5 best examples of clear and complete questions that you would include in the set of interview questions.

Topic: Security and Privacy

- What privacy precautions or policies should be built into the system?
- What recourse is available if a student forgets his/her password?
- Can the student's social security number be used as an initial password or as a database key?
- How will the data archives be secured?
- What privacy measures need to be in place when students use the system in a lab setting?
- How will the interface with the Financial Aid system be secured?
- Will the student data need to be encrypted?
- Do you want to do all system administration remotely?
- Where will the student's data be secured and maintained?
- What personal information can be viewed or changed by a student in the system?

Continue

Figure E5. Selecting Interview Questions

Gathering Requirements: Interview

Below is a sampling of the outcome of the interview. Read through each question and the stakeholder's answer. The responses will represent how knowledgeable the stakeholder is in the topic areas you selected.

Topic: Grade Display and Input Process (into the system)

Q: Is it possible to have a course without a grade assigned to it at the end of the term?

A: Yes, an Incomplete will have an 'I' instead of a grade. Also if the instructor does not get the grades in on time it will be blank.

Interview Number:

Project Overruns
Schedule (Days) Cost (\$)

Continue

Figure E6. Final Interview Status Screen for a Topic

Requirements Analysis: Scope

Identify whether each of the following requirements are within the scope of the system. The accuracy of the results will be used as input to the simulation.

	Within Scope	Not Within Scope
The system will allow the student to view but not change their Student ID, name, enrollment status and major.	<input checked="" type="radio"/>	<input type="radio"/>
Students will be able to access their program of study, while logged into the system.	<input type="radio"/>	<input checked="" type="radio"/>
The system will not allow students with a hold on their records to log on to the system.	<input type="radio"/>	<input checked="" type="radio"/>
Performance: The system will have a response time less than or equal to 5 seconds for each individual course addition or removal request. The system is assuming that the student's computer has a 56K modem connection or greater.	<input checked="" type="radio"/>	<input type="radio"/>
The system will allow students to communicate directly with the Financial Aid Office.	<input checked="" type="radio"/>	<input type="radio"/>
When a student drops a course with a lab, the system will warn them to register for another lecture section before they log out or the lab will be dropped. If they do log out without registering for another lecture the system will drop their lab.	<input checked="" type="radio"/>	<input type="radio"/>
Once the Student Information System has flagged a student's file for graduation in the current term, the system will not allow the student to access the system in any future terms unless the flag is removed in the SIS or the student is admitted for another degree.	<input type="radio"/>	<input checked="" type="radio"/>

Figure E7. Requirements Analysis: Classification Based on Scope

Requirements Validation

You will now participate in part of a requirements inspection in the role of a Requirements Engineer. Your selections will contribute to the overall inspection rather than constitute the entire inspection in itself.

The current inspection emphasis: Student Registration Questions

The current inspection team consists of:

- 3 other representatives from your Requirements Engineering group.
- 2 representatives from the Design group.
- the Registrar's Office representative stakeholder.
- You.

Select a stakeholder to include on the inspection team if you wish. You may elect to not invite a stakeholder if you feel that it is not needed.

- Information Technology representative from Cal Poly SLO
 Usability consultant specializing in Disabled & International Interfaces
- Student Representative
 No other stakeholder is needed

Figure E8. Initial Validation Screen

Requirements Validation

Using the provided checklist, review each of the following requirements to assess whether it is acceptable (as is), needs revision to correct an error, or it is out of your domain. Your contribution is included in the overall inspection, along with the other's in the inspection team.

	Accept	Revision Needed	Can't Review It (Out of Domain)
Once the Student Information System has flagged a student's file that he/she is graduating, the system will not allow access to the system for future terms unless the flag is removed in the SIS.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system will handle more than 200 concurrent students at each campus.	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
When a student registers for a variable-unit course, the system requires the student to enter the number of units that he/she wishes to register for.	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
The system is not responsible for notifying newly admitted students of their early registration position. The university notifies the students.	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
The system will allow students to add a class until the end of the first week of classes.	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
The system will accommodate courses with labs, variable-credit courses, and auditing of courses.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
The system will allow students to register for more than one term at a time without having to log into the system again. They will need to reenter it the main page for each term.	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

[Continue](#)

Figure E9. Inspecting Requirements

Requirements Validation

Below is the feedback from the other inspectors in the group. They are responding to your inspection, as well as the overall quality of the requirements based on their expertise.

Representatives from your Requirements Engineering group:

We didn't find too many requirements that need revision either. I thought we are a bit shaky on some of the details on the registration process. We will send them over to the Design group to work on.

Representatives from the Design group:

Hey, we found that several of these requirements are fine. Maybe you should review your notes. There are many different possibilities regarding types of classes, students, and grades that the system must address.

Registrar's Office representative:

If you aren't sure about the answers to these topics, you should let others with the expertise check over some of these requirements.

Project Overruns
Schedule (Days) Cost (\$)

[Continue](#)

Figure E10. Feedback from an Inspection

Development and Testing

Below is a sampling of feedback from your teammates as they progress through the design, implementation, and testing phases. After reading each set of feedback, you need to select the CONTINUE button to continue.

It is difficult testing several of the nonfunctional requirements since they were not specified quantitatively.

Project Overruns	
Schedule (Days)	Cost (\$)

Several of the requirements were specified so vaguely that we are not sure if we are testing them properly.

Continue

Figure E11. Sample Developer Feedback During Product Development

Maintenance

Below is a snapshot of the system in use. Take a moment to examine how your strategy during development manifested itself as a product.

Customer Feedback:

The system will be usable for the time being with major workarounds. We will need to meet with you in the next couple of weeks in order to get critical requirements fixed right away.

Project Overruns	
Schedule (Days)	Cost (\$)
30	50000

Continue

Figure E12. Customer Feedback after Delivery

Maintenance: Change Submission

You are a member of the Change Control Board, representing your team of Requirements Engineers. The following requirements changes have been submitted. Analyze the submission, based on the supplied heuristics/checklist. Then select the priority that each change should be applied.

The Registrar's office representatives do not like the layout of the Help section. They think that it needs to be context-sensitive and should look "nicer". You have not heard any complaints from the student focus groups or any other party. The time needed to reassign the Help section would be 1 to 2 months.

- High Priority - include now
 Medium Priority - include in next release
 Low Priority - possibly a future release

Continue

Figure E13. Sample Change Submission

Maintenance: Change Submission

Below is a snapshot of the developer's comments, user's comments, and any resources needed to execute your decision regarding this change submission. Select Continue to proceed.

Representatives from Design, Implementation, and Testing:

We may need to delay the change as more analysis is needed. We need to give our attention to more important matters.

Customer Representative:

We are not sure if we need the change this soon, but if that works out then great.

Continue

Figure E14. Sample Feedback After Change Submission Analysis

APPENDIX F
THE ASSESSMENT

Requirement Engineering Assessment

For each question, select the best answer. If you do not know the answer to a question, select "I do not know." Some of the questions will refer to one of the following scenarios.

Scenario 1

You are a requirements engineer on a project that will replace a university's student information system. The system is used to maintain student enrollment status information, course grades, student status, and general student information (e.g. name, address). The new system will interface with the Financial Aid system. Also, the new system will include course registration. Students will be able to register for classes using the Web. The task of generating reports to inform the colleges' advising centers of students on academic probation will become automated. The automation will decrease the time needed for advisors to monitor academically-challenged students, and allow for more time for advisors to meet with the students. The process of entering grades into the system at the end of the semester will improve productivity by partial automation. Rather than submit grades in handwriting, faculty will record grades using a form that can be read by an optical scanner.

Scenario 2

You are a requirements engineer on a project that will replace the current training system for a large company's training department. The orientation process will be online rather than in a traditional setting with a trainer and a group of new employees (either new to the company or to a new position within the company for a current employee). Some professional development courses will be online for use at the employee's leisure, while other courses will be conducted traditionally with computer-based activities. The course registration and payment systems will be revised to improve productivity through partial automation. The employee's department pays for the courses that the new employee needs to complete.

Change Proposal #1 (for Scenario 2)

After the system was delivered, several department representatives express interest in being able to check whether or not their employee has completed his/her courses or orientation. The worker's supervisor would be able to see the date and time that the employee completed the orientation and/or courses without the employee's consent.

1. Which one of the following best describes a facilitated meeting as a requirement elicitation technique?
 - a. Facilitated meetings are unstructured brainstorming sessions between the stakeholders and the developers..
 - b. Facilitated meetings allow the customers to ask the developers questions about the project.
 - c. Facilitated meetings are structured brainstorming and problem-solving sessions between the stakeholders and the developers.
 - d. Facilitated meetings provide the status of the schedule and budget for the project.
 - e. I do not know.

2. Which one of the following best describes an interview as a requirement elicitation technique?
 - a. Interviews are unstructured brainstorming sessions with a single stakeholder.
 - b. Interviews are unstructured brainstorming sessions with multiple stakeholders.
 - c. Interviews are narrowly-focused facilitated meetings (in terms of the topics covered).
 - d. Interviews are question-and-answer sessions with one or more stakeholders.
 - e. I do not know.

3. In the area of requirements analysis, requirement prioritization is best described as:
 - a. The task of assigning a rank to the importance of each system requirement.
 - b. The task of determining the ease of implementation of each system requirement.
 - c. The task of estimating the testing of each system requirement.
 - d. The task of assigning a rank to the ease of each system requirement's traceability factor.
 - e. I do not know.

4. In the area of requirements analysis, a requirement's scope is best defined as:
 - a. The extent to which a requirement affects the system's functionality or attributes.
 - b. The extent to which a requirement reflects the systems nonfunctional requirements.
 - c. The extent to which the requirement set is feasible in the system schedule.
 - d. The extent of the completeness of the requirement set.
 - e. I do not know.

5. In the area of requirements analysis, a requirement's volatility is best defined as:
 - a. The extent to which a requirement is likely to change during development.
 - b. The extent to which a requirement affects the other functional requirements.
 - c. The extent to which a requirement can be tested.
 - d. The extent to which a requirement affects the development schedule
 - e. I do not know.

6. In the area of requirements analysis, a functional requirement is best defined as:
 - a. a feature desired by the end user.
 - b. a feature that the system must be able to perform
 - c. a system attribute or constraint
 - d. I do not know.

7. In the area of requirements analysis, a nonfunctional requirement is best defined as:
 - a. the hardware requirements of the system.
 - b. a system feature
 - c. a function of the system with low priority
 - d. a system attribute or constraint
 - e. I do not know.

8. Which one of the following is an appropriate stakeholder to include on a requirements validation team?
 - a. Customer
 - b. Market analyst
 - c. Domain expert
 - d. A and C
 - e. A, B, and C
 - f. I do not know

9. Which one of the following is the purpose of the Validation Checklist, used during the requirements validation tasks.
- To provide a checklist, whereby each requirement is checked off after it is determined that it is valid.
 - To provide a document where the inspectors can list the requirements that need to be revised.
 - To provide a checklist that enables inspectors to check that each requirement meets all listed criteria.
 - To provide a document to list the defects that are detected during the requirements validation tasks.
 - I do not know
10. You are working on a large project where several stakeholders are providing some conflicting requirements. Also, you know that some of the requirements cannot be completed until the design phase. As the project enters testing, the customer asks that the system be administered remotely rather than at the server. Which one of the following is the point where requirements validation activities should be conducted?
- During the requirements analysis phase
 - During the design phase
 - During the testing phase
 - A and B
 - A, B, and C
 - I do not know
11. Which of the following best describes the role of change management throughout the development lifecycle?
- Change management is intended to identify, control and track requirements and any changes to requirements at any time.
 - Change management is intended to identify, control and track requirements and their changes during the Requirement Specification phase only.
 - Change management is intended to monitor and integrate requirement revisions at any time.
 - Change management is intended to assess and prioritize requirement revisions during the maintenance phase.
 - I do not know.

Please see Scenario 1 for the question 12 through 22.

12. Which of the following stakeholders would you select to interview regarding the current system in order to elicit requirements for the new system most efficiently? You can select up to 4 stakeholders.
- University administration
 - Registrar's office staff
 - Students
 - Representatives of the departmental administrative assistants (secretaries)
 - Financial Aid Office
 - Associated Students Incorporated student government office
 - Student Advisors (from the Department College Advising Center)
 - I do not know.
13. Which one of the following requirements can be classified as a nonfunctional requirement?
- In order to allow the university to easily support the Web registration client, the Microsoft Internet Explorer version 5 or higher must be used.
 - The system generates an official transcript when a student has completed four semesters at the university.
 - The system generates a report, a list of students' names who are not registered for more than 12 credit hours, and sends it to the Financial Aid office.
 - A student begins the registration process by entering his or her student id and password.
 - I do not know.
14. Which one of the following requirements can be classified as being out of scope for the system?
- The system generates a report, a list of students' names who are not registered for more than 12 credit hours, and sends it to the Financial Aid office.
 - The system generates an official transcript when a student has completed four semesters at the university.
 - The Human Resources Payroll system is alerted by the Financial Aid System when a student's Federal Work Study payments are terminated due to low grades.
 - A student begins the registration process by entering his or her student id and password.
 - I do not know.

15. Which one of the following requirements can be classified as having a low priority (optional)?
- The course registration module also includes hyperlinks to the course descriptions in the course catalog.
 - The system generates the individual student grade reports after end-of-semester grades have been submitted.
 - The system generates a report, a list of students' names who are not registered for more than 12 credit hours, and sends it to the Financial Aid office.
 - A student can change his or her password in the course registration module.
 - I do not know.
16. Which one of the following requirements can be classified as having high volatility?
- The course registration module is accessible to visually impaired students.
 - The unofficial transcript includes a student's semester and cumulative grade point averages.
 - The Grade Report layout consists of a table containing the student identification information and a table listing the courses and grades.
 - A student begins the registration process by entering his or her login id and password.
 - I do not know.
17. You have analyzed the set of requirements from Scenario 1. You classified the following requirements as being out of the system's scope.
- A student's tuition payment status is kept in the Financial Aid System.
 - Students can change their school and permanent addresses online.

Which one of the following is the likely result of this classification.

- The customer will ask for the address update feature later since you are omitting it now. Adding the feature into the registration module later will delay system delivery.
- The system's schedule and budget will remain on target.
- More developers will need to be hired to complete the Financial Aid system and the module to facilitate the address change since you are including these feature now..
- The design team will ask for a clarification of the term "tuition payment status."
- I do not know.

18. You have analyzed the set of requirements from Scenario 1. You classified the following requirements as being Optional, in terms of priority.

- When the student's residency code is "OTS", the out-of-state tuition announcement will be displayed during course registration.
- Each student's password, used for registration, is synchronized with the password on the university computer account.

Which one of the following is the likely result of this classification.

- a. The use of the residency code is automatically removed so that all students see the announcement. The schedule and budget are unaffected.
- b. When development is behind schedule, both requirements are removed. Upon system delivery, students complain about having yet another password. The feature is then re-added at your cost.
- c. During testing, the customer revises the password requirement to provide students with randomly-generated passwords.
- d. I do not know.

19. You have analyzed the set of requirements from Scenario 1. You classified the following requirements as being Highly Volatile.

- The student's local address is used when grade reports are mailed.
- A report, a list of students' names who have registered for fewer than 10 units, is generated for the Financial Aid office at the end of the fourth week of the semester.

Which one of the following is the likely result of this classification..

- a. Subsequent changes to the mailing address and the report surprise the development team. The schedule is pushed back and the development costs are increased.
- b. Subsequent changes to the mailing address and the report that occur during design and testing are anticipated and accommodated in the schedule and budget.
- c. Subsequent changes to the mailing address and report need to occur at various times. The inflexible design struggles to accommodate the changes.
- d. I do not know.

20. You are planning the meetings to validate the set of requirements for the system. Select the most appropriate stakeholders to include in the inspections. You can assume that other requirements engineers and developers are already in the group. Select up to 3 stakeholders.
- Registrar's office staff
 - Students
 - Faculty
 - Financial Aid Office
 - Student Advisors (from the Department College Advising Center)
 - No other stakeholders are needed.
 - I do not know
21. At the inspection, many defects are revealed that need to be fixed by the requirements engineers. In addition, you notice several potential problem areas in the system. Which one of the following documents best represents the type of document that you need to provide for the requirements engineers?
- Validation checklist
 - Review summary report
 - Review issues list
 - I do not know
22. At the requirements phase, your team conducts the initial requirements validation activity. More validation is conducted after substantial rework is conducted by the requirements engineers. During the subsequent phases and maintenance, the requirements are validated every three months by the original inspection team. Based on this description of the requirements validation process schedule, which one of the following best illustrates the reasoning behind the requirements validation schedule.
- The set of requirements is continuously changing and needs to be checked throughout the lifecycle.
 - The project schedule is slipping due to requirements problems.
 - The developers are not experienced in the particular domain.
 - Customer interviews are not progressing according to the project schedule, which requires the team to keep rechecking the updated set of requirements.
 - I do not know

Please see Scenario 2 for Questions 23 through 26.

23. Select which of the following stakeholders are needed at the facilitated meetings in order to elicit requirements most efficiently. You can select up to 4 items.
- a. Sampling of new employees
 - b. Training personnel
 - c. Sampling of departmental accountants
 - d. Sampling of current employees
 - e. Sampling of purchasing liaisons for the various departments
 - f. Sampling of department heads
 - g. All department heads
 - h. I do not know
24. Once the change proposal, described above, has been submitted, the next step in Change Management is to evaluate the change in terms of how it will affect parts of the project directly and indirectly. Select the motive that best presents the need for such evaluation.
- a. Evaluating each change will allow each change to be implemented as soon as possible.
 - b. Each proposed change means that parts of the project will need to be redone. The extent of such work needs to be noted as soon as possible.
 - c. Each proposed change needs to be assigned to an available programmer.
 - d. Each proposed change has an impact on the cost, schedule, quality, and other aspects that needs to be considered.
 - e. I do not know.
25. Change Proposal #1 is being considered. Select the primary consideration that must be addressed before you can decide to implement the change.
- a. The amount of time the change will take to implement.
 - b. The person who will be assigned to make the change.
 - c. The affect of the change on the schedule and resources
 - d. The affect of the change on customer support's workload.
 - e. I do not know.

26. Given that Change Proposal #1 has been accepted and implemented. What is the final step in the Change Management process?
- a. Release the new version of the system to the customer.
 - b. Audit the change to see if it was completed correctly.
 - c. Implement the change.
 - d. Update the design document.
 - e. I do not know.

APPENDIX G
ASSESSMENT TRACABILITY MATRIX

Software Requirements Analysis	Pre-Test (Benchmark)	Question #	Post-Test (Objective)	Question #
II. Requirements Elicitation				
B. Elicitation Techniques				
1. Interviews	Comprehension	2	Application	12
3. Facilitated Meetings	Comprehension	1	Application	23
III. Requirements Analysis				
A. Requirements classification				
1. Functional & Nonfunctional	Knowledge	6.7	Comprehension	13
4. Priority	Knowledge	3	Comprehension	15
5. Scope	Knowledge	4	Comprehension	14
6. Volatility	Knowledge	5	Comprehension	16
V. Requirements Validation				
A. The conduct of requirements reviews				
1. Group composition is appropriate (may include customer)	Comprehension	8	Application	20
2. Use of guiding documents like checklists to guide review and to doc findings	Comprehension	9	Analysis	21
3. Review process is at specified checkpoints and redone as appropriate	Application	10	Analysis	22
VI. Requirements Management				
A. Change management				

1. Understanding the role of Change Management throughout lifecycle	Comprehension	11	Application	-
2. Have procedure in place	Comprehension	24	Application	26
3. Analyze proposed changes	Comprehension	24	Application	25

All question numbers are based on the order used for the pre-test.

Other levels of understanding were assessed as well, in order to ascertain the extent of understanding of a topic. For example, requirements classification was tested at the Application level (question # 17, 18, and 19). Also, while students do conduct interviews to a limited extent, the pretest benchmark was set to Comprehension in order to address the concept of Interviews in a more general sense. The pretest benchmark was also set to the Knowledge level for the Functional Nonfunctional and Volatility classifications (in are III.A) to measure the extent of understanding of these requirement types. The Knowledge level was selected as the benchmark for Functional Nonfunctional classification due to historical patterns of misunderstanding in past exams. Since Volatility is barely addressed in the course, the Knowledge level was selected.

Furthermore, a question was not developed to test topic VI.A.1 at the Application level, due to the incompatibility of the topic with multiple-choice assessment. Nonetheless, the pretest benchmark was assessed in order to measure the level of understanding of the role of change management throughout the lifecycle.