Adding Software Engineering Emphasis to an ECE curriculum

Mr. Pat Smith, Oklahoma Christian University of Science and Arts

Mr. Pat Smith lives in Edmond Oklahoma. He has been married for 32 years to Dr. Virginia Smith, a leading expert and author in the field of higher education. He has a son who is a physician and daughter who teaches physics and chemistry. Mr. Smith enjoys long distance road cycling and board games. He gained his undergraduate degree at the University of Oklahoma in Computer Engineering in 1984. Following this he worked in industry for Conoco Inc, Du Pont, and Cisco Systems. At Cisco Systems he worked with Cisco customers designing and deploying core Internet designs and technologies. In 2005 Mr. Smith earned a master's degree in Computer Science from Colorado State and 1 year later left industry to teach engineering at Oklahoma Christian University. Mr. Smith's emphasis is in first year student success, mentoring young engineers, and data communications. He consistently scores well in student feedback and enjoys regular strong relationships with his students and classes.

Adding Software Engineering Emphasis to an ECE curriculum

Abstract

This paper describes a project conducted at Oklahoma Christian (OC) University to add software engineering emphasis to an existing Electrical Engineering (EE) and Computer Engineering (CompE) Program. Over a 2 year period the OC Electrical and Computer Engineering (ECE) department received input from alumnae, our ABET Industry Advisory Board, and other stakeholders that indicated there is an increased value to EE and CompE graduates for having greater knowledge in areas related to software engineering. In response two faculty members from the ECE department conducted a survey of alumnae and local engineering industry to gauge the need and determine what software engineering topics should be emphasized in an undergraduate ECE program to prepare EE and CompE students for entry level work involving high levels of software engineering responsibilities. The investigation confirmed the need for added software engineering topics and provided insights which guided topic selection. The results of that investigation are presented here. The OC ECE department modified EE and CompE curriculum to add specific software engineering topics to existing courses in order to create a logical progression of software engineering education through a series of courses. Several courses now contain relevant additional software content to prepare students for entry-level engineering work. This content is coordinated across the curriculum to build from the junior through senior courses and makes use of required and elective courses. These are now formally identified together and, if taken together, provide an emphasis in the software engineering topics that were requested by local engineering industry. The changes to curriculum are presented in this paper. OC's Electrical and Computer engineering students can now graduate with an emphasis in software engineering and will be better prepared for the high degree of software related engineering work they are likely to face as they begin their careers.

Initial Motivation

During the time period from 2011 – 2013 the OC Electrical and Computer Engineering (ECE) department received input from multiple sources indicating that software engineering activities were becoming a greater portion of job responsibilities expected in jobs filled by both electrical and computer engineering graduates.

Our graduates were providing this feedback to us in informal alumnae and faculty meetings taking place at events such as homecoming or local society meetings. Stakeholders such as local industry representatives who meet with the faculty as part of our ABET advisory board also communicated the growing demand for graduates with stronger software engineering skills.

The most significant event took place during the summer of 2012. Several local employers including Boeing, Northrup Grummond, and local energy and utility companies called a meeting with the departmental chairs of regional universities to discuss industry's need for higher volume of electrical and computer engineers – as well as the needed skill sets for those graduates.

Industry representatives shared that the most pressing need was for greater software engineering skillsets in both electrical and computer engineering graduates. This feedback specifically requested that regional university Electrical and Computer Engineering departments strengthen their software engineering content areas.

Why, if these employers need more software engineering skills in their employees, didn't they hire graduates of computer science or software engineering degree programs?

These companies engineering functions are involved in developing complex systems including both hardware and software. Although many are involved in the aeronautical industry, the problem they describe is similar to that described by McDonald [1] in reference to software engineers developing motor control algorithms – that is, in the case of motor control algorithm designers they require an understanding of dynamics and electro-mechanical systems. This is consistent with the definition cited by Fairley [2] of a system being a “complex arrangements of diverse hardware, software, and people elements that include computing hardware and software.” What regional employers were telling us was that they need engineers who can understand the full system including both electrical subsystems and software subsystems.

One company, Flight Safety (based in Tulsa, Oklahoma) addressed this question in a way that found consensus among the industry representatives present. Flight Safety stated that they implemented software systems which emulate complex electrical and computer systems in the airplanes they for which they design full-size cockpit simulators. They need engineers who understand the underlying electrical and computer systems and can then implement simulations of those complex systems in large software and hardware integrated simulations. This sentiment was affirmed by the other company representatives present.

The accumulated weight of this consistent feedback and input from multiple sources motivated the faculty of the ECE department to consider how more software engineering content could be included in the program.

Several questions needed to be addressed.

- How serious was the need?
- What knowledge and skills were most important, especially in the skill set of a recent graduate?
- Was the desired skillset uniform across local industry or was there considerable variance from one employer to another?
- Where in our program could we add content? Would we be adding courses, modifying existing courses, or some combination?

A two phase project was defined and included in our ABET improvement project process. The process was similar to the process defined by Sticklen et al [3] to gather input from industry and implement in a local program. The first phase would involve an assessment of needs as defined by local industry stakeholders. The second phase would be a process of deciding how to integrate this into our curriculum.

Requirements Collection and Verification

Our first step was to create a survey to serve as a tool for gathering input. Constituents including local industry representatives and alumnae were invited to complete the survey, ranking software engineering topics and skills in terms of “desired skill in new graduates”.

The survey results would then be vetted in a second qualitative step through a committee of alumnae engineers from local companies seeking stronger software engineering skills in new graduates. The committee would review the survey results and provide clarification and guidance around priorities revealed by the survey tool.

Two faculty members, Professors Ken Bell and Pat Smith, volunteered to lead the requirements collection and verification project.

A survey tool covering all areas of software engineering skills and knowledge was needed. Initially several texts were considered as sources along with collecting information from existing coursework topics covered in software engineering programs. The results of these early attempts to create a list of topics to include in a survey were unsatisfying. We were uncertain whether or not we had a complete list.

It was at this time that Mr. Bell suggested we use the IEEE Software Engineering Body of Knowledge (SWEBOK [4]) book for a comprehensive list of topic areas. The SWEBOK provided a complete list of knowledge and skill areas for inclusion in the survey. In addition, it provided a structured organization of these skill/knowledge areas and enabled the survey to be designed in a way that grouped questions into knowledge areas.

The SWEBOK has 15 high level topics and 78 subtopics. A survey was developed that asked participants to provide feedback on a Likert scale. Survey participants were asked to identify the level of expertise expected in an entry level engineer for each subtopic in the SWEBOK. The survey specifically asked participants to rate the level of knowledge desired in an engineer who has just graduated from college. The five levels of expertise were: “Not Expected”, “General awareness”, “Detailed Awareness”, “Practical Experience”, “and Experience with specific tools or techniques”.

We also wanted to know if there were specific tools, languages, or other unique requirements related to the topic. A space was provided for the survey participant to provide such feedback.

The survey’s initial instruction was as follows:

“The survey is divided into major topics. Each topic has a list of subtopics. For each subtopic please place a mark in the column under the priority you feel is most appropriate for an entry level engineer who will be doing software development.”

In figure 1 (below) you can see a sample from the first topic area from the survey. The entire survey is available from the author if desired.

Topics		Not Expected	General awareness	Detailed awareness	Practical experience	Experience with specific tools or techniques
Software Requirements – this is concerned with the elicitation, analysis, specification and validation of software requirements. This includes understanding of :						
a.	The requirements process – How are the activities associated with software requirements are organized for different projects and constraints.	1	8	1	1	
b.	Methods for eliciting requirements	1	5	4	1	
c.	Analysis of requirements	1	1	1	2	
d.	Specifying requirements clearly	1	4	1	5	1
e.	Validating requirements with clients		6	2	3	

Figure 1 – Partial Survey

Survey Process

The OC ECE department has an Industry Advisory Board (IAB) which is made of representatives from local engineering firms. The board has over 30 members with responsibilities ranging from staff engineers to executive management. A special advisory board meeting was scheduled to seek input on this process and ask for assistance in reaching a broader audience for participation in the survey.

Two positive results came out of this meeting. First, the local industry representatives on the IAB were very enthusiastic and supportive of the process. Once again we were strongly encouraged to strengthen the software engineering skillsets of our graduates. Second, board members agreed to both participate in the survey and ‘sponsor’ the survey within their own companies, providing names and contact information for people who could provide feedback based on first-hand knowledge and experience.

A list was generated by our advisory board of contacts at the board members’ individual employers who would be appropriate for providing feedback through the survey tool. The survey was sent to 19 people. These people all had leadership roles in their organization. Their titles include Software Engineer Team Lead, Supervisor, and Associate Director. Thus they included both technical leaders and management.

Of the 19 people who received the survey 11 responded resulting in a 58% response rate. Five companies were surveyed and at least one person from each company responded, resulting in a 100% response rate across the group of companies who prompted this work through their meeting in 2012.

As second, qualitative step, was then taken using the results of the survey. The raw results were tabulated and provided, with some commentary and explanation of terms and word usage based

on SWEBOK's language, to a group of 8 alumnae who are activity involved primarily in software development. This group reviewed the results and provided additional commentary and input. This meeting did not change the fundamental results but it did provide valuable insight into the motivation behind the results.

The remainder of this paper discusses the results and how these are being implemented across OC's ECE program.

Survey Results

In the original survey there were 15 topic areas with 78 subtopics individually listed. The 15 broad topic areas are taken directly from the SWEBOK document. The Likert scale responses were assigned values from zero to four, with zero meaning employers had no expectation of expertise in this area and a score of four meaning employers expected graduates to have experience in the area including specific tools or techniques. The other scale responses can be seen in Figure 1 (above). In Figure 2 (below) the average score for each broad SWEBOK area is seen.

Figure 2: Survey Response Scores for Broad SWEBOK Topic Areas

The reader is reminded that these scores are indicating the employer's desire to see these skills in graduates. These do not represent the responder's view of the importance to their overall business or organization. For example, Software Engineering Management is rated second lowest in the survey results. The follow-up qualitative group that met shared with us that topics such as this and the Software Engineering process are important, but new graduates will probably not be hired to work in those areas – they are typically filled by experienced engineers and are thus weighted lower for new graduates.

The four most important topics were Mathematical Foundations, Engineering Foundations, Professional Practice, and Computing Foundations. These could be described under a broad umbrella category of “Engineering Fundamentals”. OC’s ECE program covers the “Engineering Fundamentals” topics thoroughly throughout the curriculum as verified by OC’s ABET Feedback and employer feedback surveys (not included in this document due to space limitations). This broad area includes topics such as programming languages, basic debugging, problem solving, algorithms, mathematical skills (rated highly), standards, and root cause analysis.

The survey confirmed to the department that the current inclusion of these topics is important and we should not reduce these at the cost of other topics. In our follow up work we took care to ensure that these topics continued to be a strong part of our degree plan.

A second group of topics that were ranked closely together could be collected under the broad category of “Software Design, construction, and testing”. Upon closer examination of our degree plan, and with consultation with our qualitative follow-up group it was determined that our program should offer the opportunity for students to have greater exposure and education in these areas. Essentially, OC’s ECE department had room for improvement in the area of “Software Design, construction, and testing”. The survey suggested we improve these should add content around these broad topic areas in the following order of importance:

1. Code Generation
2. Software testing
3. Requirements Elicitation

The remaining major SWEBOK topics from SWEBOK that were ranked lower in priority in our survey results and were not considered for additional inclusion within the ECE degree plans.

Final results (priorities)

As noted above, a follow-up meeting was arranged to gather qualitative input on the survey results. In that meeting we focused on the areas that were rated highly in the survey results but were areas we could readily see were not addressed in our curriculum. In the area of Code Generation, Unit Testing, and Requirements Elicitation we were able to identify subtopics for content to augment or add across the ECE curriculum. These are identified and discussed in more detail below.

CODE GENERATION

Under the general topic of code generation there are several specific topics to be covered. This include version control, working with existing code bases, using libraries, debugging code (especially code developed by someone else), and dealing with maintenance issues (changing code base).

Version Control:

Version control was considered to be of high importance. All of our respondents emphasized what an important tool and concept this is, and noted that it should integrated across the

curriculum if possible. It was suggested that the topic be introduced as a concept, used on an individual basis later, and finally used in a team setting (either on new large code base or to access an existing code base). This topic was not covered in our curriculum except in the CS course “Software Tools”, which our engineering students cannot typically fit into their schedule.

Working with existing code

While it is a challenge to approach this in a class setting, the results indicate that it would be helpful if students could gain experience at working with code they didn’t develop, and with bodies of code larger than they construct themselves for a single assignment. The suggestion was made that perhaps existing code (such as open software foundation projects) could be used as a code base and a project to add a feature could be assigned. This would give experience in this topic area.

Using Libraries

Students need to have experience at using software libraries. BOOST (boost.org) was cited as a common example. Undergraduates typically develop code based on integrated libraries in the C++ or C# development environment. EE and CompE graduates should know how to use and integrate libraries other code sources.

Debugging Code

Students get good training at debugging code they write in our curriculum, but it was stated that the more common situation is that programmers must debug someone else’s code. It was strongly encouraged that students should gain experience with this.

QUALITY AND MAINTENANCE

Students do not clearly understand the issues related to writing code that can be maintained, and writing quality code. This can partly be accomplished through education regarding software processes (agile methods particularly mentioned). We were also encouraged to develop educational situations where students would develop code in a project, and later receive new instructions to modify the code to meet new (different) requirements.

UNIT TESTING

The discussion of gaining testing experience focused on teaching topics related to methods and tools that enable systematic or automated testing. Our IC Design course contained an example of this with the vector testing that is taught (this was specifically mentioned by the group). However, feedback indicated that this needed significant strengthening.

REQUIREMENTS ELICITATION

Some knowledge in this area (UML, writing and reading requirements) was desired, but in general the recommendation from the group was to spend less time on this topic and more time on Code Generation and testing issues. The reasoning is that requirements development is typically done by senior engineers, and so it is seen as less important for an entry level position.

Summary of New Content Recommendations

The survey and qualitative feedback confirmed the need for foundational engineering knowledge and programming skills. The department already provided these throughout the curriculum. No additional work was required.

The topics listed above including code generation, unit testing, and requirements elicitation, should be strengthened if possible.

Once we had this information we proceeded to the more difficult question of where and how we could add this material into the curriculum without sacrificing other content considered important to our program's students.

Integration into OC's ECE curriculum

The faculty had already agreed to make a good faith attempt to strengthen the software engineering content across the curriculum. We now considered several questions:

- Should we add new coursework or attempt to augment existing coursework?
- Should all students be expected to study the material?
- Should we highlight the availability of the content in the coursework or in the curriculum?
- If an employer desires to hire an EE or CompE graduate who has emphasized software engineering topics how would they know that the graduate had actually focused on software engineering topics if there were not specific courses taken? In other words, if we "buried" the content in existing EE and CompE courses, who outside the program would know that the "Digital Signal Processing" course contained additional software engineering content?

A series of meetings of the entire ECE faculty were held. Initially the results were discussed and an open discussion of ideas and concerns was held. After much discussion the group developed three possible approaches:

Approach 1: Add electives to which cover software engineering topics. Our curriculum is very tight. Students choose two electives in their senior year. Given the volume of software engineering content desired it was believed that this option would require to a two-semester sequence to properly cover software engineering topics. Our concern was two-fold. First, who would take on additional coursework and how would they add it to existing load? Second, the faculty enjoy teaching their senior elective courses but sometimes low enrollment in an elective results in cancellation of that course. Would adding another, potentially popular elective sequence result in other electives being canceled as students migrate to the software engineering coursework? Finally, there was concern that if the software coursework was deemed by students to be "easier" than more challenging electives such as advanced electronics then many students would choose to take them simply to have easier courses.

Approach 2: Break the desired topics into smaller units and assign them to courses where similar material is present. Software engineering would be strengthened across the curriculum and all

students would benefit from this approach— though this would clearly be at the cost of other content and material in courses identified to add or augment software engineering related topics. Concerns around this included a question of whether all EE and CompE students should be exposed to this material regardless of their interest. Not all of our students are interested in this material or plan to follow a career where this work will be performed. Additionally, it was observed that in several cases the courses that would be augmented were electives. While this was viewed as a plus in that only those students interested in the material would enroll in those courses, it was also seen that the material would be “buried” in courses whose title didn’t clearly communicate that software engineering methods and skills were strongly present in the courses.

Approach 3: This is a hybrid of options 1 and 2. In this option a “Software Engineering Emphasis” track is identified. This track will be available to both EE and CompE students. Software engineering topics are added or strengthened in existing required and elective course. Students wishing to study software engineering choose this track and take a modified course plan. This was the approach ultimately chosen.

In this solution five courses are modified to contain more software engineering content. They are:

- Intro to Microprocessors. This course already contained a large software project based in assembly language. This will now be modified to contain both assembly and C-language programming. Topics around unit testing and version control will be added.
- Embedded Systems and HDL: both of these courses already contain content around strong testing methodologies and these topics will be strengthened.
- Real Time Signal Processing: This course is a senior EE elective. It contains considerable information in the area of requirements elicitation, use of UML in the design process, and implementation of DSP methods in real-time environments. In this course these topics will be strengthened and greater emphasis will be placed on them as real-time DSP solutions are developed and studied. This course is renamed “Software Engineering of Real Time Systems”.
- Network Systems: This course is a senior CompE elective. It implements a full data communications router on an android based platform as an approach to understanding issues related to design of network systems design. This course will now integrate advanced software engineering topics around version control, software development processes, and an introduction to patterns in software development. Experience with external libraries and existing code will also be included. This course will now be titled “Software and Network Engineering”.

In order to allow both EE and CompE students to emphasize software engineering a new EE and CompE degree plan is created. The new EE degree plan drops the CompE course Computer Systems and one senior EE elective, adding Data Communications and Software and Network Engineering. The new CompE degree plan drops the EE course Electronics and one senior elective. It adds Digital Signal processing and Real Time Signal Processing.

The following advantages are now realized:

1. Students who want to emphasize software engineering can do so but students who are not interested in this field of study are not required to follow this track.
2. Students who choose to follow the Software Engineering Track will be able to advertise this choice to potential employers. The change in title for the two senior electives taken by students following this track will communicate clearly the presence of software engineering related topics in the course of study.
3. The courses identified for inclusion in the Software Engineering Track are considered to be equally challenging, containing workload comparable to alternative tracks within the EE and CompE degree plan. It is not anticipated that students will choose this plan based on a perception of “easier coursework” or “lighter workload”. It is expected that students will choose this based on interest or career objectives.
4. Courses which are being augmented already contained some of the material desired by our stakeholders. Those faculty teaching these courses are able to see a path to strengthening this material without sacrificing core elements in these courses. Particularly in the EE and CompE senior electives both of these courses already used software to explore in-depth topics. These courses will restructure the software portion in a way that emphasizes software engineering approaches and tools in the study of the topic at hand. In this way software engineering can be augmented without sacrificing other core topical content.
5. This approach has as a final unintended benefit of balancing our EE and CompE junior course loading. While we will not know the impact on senior courses until the spring of 2017 the early numbers are promising a similar effect in the electives.

This approach, including a new degree plan for both EE and CompE majors, was approved by the university’s Academic Affairs committee in January, 2015 for implementation beginning in the fall, 2015 semester. This year’s junior class is the first group of students to have the option to choose this path. Out of xxx students, yyy students have indicated that they will choose the software engineering track.

Summary

Stakeholders in the greater Oklahoma City region approached Oklahoma Christian University and other regional universities such as Oklahoma University, Oklahoma State University, and others with a request we add more content related to software engineering in the EE and CompE programs. Oklahoma Christian University agreed to investigate the feasibility of adding software engineering content. A study indicated that basic skills were already well taught, but more content related to code generation, testing, and requirements elicitation should be taught to EE/CompE students who desired to pursue careers with software engineering responsibilities. After comparing the desired topics with our existing degree plans and course work we were able to devise a new track that modified both EE and CompE degree plans. These new plans modify the degree plan to enable EE and CompE majors to study software engineering. Students can now emphasize software engineering if desired. The changes will be re-evaluated as part of our standard ABET processes. The solution satisfies industry stakeholder requests, student desires, and faculty concerns.

The survey, results, and modified degree plans are available to interested readers by emailing pat.smith@oc.edu

References

- [1] D. McDonald, "Engineering And Technology Education For Electric Vehicle Development," in *Annual Conference & Exposition*, Louisville, Kentucky, 2010.
- [2] R. Fairley, "Teaching Software Engineering to Undergraduate System Engineering Students," in *2011 Annual Conference and Exposition*, Vancouver, BC, 2011.
- [3] J. & U.-L. M. & B. N. & W. T. & V. C. E. & C. T. & M. T. & D. C. & F. K. & B. D. & L. J. & P. L. Sticklen, "Leveraging Workforce Needs To Inform Curricular Change In Computing Education For Engineering: The Cspace Project," in *2009 Annual Conference & Exposition*, Austin, Texas, 2009.
- [4] IEEE, "Software Engineering Body Of Knowledge," IEEE, [Online]. Available: <https://www.computer.org/web/swebok/v3> .