Engineering Project Management Graduate Education in Integrated Software and Systems Engineering Environments

Dr. 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.

Engineering Project Management Graduate Education in Integrated Software and Systems Engineering Environments

Abstract

During the last academic year, the Electrical, Computer, Software, and Systems Engineering Department at Embry-Riddle Aeronautical University started an innovative effort to integrate a series of graduate systems engineering and software engineering courses. The advent of embedded systems and cyber-physical systems, as integration of computation, networking and physical systems and processes, and, consequently, the goal to impart the needed knowledge to design and lead the operations for such systems, is one motivation. The other significant drive to pursue the software and systems engineering integration project is the ubiquitous use of integrated engineering and software systems of all sizes and complexity levels in the recent years. This work reports on the preparation, teaching pedagogy, class experience, and evaluation of the accomplishment of the course learning objectives for the combined offering of the "*Engineering Project Management*" and "*Software Project Management*" courses. Given the fact that the large majority of the systems built today include both hardware and software components, the combined offering emphasizes the benefits gained by the graduate engineering students when exposed to an integrated software and systems engineering development effort, characterized by contractually-defined system performance, risk, cost, and delivery schedule. As such, the active learning teaching practice implemented in the combined project management offering is the continuous student exposure to a mimicked integrated development environment resembling their future work positions, in which students ought to provide system-optimized solutions, rather than component-best solutions, within the performance-risk-cost-schedule framework.

1. Introduction

During the last academic year, the Electrical, Computer, Software, and Systems Engineering Department at Embry-Riddle Aeronautical University started an innovative effort to integrate a series of graduate systems engineering and software engineering courses. The advent of embedded systems and cyber-physical systems, as integration of computation, networking and physical systems and processes, and, consequently, the goal to impart the needed knowledge to design and lead the operations for such systems, is one motivation. The other significant drive to pursue the software and systems engineering integration project is the ubiquitous use of integrated engineering and software systems of all sizes and complexity levels in the recent years[1]. Overall, the academic institutions responded with a significant increase in the number of systems engineering graduate programs[2].

Teaching *Requirements Engineering* in integrated software and systems engineering classes was already reported by our engineering department in a previous work. This new work reports on the preparation, teaching pedagogy, class experience, and evaluation of the accomplishment of the course learning objectives for the second offering of the four-sequence of integrated software and systems engineering courses: the combined offering of the "*Engineering Project Management*" and "*Software Project Management*" courses. The first course is a requirement for the master's program in Electrical and Computer Engineering, while the second one is a requirement for the students enrolled in the graduate Software Engineering program. Given the fact that the large majority of the systems built today include both hardware and software components, the combined offering emphasizes the benefits gained by the graduate engineering students when exposed to an integrated software and systems engineering development effort, characterized by contractually-defined system performance, risk, cost, and delivery schedule. As such, the active learning teaching practice implemented in the combined project management offering is the continuous student exposure to a mimicked integrated development environment resembling their future work positions, in which students ought to provide system-optimized solutions, rather than component-best solutions, within the performance-risk-cost-schedule framework.

The current report on the *Engineering and Software Project Management* combined offering includes the status update of the overall software and systems engineering integration effort, the easily-observable similarities of the engineering and software project management domains, as well as the instructor's approach to accommodate the inherent differences of these two project management areas. In addition, the instructor reports on the teaching pedagogy followed and the specific lessons learned during the course of the combined offering. Ultimately, the combined engineering and software project management offering is designed and implemented with the objective of offering the graduate engineering students the necessary knowledge and training to successfully serve in project manager positions for large-scale integrated engineering design-software development projects.

2. Background and Motivation

The motivation for implementing the "software and systems education integration project" at Embry-Riddle is twofold[3]:

- 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.

The success of the first year combined software and systems engineering courses prompted the faculty and the administration at our institution to continue the integration effort armed with the lessons learned and the student feedback received in the first year. The second year of the graduate systems engineering and software engineering integration effort features the following combined offerings:

- A combined section of the "*Software Requirements Engineering*" and "*System Requirements Analysis and Modeling*" courses in the Fall 2014 semester.

- A combined section of the "*Software Project Management*" and the "*Engineering Project Management*" courses in the Spring 2015 semester.

An update on the "software and systems education integration project" at Embry-Riddle and its association with the graduate engineering curriculum is presented in Table 1. The combined *System and Software Quality Assurance* course taught during the Spring 2014 semester had only software engineering students registered, so the course was taught using the traditional software engineering framework. A subsequent offering of the combined section may provide the opportunity to evaluate the software and systems integration for that subject as well. The "*Model-Based Systems Engineering*," the proposed name for the systems engineering counter-part course of the "*Model-Based Verification of Software*" is still under development. As the proposed instructor for the MBSE course already presents the Systems Modeling Language (SysML) in several of the classes taught, the addition of the "*Model-Based Systems Engineering*" course is expected to enhance the understanding of the course material in all the other systems engineering courses, offered either in combined or separate sections.

3. Software and Systems Integration Framework for Teaching Project Management

3.1 Course Preparation. As for the preparation of the combined software and system requirements engineering course, the instructor consulted several reference materials covering both the engineering project management and the software project management domains. Once again, the instructor confronted the reality of unavailability of a definitive text covering both domains, so the course materials are prepared based on a combination of text resources which are supplemented with general knowledge resources such as:

- PMI Project Management Body of Knowledge[4]
- INCOSE Systems Engineering Handbook[5]

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

Integration status	Systems Engineering courses	Curriculum requirement	Software Engineering courses	Curriculum requirement	Up to date offerings
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 Fall 2014
	Engineering Project Management	E - MS SysE R - MS ECE	*Software Project Management*	R - M SE	Spr. 2014 Spr. 2015
	System Quality Assurance	R - MS SysE E - MS ECE	*Software Quality Engineering and Assurance*	E - M SE	Spr. 2014[7]
	Model-Based Systems Engineering[6]	E - MS SysE E - MS ECE	*Model-Based Verification of Software*	E - M SE	TBD
Integration status	Systems Engineering courses	Curriculum requirement	Software Engineering courses	Curriculum requirement	Up to date offerings
Separate sections	*Fundamentals of Systems Engineering*	R - MS SysE R - MS ECE	*Software Engineering Discipline*	R - M SE	Every semester
	System Architecture Design and Modeling	R - MS SysE E - MS ECE	*Software Systems Architecture and Design*	R - M SE	Fall semesters
	System Safety and Certification	E - MS SysE R - MS ECE	*Software Safety*	E - M SE	Spring semesters

[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; [7]Software engineering only offering.

- NASA Systems Engineering Handbook[6]
- Systems Engineering Body of Knowledge (SEBoK)[7]
- Software Engineering Body of Knowledge (SWEBoK)[8]

The course materials are designed such that students gain an in-depth understanding of both the engineering project management and software project management areas. Considering the overarching aeronautical theme of our institution, the course also covers the application of the engineering and software project management tools and techniques to the aeronautics and aviation domains. Regardless of the registered students' software or computer engineering background, the project management process is fairly similar for both software and hardware systems development, so the need to reconcile the two views is not as acute as in the case of requirements engineering. The course specific learning goals are to provide the students with:

- An understanding of the steps of the engineering and software project management processes.
- An exposure to the quantitative engineering and software project management techniques.
- An understanding of the application domains of engineering and software project management.
- The overall knowledge needed to successfully serve as project manager for complex engineering and software projects.

3.2 Similarities of Software and Systems Engineering Project Management. The tentative course schedule of the combined course syllabus included the topics expected to be covered in an engineering project management course, such as:

- Project Proposal, Selection, and Planning
- Project Roles and Organizational Structure
- Project Cost Estimating and Budgeting
- Project Activity Scheduling
- Project Resource Allocation
- Project Quality Management
- Project Risk Management
- Project Execution, Control, and Evaluation

These topics translate in well-defined course learning outcomes that are generic for both software background students and computer engineering background students. The instructor preparation to cover both the management of the software and hardware engineering development projects is minimal from the integration project point of view, as the management processes are practically the same. There are also some differences which will be highlighted in the next section. At the completion of the course, the registered students will be able to:

- Apply quantitative and qualitative techniques for the project selection process.
- Design and select appropriate project organizational structures.
- Develop sound project plans, including realistic project schedules.
- Prepare and justify economically feasible project budgets.
- Formulate and solve the resource allocation problem among one or more projects.

- Address project quality and project risk processes during the planning and execution stages.
- Monitor, control, and evaluate a project progress based on established performance measures.
- Identify and account for the characteristics of aeronautics and aviation project management.

3.3 Identifying Differences of Software and Systems Project Management Processes. The fact that software products are intangible and hardware products exhibit physical form does not make any difference in the process of managing the development projects. The difference between the development processes of software and hardware products comes from the actual view towards the deployment of the two. Most software systems (with safety-critical systems as a notable exception) are developed on a accelerated development cycle with the purpose of getting to market in the shortest amount of time, which comes with the advantage of reduced development time, but, at the same time, with the disadvantage of getting to market with products that may include a sizeable amount of errors. Now, the intangible characteristic of software makes the corrections of those errors easy during the operational stages, so the initial risk of higher number of errors introduced by an accelerated development cycle is counter-balanced by the opportunities given by getting to market in a short amount of time.

Besides the traditional software development processes, such as waterfall model, V-model, spiral model, and even prototyping/evolutionary models, which are all also used in systems engineering, software development is also conducted based on agile and extreme programming methodologies (Scrum, RAD, Kanban), which significantly reduce the development time[9]. Those agile development methods are studied in detail in the graduate "*Software Engineering Discipline*" course. The combined project management offering presents to all students the project management implications of the accelerated development cycle process. Students entering the course with only an engineering background also benefit from the presentation of the management of the accelerated development cycle projects as there are striking similarities with the product design and development when rapid prototyping and the more recent 3-D printing technologies are employed in the design and development process.

3.4 Software and Systems Project Management Teaching Pedagogy. Both the "*Engineering Project Management*" and "*Software Project Management*" courses are introductory graduate course designed for students coming from different engineering disciplines, such as Electrical and Computer Engineering, and Software Engineering. In addition, many times the courses are populated with students coming from Mechanical Engineering and Aerospace Engineering graduate programs. There are certain aspects of the courses that traditionally make the combined section more of an abstract, dry, course rather than an engineering hands-on course. First, the combined course covers the entire systems and software engineering development management process, which is a vast area, and does that in a fast-paced manner, leaving not much room for student experimentation with the techniques presented in class. Secondly, the topics covered are generic, as they apply for multiple engineering disciplines and the instructor needs to talk about "*Activity A*" and "*Activity B*" when presenting the modeling techniques rather than the "design the propulsion system" and "design the software control system." Having said that, the instructor developed the course learning outcomes in such a manner to provide the students with the foundations of the engineering project management processes such that they are able to develop and implement management process solutions to today's complex engineering design and operational problems.

In addition to the software and engineering project management course learning outcomes presented in Section 3.2 above, during the current offering of the Spring 2015 semester, the instructor introduced an overarching theme for the learning outcomes of the combined offering and enhanced the learning environment. The active learning teaching pedagogy adopted during the Spring 2015 offering translates into the following overarching learning outcome that the students should meet by the end of the course:

- Understand the essential steps of the systems and software engineering program management; and, be able to develop and follow them for a new design.

The active learning techniques to be implemented in the course are intended to address the first part of the above overarching outcome statement (the "understand…" part), such that, by the end of the course, the students succeed to "master" it, which is what is meant by the second part of the outcome statement (the "be able to…" part). Following this learning framework, the students can better apply their knowledge and skills when faced with real-world projects. Examples of active learning techniques to be implemented are presented next:

- Student exposure to a mimicked integrated development environment resembling their future work positions, in which students ought to provide system-optimized solutions, rather than component-best solutions, within the systems engineering "performance-risk-cost-schedule" framework.
- Student immersion into new experiences, such as project team work on complete systems and software development management projects, rather than traditional engineering discipline component development projects. The Kolb experiential learning framework[10] will be used as model. The experiential learning framework comprises 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.
- Student opportunity to critique the development project of other project teams expected to result in a critical-thinking debate on key aspects such as modeling methodology, solution approach, and performance-cost trade-offs.
- Student weekly engagement in their own learning through in-advance reading assignments of the lecture materials, which is expected to reduce the dry presentation of lecture materials and increase the time for lively interaction and debates on modeling techniques, development solutions, and their performance evaluation, and also gain some valuable time to link the generic software and engineering project management theory to real-world examples from all student-represented engineering disciplines.

Research in education has proved that active learning techniques can successfully address the limitations of both individualistic and competitive learning, and results in improving academic achievement, interpersonal relationships, increased student self-esteem, and student retention in academic programs[11]. Nevertheless, given the volume of the material to be covered and the sometimes "abstract" aspects of it, there are also challenges to be overcome as the instructor progresses towards the active learning-based environment. The major challenge identified so far is that active learning implementation is time consuming. The active debates can be longer than expected due to the different positions argued and defended by the students who certainly prefer such debates. This comes, as expected, at the expense of the amount of material covered. Another challenge identified is the student personality barrier, as not all students enjoy the immersion in active learning environments, a part of them preferring to just listen rather than actively participating. The instructor believes that the implementation of active learning will be beneficial for all the students and is willing to design the techniques such that student-response to the proposed class activities will follow an increasing trend. Also, the instructor is willing to work individually with students on a case-by-case basis to raise the awareness of active performing environments. Eventually, all students will be asked to perform in an active work environment after their graduation. Those who are currently not comfortable with performing in active (learning) environments will be forced to adjust, so it is the duty of engineering faculty to ensure that the students graduate with the understanding of the need for and the benefits of active learning and working environments.

4. Lessons Learned and Continuous Improvement for the Software and Engineering Project Management Courses Integration

The end-of-the-semester student evaluation of the course after the first offering in the Spring 2014 semester revealed that the students were generally favorable of the instruction received in the combined software and engineering project management course. As this work reports on the performance of the software and engineering project management integration, the instructor selected students' assessment related to the achievement of the course learning outcomes. Fig. 1 presents that assessment graphically.

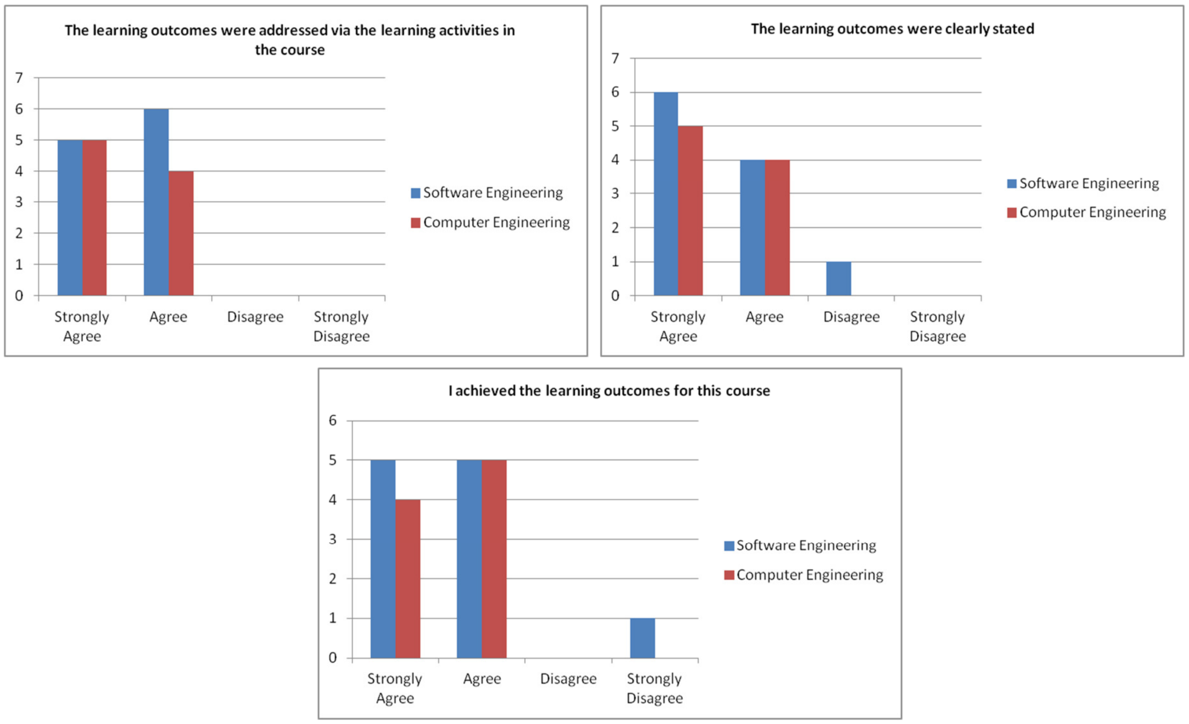

Fig. 1: Student evaluation of the course learning outcomes-related questions

Using the anonymous written feedback, one student noted that: "*It was (a) really good class and (I) learned a great deal from it as I am having much interest in this field… it helped me a lot understanding the concepts which I can use in my future work.*" A majority of 55% of the enrolled students selected the *Strongly Agreed* option, and another 40% of them selected the *Agreed* option when asked to identify themselves with the following statement: "The learning outcomes were clearly stated." Nevertheless, a minority of only one student disagreed with the statement. In terms of the instructor's performance on course development directly related to the course learning outcomes, captured by the following statement "The learning outcomes were addressed via the learning activities in the course," the students identified in equal proportions to both the *Strongly Agreed* and the *Agreed* options. For what was considered the most important question of the questionnaire, namely, "I achieved the learning outcomes for this course," 95% of the students responded with either the *Strongly Agreed* or the *Agreed* options. There was one answer in disagreement with the question, which could be just an outlier, but in any case prompted the instructor to more carefully present, address, and evaluate the learning outcomes during the Spring 2015 offering.

The active learning techniques introduced in the current offering, and presented in Section 3.4, are the result of the instructor's evaluation of the student accomplishment of the course learning outcomes during the first offering in the Spring 2014 semester. In addition, for the Spring 2015 semester, the instructor introduced beginning- and end-of-the semester survey questionnaires that intend to evaluate the before and after the course understanding of the need and value of the "software and systems engineering education project." The students are asked to answer, with yes/no/not sure type of answers, two questions using their current understanding of the necessary integration process of engineering hardware and software and the perceived benefits of this integration in relation to their future careers, as follows:

- The initiative to integrate *Software Project Management* and *Engineering Project Management* will help/helped me better understand the overall project management process as it applies to today's high-tech integrated hardware and software systems.
- As a computer engineering (hardware) graduate student, the exposure to the software engineering development methods and tools, such as the software project management process, is valuable for my future career.
- As a software engineering graduate student, the exposure to the engineering (hardware) development methods and tools, such as the engineering project management process, is valuable for my future career.

The evaluation of these survey questionnaires will be reported in a future presentation and paper submission. The same type of survey questionnaires are intended to be implemented in all other combined software and systems engineering offerings.

References

1. Badiru, A. B., *Project Management: Systems, Principles, and Applications*, Taylor & Francis Group, 2012.
2. Fabrycky, W. J., Systems engineering: Its emerging academic and professional attributes, *Proceedings of the ASEE Annual Conference and Exposition*, Louisville, KY, June 2010.

3. Babiceanu, R. F., A "Software and Systems" Integration Framework for Teaching Requirements Engineering, *Proceedings of the ASEE Annual Conference and Exposition*, Indianapolis, IN, June 2014.
4. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, Fifth Edition, Project Management Institute, 2013.
5. *INCOSE Systems Engineering Handbook v. 3.2,* INCOSE-TP-2003-002-03.2, International Council on Systems Engineering, 2010.
6. *NASA Systems Engineering Handbook*, NASA/SP-2007-6105 Rev1, National Aeronautics and Space Administration, 2007.
7. 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.
8. *Software Engineering Body of Knowledge (SWEBOK)* version 2, IEEE Computer Society, 2004.
9. Villafiorita, A., *Introduction to Software Project Management*, Taylor & Francis Group, 2014.
10. Kolb, D. A., *Experiential Learning: Experience as a Source of Learning and Development*, Prentice Hall, Inc., 1983.
11. Prince, M., Does Active Learning Work? A Review of the Research, *Journal of Engineering Education*, 93(3), pp. 223-231, 2004.