121st ASEE Annual Conference & Exposition
Indianapolis, IN
June 15-18, 2014
Paper ID #9429

A "Software and Systems" Integration Framework for Teaching Requirements Engineering

Radu F. Babiceanu, Embry-Riddle Aeronautical University

Dr. Radu Babiceanu is an Associate Professor with the Department of Electrical, Computer, Software, and Systems Engineering at Embry-Riddle Aeronautical University in Daytona Beach, Florida. He holds a Ph.D. degree in Industrial and Systems Engineering from Virginia Tech, a M.S. in Mechanical Engineering from the University of Toledo, and a B.S. degree in Manufacturing Engineering from the Polytechnic University of Bucharest. His research provides a systems engineering approach to modeling and operation of large-scale complex systems, including requirements, architecture, integration, and evaluation of systems considering their lifecycle effectiveness and sustainability characteristics.

A "Software and Systems" Integration Framework for Teaching Requirements Engineering

Abstract

Most of the engineered systems used in our daily lives include some sort of a computing unit that runs software. The integration of systems engineering and software engineering disciplines to enhance the engineering design process grew significantly in the last decades to become the norm in the engineering world. Consequently, a coordinated approach is needed to educate the new generation of systems engineers able to design and build the integrated engineering and software systems. At Embry-Riddle Aeronautical University, there is an effort underway to do just that: integrate the software engineering and systems engineering education for graduate engineering students. The effort started in the Fall 2013 semester with a combined offering of the "*Software Requirements Engineering*" and "*System Requirements Analysis and Modeling*" courses. Serving as instructor for the "combined software and systems requirements" course comes with inherent challenges due to students' different backgrounds, but also provides opportunities for the instructor to address the need for coordinated software and systems engineering education. This work reports the pedagogical methodologies used and the findings uncovered during the entire offering of the "combined software and systems requirements" course.

1. Introduction

It is long ago now when engineered systems were made only from physical components and merely the traditional engineering disciplines were involved in the creation of systems. Nowadays for example, every manufactured vehicle used for our daily commute includes some sort of a computing unit that runs software. Software engineering is not anymore needed only for, let's say, computers and space rockets, it is now part of most of the engineered systems we use in our daily lives. We are also moving towards building smart homes and unmanned autonomous vehicles, which include software execution at their core. Consequently, the integration of systems engineering and software engineering disciplines to enhance the engineering design process became the norm in the engineering world. The recently issued *Guide to the Systems Engineering Body of Knowledge* (SEBoK)[1] explicitly recognizes and embraces the intertwining between systems engineering and software engineering:

> "Software is prominent in most modern systems architectures and is often the primary means for integrating complex system components. Software engineering and systems engineering are not merely related disciplines; they are intimately intertwined."

Therefore, a coordinated approach is needed to educate and graduate the new generation of systems engineers able to design and build the integrated engineering and software systems. One key solution to train proficient systems engineers is to provide them with knowledge of both

systems engineering and software engineering starting with their higher education years. The College of Engineering faculty at Embry-Riddle Aeronautical University acknowledged that and started a coordinated effort to address the needed software and systems integration. The "software and systems education integration project" started in the Fall 2013 semester with the offering of the graduate "*Software Requirements Engineering*" and "*System Requirements Analysis and Modeling*" courses in one combined section. The first course is required for the master's program in Software Engineering, while the second one is an elective course for the students enrolled in any other graduate engineering programs. The encouraging course discussions and student feedback received during Fall 2013 halfway into the semester strengthened the faculty belief in the software and systems integration effort and prompted the implementation of the other proposed combined sections. Therefore, the Spring 2014 semester was scheduled to feature two new combined sections part of the "software and systems education integration project" as follows:

- A combined section of the "*Software Project Management*" and the "*Engineering Project Management*" courses; and,
- A combined section of the "*Software Quality Engineering and Assurance*" and the "*System Quality Assurance*" courses.

The experience and findings to be revealed during these two offerings is envisioned to be disseminated at the 2015 ASEE Annual Conference. Recognizing also the inherent differences between the software and systems engineering disciplines, the faculty recommended that other pairs of candidate courses existing in the curriculum ("*Software Systems Architecture and Design*" - "*System Architecture Design and Modeling*" and "*Software Safety*" - "*System Safety and Certification*") be kept as individual offerings. With the expected development of a model-based systems engineering course in the near future, another candidate course to round-up the "software and systems engineering education integration project" will be the combined course: "*Model-Based Verification of Software*" (already listed in the catalog) and "*Model-Based Systems Engineering*" (the proposed name for the systems engineering counter-part course). The current and tentative offerings and their association with the graduate engineering curriculum are summarized in Table 1 below.

Table 1: Summary of the software and systems engineering education integration project

Integration status	Systems Engineering Courses	Curriculum requirement	Software Engineering Courses	Curriculum requirement	First offering
Combined sections	*System Requirements Analysis and Modeling*	R[1] - MS SysE[3] E[2] - MS ECE[4]	*Software Requirements Engineering*	R - M SE[5]	Fall 2013
	Engineering Project Management	E - MS SysE R - MS ECE	*Software Project Management*	R - M SE	Spring 2014
	System Quality Assurance	R - MS SysE E - MS ECE	*Software Quality Engineering and Assurance*	E - M SE	Spring 2014
	Model-Based Systems Engineering[6]	E - MS SysE E - MS ECE	*Model-Based Verification of Software*	E - M SE	TBD
Separate sections	*System Architecture Design and Modeling*	R - MS SysE E - MS ECE	*Software Systems Architecture and Design*	R - M SE	N/A
	System Safety and Certification	E - MS SysE E - MS ECE	*Software Safety*	E - M SE	N/A

[1]Required; [2]Elective; [3]Proposed Master of Science in Systems Engineering; [4]Master of Science in Electrical and Computer Engineering; [5]Master of Software Engineering; [6]Proposed course.

This current work reports the experience and findings uncovered during the entire offering of the combined "*Software Requirements Engineering*" and "*System Requirements Analysis and Modeling*" class during the Fall 2013 semester. The literature review identified related research with this current one that describes the pedagogy to teach systems engineering concepts to software engineering students and also software engineering concepts to systems engineering students[2-4]. The difference from this current work is given by the target student audience level (undergraduate vs. graduate student audience), and the application domain (general concepts vs. specific requirements engineering topic).

From this point forward, the paper is structured as follows: Section 2 presents specific background information and outlines the motivation for the proposed course integration. Then, Section 3 presents in detail the entire teaching experience coming from the combined software and systems requirements engineering course. Finally, Section 4 outlines the most important lessons learned during the semester that are to be applied in the subsequent Spring semester offerings of the combined sections for the software and systems quality and project management courses. The Appendices located at the end of the paper present the course catalog descriptions of the two requirements engineering courses under evaluation in this work, and a problem statement example for one of the assigned requirements engineering course projects.

2. Background and Motivation

In the last decades, the need for systems engineering increased exponentially due to the upsurge of larger and more complicated systems being developed and operated worldwide. The academe responded with a sustained growth in the number of systems engineering graduate programs offered nationally and internationally[5]. Also, it is obvious to everyone, not only to the computing and engineering communities, the enormous growth experienced in the last decades by the software industry. Consequently, the academe responded as well, not only by preparing competitive graduates in computer science and computer engineering who have the adequate knowledge to build, operate and maintain software systems, but also by proposing stand-alone software engineering programs.

The motivation for implementing the "software and systems education integration project" at Embry-Riddle Aeronautical University is twofold:

- First, as the engineered systems continue to grow in complexity and depend even more on software execution, the new systems engineering graduates require more in-depth software engineering knowledge than ever before to be able to carry out the systems engineering design and operational tasks.
- Second, software engineering also recognizes the increased complexity of today's systems, the expected even larger complexity of future's systems and the widespread inclusion of software in almost every type of engineered system built today and/or envisioned for the future.

These facts prompted the engineering faculty at Embry-Riddle to seek a solution for the integration of the related software engineering and systems engineering courses part of the graduate engineering curriculum. The expected outcomes of this integration are:

- An increased exposure to the software engineering development methods and tools for the electrical and computer engineering, and potentially other engineering disciplines students who consider entering systems engineering career paths; and,
- An increased familiarity with the systems engineering process, systems analysis methods and tools, and system operational maintenance for the software engineering students who consider careers in software development for large-scale systems.

There is consensus among the industry practitioners that superior requirements engineering is critical for the development of quality systems[6]. Moreover, academe people consider that the software industry use of requirements engineering is obstructed by relatively poor understanding of requirements engineering practices and benefits. In this context, teaching requirements engineering at university level becomes a critical responsibility[7]. Scheduled for the Fall 2013 semester, the "*Software Requirements Engineering*" course was found to served well the software and systems engineering education integration purpose, so the corresponding systems engineering course ("*System Requirements Analysis and Modeling*") was made available for the graduate engineering students before the start of the Fall semester. Since there is a clear commonality across the entire software engineering requirements and systems engineering requirements processes, the integration of the two courses was expected to occur without major barriers. Reputed software engineering academics also acknowledged this commonality and referenced it widely in literature[8]. The Embry-Riddle Aeronautical University Graduate Catalog course descriptions of the two courses are presented in Appendix A.

3. Software and Systems Integration Framework for Teaching Requirements Engineering

3.1 Course Preparation. Several steps taken by the instructor during the course preparation stages are outlined below. First, the instructor acknowledged that this combined "software and systems requirements" section is, nevertheless, a course in *Requirements Engineering*. This is the overarching theme of both course syllabi*, as both of them emphasize the requirements engineering process stages. Then, in the same manner as real-life requirements engineers, the instructor had to elicit the specific requirements from the two syllabi, by carefully analyzing the syllabi course descriptions and performing an analysis of the software and systems requirements domains. Given the fact that the enrolled students are likely to have either software or systems development background, the course preparation, management, and evaluation of course learning outcomes needs to address the common aspects of software and system requirements, as well as reconcile the differences between the two.

Since there is no definitive text that covers both the software and systems engineering requirements, the textbook selection could be a time-consuming process with no definitive selection. Rather than requiring a single text, the instructor decided to recommend several authoritative resources from both the software engineering and systems engineering areas and prepared and made available the entire course materials on the Blackboard system for the students to download. Additional materials, listed next, were also made available to the students to compare the seemingly different requirements engineering approaches of the software and systems engineering disciplines.

*The two course syllabi are available upon request by contacting the course instructor at: babicear@erau.edu

- INCOSE Systems Engineering Handbook[9]
- NASA Systems Engineering Handbook[10]
- Systems Engineering Body of Knowledge (SEBoK)[1]
- Software Engineering Body of Knowledge (SWEBoK)[11]

The course learning outcomes were designed to provide the students with an understanding of the software and system requirements engineering process and its fundamental role in the software and systems engineering process. Upon the completion of the course, the students are expected to be able to:

- Describe the essential elements of software and system requirements engineering.
- Identify the major issues in large systems and software systems development.
- Describe and understand various requirements specification techniques.
- Select a set of requirements techniques, tools, and/or languages that help in analyzing software and systems requirements.
- Identify tools and techniques for software and systems requirements management.
- Develop, analyze, and critique requirements and specifications for large-scale software and systems.

3.2 Highlighting Similarities of Software and Systems Engineering Disciplines. The tentative course schedule proposed in the combined course syllabus included the topics one expects are covered in a requirements engineering course, such as:

- Requirements Stakeholders
- Requirements Elicitation
- Requirements Analysis
- Requirements Specification
- Requirements Verification and Validation
- Requirements Management
- Requirements Standards and Tools

One key aspect that supports the software and systems engineering integration project is the fact that the terminology used for software requirements engineering and systems requirements engineering is similar. Even though the disciplines of software engineering and systems engineering may have different terminology, in the case of requirements engineering, the terminology is practically the same. At the same time, all the above listed requirements engineering topics can be covered from either the software or systems engineering points of view individually. However, the key for the software and systems engineering integration effort is to cover the above topics from a common overall system viewpoint. Given the instructor's background the task was carried out using systems engineering integration principles. Each of the specific topics listed above was addressed during lectures with the system viewpoint in mind. Also the homework assignments and class discussions were designed and conducted with the system viewpoint as focal point.

There was no coding required, which was somehow unexpected especially for the software engineering students, as uncovered from the student feedback received during the semester. But, the class time was wisely used to relate all the covered material to real-world requirements

engineering projects. All course lectures were complemented with real-world examples from software and systems engineering area to familiarize the students with real-world project implementations. Also, the class discussions and assignments asked them to understand, discover, specify, analyze, and verify system requirements for the duration of the entire semester. There was a very fine level of detail targeted for the requirements specification process in both class lectures and discussions, and all the credit assignments. As the course progressed, it became obvious to the students that the lack of coding did not reduce the amount of work expected in the course, as all the specified and analyzed requirements were always checked for accuracy, completeness, consistency, and correctness.

3.3 Accommodating Differences of Software and Systems Engineering Disciplines. The most obvious difference between software and systems engineering arises from the intangible nature of software and the physical nature of the hardware. But this difference is undergoing a continuous dilution as mentioned at the beginning of this paper: most of today's engineered systems are composed of integrated hardware and software components, and the process is only going in one direction, towards more and more integration.

Starting with the course kick-off, the students enrolled in the engineering graduate master's programs expect that the education received during the course will help them in their future careers. As such, they anticipate that the course lectures and assignments will resemble real-world projects such that they become prepared to apply the knowledge and skills in projects that have relevance to real-world organizations. Besides the acknowledged commonality mentioned in Section 3.2 and above, there are also certain differences between the software and systems engineering disciplines, as the former looks mainly at software development, while the latter considers the entire engineering effort. Since both the software and systems requirements follow practically the same requirements engineering process, the differences would only be a small barrier from the teaching point of view, if not for the integration of the students' dissimilar views of software and systems requirements coming from their different academic backgrounds. Therefore, the instructor designed the course in such a manner that students themselves were encouraged to attempt the reconciliation of their different views through participation in constructive discussions.

3.4 Software and Systems Requirements Engineering Course Assignments. As with all other engineering course offerings, if the instructor wants to successfully achieve the course learning outcomes, the course must include more than traditional lectures. The students must be able to involve themselves in new experiences, in this case by working on complete requirements engineering projects. This approach, depicted in Fig. 1, is called in the literature experiential learning, and includes the students' experience, their skills to observe and reflect on the experiences, their abilities to learn from the experience, and their proficiency to try out the learned facts[12]. A similar pedagogical approach was identified by the literature review in other requirements engineering works. For example, a comparable approach, called "learning by doing," exhibits the goal to provide exposure to student teams in eliciting and specifying requirements[13].

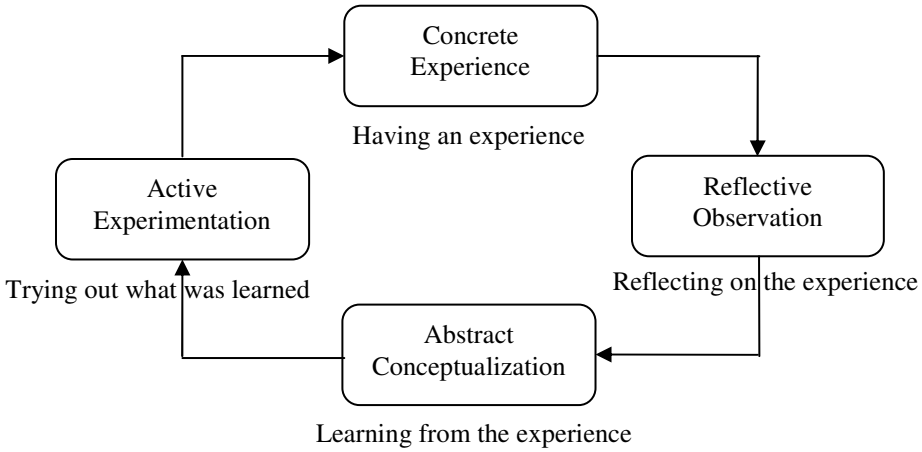

Fig. 1: The Kolb experiential learning framework[12]

Out-of-class assignments, composed of homework assignments and the course project, accounted for half of the course grade, with the remaining half coming from the in-class tests. Both the homework assignments and the course project were designed such that the rationale for correctly answering or addressing the questions follows the above experiential learning framework. This was particularly clear for the course project, which was carefully designed to achieve all the experiential learning aspects. The description of the course project assignment, which accounts for 25% of the final course grade, is presented below.

The purpose of the course project is to provide students with the opportunity to work on a one-month long project in which they can utilize their requirements engineering process knowledge acquired during the course. The main learning objective of the course project is to test students' abilities and knowledge necessary to successfully complete a real-world requirements engineering project. This is accomplished through the completion of the following project assignment: students are tasked to "prepare a written report in the form of a Requirements Document using the *IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications*[14] template for your selected system." The students have the choice of selecting one of the systems listed by the instructor or prepare their own system proposal, in which case the instructor approval is required to assure the same amount of work as in one of the instructor-proposed systems. Each of the instructor-proposed systems, listed below, comes with a one-page problem statement description which helps students to make informed decisions for their project selection.

- *Airline Management Information System*
- *Airport Management Information System*
- *Air Traffic Control Decision Support System*
- *Automated Material Handling System*
- *Condition Monitoring and Fault-Recovery System*
- *Enterprise Decision Support System*
- *Multi-Agent Decision Support System*
- *Remote Healthcare Monitoring System*
- *Supply Chain Information System*

- *System of Systems Integration*
- *Wireless Sensor Network System*

All the instructor-proposed projects are former actual software and systems engineering projects in which the instructor worked in the last several years. This fact helped in providing significant informed feedback to students during their work on the project and after submission as part of the grading process. As an example, Appendix B provides the problem statement the students received for the first project listed above, the "*Airport Management Information Systems*" project.

The detailed project instructions ask students to "read carefully the *IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications* as it provides valuable information related to all the parts of the requirements document." The students are also advised that it may be also useful to review the literature in the area of their selected system such that they can provide a well-defined purpose, scope, overall description, as well as identify specific, not trivial, requirements for their selected system. The deliverables of the project mention that the students' Requirements Document submission "should include all section of the Prototype SRS Outline depicted at page 11 of the *IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications*." In addition, the students' submissions should also address the following tasks:

- Identify the section(s) of your Requirements Document where the information related to the requirements' customers and stakeholders is to be presented. Provide the customers and stakeholders information as part of the document or as an Appendix.
- Identify the section(s) of your Requirements Document where, besides the natural language requirements, the requirements analysis and specification process would benefit from the use of diagrams (use cases, data flow diagrams, state-machine-diagrams, etc.) to better understand the needed functionality. Provide the identified diagrams as an Appendix to the Requirement Document.
- Identify the section(s) of your Requirements Document where the listed requirements would benefit from a formal verification and validation process. Provide the verification and validation process as an Appendix to the Requirement Document.
- Identify the section(s) of your Requirements Document where the listed requirements would benefit from the use of traceability methods to provide a better requirements management. Provide the identified traceability techniques as an Appendix to the Requirement Document.

4. Lessons Learned for Software and Systems Engineering Integration

In only a matter of weeks from the beginning of classes, the instructor noticed the different approaches of the students in terms of the real-life examples they relate to during class discussions. It is the needed work of the course instructor to fine tune the seemingly different approaches of "software or systems" and use them in a "software and systems" integration framework throughout the semester long course. Lectures, discussions, assignments, and case studies are to be developed with the "software and systems" integration framework in mind.

The end-of-the-semester student evaluation of the course revealed that the students were favorable of the instruction received in the combined software and systems requirements

engineering course. Given the challenges raised by the integration aspects of the software and systems engineering disciplines, the instructor considers that the most important assessment related to the "software and systems education integration project" effort is the achievement of the course learning outcomes. A majority of 66.7% of the enrolled students selected the *Strongly Agreed* option, and another 33.3% selected the *Agreed* option when asked to identify themselves with the following statement: "I achieved the learning outcomes for this course." There were no students that selected any of the remaining options (disagree or strongly disagree) for this most important question. In another question, the students were asked to identify themselves with the following statement: "The learning outcomes were addressed via the learning activities in the course." A majority of 55.6% of the enrolled students selected the *Strongly Agreed* option, and another 44.4% selected the *Agreed* option. Once again, there were no students that selected any of the remaining options (disagree or strongly disagree) for this equally important question. Fig. 2 presents graphically the student evaluation of the course learning outcomes depicted above.

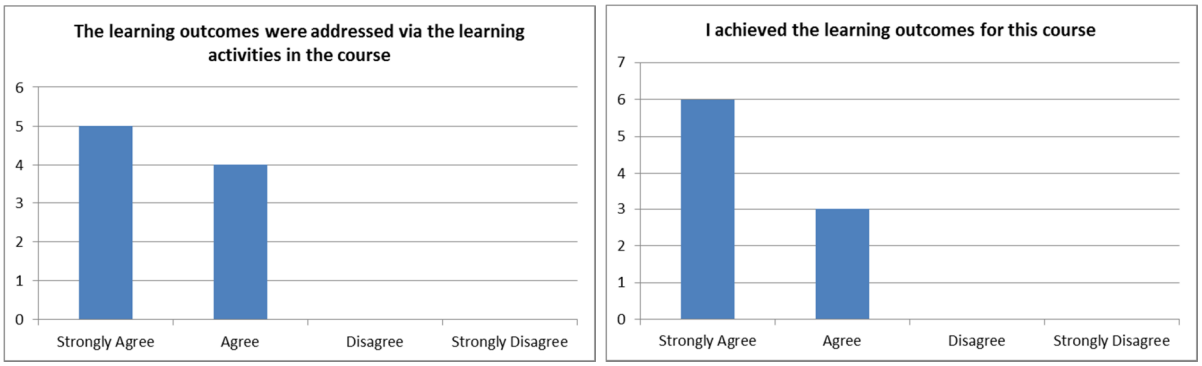

Fig. 2: Student evaluation of the course learning outcomes

As always, anonymous student written feedback is also very useful for the purpose of course assessment and continuous improvement. The comments received from the enrolled students proved that the integration of the software requirements engineering and system requirements engineering processes was successful. One student noted that: "*The breadth of the course provided a way to not only fully understand how to write requirements, but also, the activities and important aspects that surround that process…I had not learned these surrounding activities in depth before, and the time spent on these non-writing-requirements topics were likely the most valuable from the course.*" The student feedback clearly points out the successful integration between the software requirements engineering and the systems engineering processes throughout the course.

As outlined in Section 2, the overall objective of the "software and systems education integration project," attempts to elevate the future careers of Embry-Riddle engineering graduates, by aiming to:

- Equip software engineering students with the systems engineering process knowledge, systems analysis methods and tools, and system operational maintenance practices to effectively perform in large-scale software development projects; and,

- Arm systems engineering and electrical and computer engineering students with adequate software engineering process knowledge, software development and quality assurance techniques to become proficient in managing and communicating across the physical and cyber domains with all the stakeholders involved in the development of large-scale cyber-physical systems.

From this perspective, the instructor reports the first success of the "software and systems education integration project," with the integration of the software and systems requirements engineering courses and looks forward to offering the combined sections for the software and systems quality and project management courses in the Spring 2014 semester.

References

1. Pyster, A., Olwell, D., Hutchison, N., Enck, S., Anthony, J., Henry, D., and Squires A. (eds.), *Guide to the Systems Engineering Body of Knowledge (SEBoK)* version 1.0, The Trustees of the Stevens Institute of Technology, 2012.
2. Fairley, R. and Willshire, M. J., Teaching software engineering to undergraduate systems engineering students, *ASEE Annual Conference and Exposition*, Vancouver, Canada, June 2011.
3. Fairley, R. and Willshire, M. J., Teaching systems engineering to software engineering students, *IEEE-CS Conference on Software Engineering Education and Training*, Honolulu, HI, May 2011.
4. Callele, D. and Makaroff, D., Teaching requirements engineering to an unsuspected audience, *Proceedings of the SIGCSE Technical Symposium on Computer Science Education*, Houston, TX, March 2006.
5. Fabrycky, W. J., Systems engineering: Its emerging academic and professional attributes, *Proceedings of the 117th ASEE Annual Conference and Exposition*, Louisville, KY, June 2010.
6. Mead, N. R., Shoemaker, D., and Ingalsbe, J., Teaching security requirements engineering using SQUARE, *Fourth International Workshop on Requirements Engineering Education and Training*, Atlanta, GA, 2009.
7. Danielsen, O., Teaching requirements engineering: An experimental approach, *Norwegian Information Technology Conference*, Gjovik, Norway, 2010.
8. Laplante, P. A., *Requirements engineering for software and systems*, Taylor and Francis Group, 2009.
9. *INCOSE Systems Engineering Handbook v. 3.2,* INCOSE-TP-2003-002-03.2, International Council on Systems Engineering, 2010.
10. *NASA Systems Engineering Handbook*, NASA/SP-2007-6105 Rev1, National Aeronautics and Space Administration, 2007.
11. *Software Engineering Body of Knowledge (SWEBOK)* version 2, IEEE Computer Society, 2004.
12. Kolb, D. A., *Experiential learning: Experience as a source of learning and development*, Prentice Hall, Inc., 1983.
13. Suri, D. and Durant, E., Teaching requirements through interdisciplinary projects, *Proceedings of the 2004 ASEE North Midwest Regional Conference*, Milwaukee, WI, 2004.
14. *IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications*, IEEE Computer Society, 1998.

Appendix A: Catalog Course Descriptions

***Software Requirements Engineering*:** This course is concerned with the development, definition, and management of requirements for a software system or product. Topics include the software requirements process, requirements elicitation, requirements analysis, requirements specification, requirements verification and validation, requirements management, and requirements standards and tools. Students will participate in individual and group exercises related to software requirements engineering tasks.

System Requirements Analysis and Modeling: This course is concerned with the development, definition, and management of requirements for system or product. Topics include the system requirements process, requirements elicitation techniques, alternative requirements analysis techniques, requirements specification, requirements verification and validation, requirements management, and requirements standards and tools. Issues such as stakeholder identification, risk analysis, trade off analysis as it relates to the requirements will be covered.

Appendix B: Course Project Problem Statement

Airport Management Information System: Currently, many small and medium-sized airports operate using traditional procedures such as: self-reporting of aircraft data for billing purposes by the air carriers, manual updates on flight information, and long-term lease agreements for gates and ticket counters. Self-reporting billing is imprecise and puts additional labor strains on air carriers that try to avoid or are in the process of recovering from economic hardships. The self-reporting procedure is utilized by air carriers to report gate usage and landed weights to the airport management. Air carriers may unintentionally forget to report data, or report late or inaccurate data. The understaffed and overworked personnel of air carriers are also responsible for updating the arrivals/departures flight information displays that can be found in any airport, called Multiple User Flight Information Displays (MUFIDS). This task, on occasion, leads to inadvertent misinformation. A potential opportunity for change is in the current lease structure. Gates and ticket counters are currently on long-term leases with the air carriers allowing the opportunity to maximize their branding at the gates and ticket counters, which limits the airport growth and profitability. Gates that are under leases cannot be utilized to bring new air carriers to the airport. This dilemma also extends to the ticket counters. If there are gates for flights but no ticket counters, then delays occur within the system. Dynamic flexible systems are needed to automate certain airport functions to overcome long delays and facilitate increases in the volume of passengers. As of 2007, some small and medium-sized airports started a series of tests for switching to a per-use policy system utilizing two gates. If fully implemented, this policy will eventually lead to the airport controlling all of the gates and ticket counters without long-term leases to air carriers. In view of the potential implementation of the new policy system, the airport management has identified the need for an automated system to manage the fees to be collected from the air carriers. The new automated system should be able to track these fees throughout the system, and give real-time updates on the financial transactions between the airport and airlines. Moreover, current operations at the airport terminal also use the error-prone manually update process for the arrival/departures information displays. The new system needs to be able to automatically update the MUFIDS without human intervention, thus providing real-time information on flights to air travelers in the terminal.