An Analysis of the Software Engineering Curriculum Using the Guideline Models

Massood Towhidnejad, Omar Ochoa and Anton Kiselev
Embry-Riddle Aeronautical University

Abstract

The computing field is a very dynamic environment and as such, computing education must be able to respond to these changes. Modifications of the curriculum in an academic setting is a long and tedious process. By the time a curriculum modification is proposed, submitted through the approval process, and accepted, the industry needs have changed. Complicating this process further, there are multiple computing curriculum recommendations endorsed by the computing professional organizations that provide guidelines for curriculum design. The Software Engineering curriculum guideline (SE2014) is widely used by the educators for the design and modification of undergraduate software engineering programs. On the other hand, the IEEE Software Engineering Competency Model (SWECOM) provides a set of competencies that a software engineering professional should possess. In this paper, we present the approach, analysis and result of comparing the SE2014 curriculum, which guide the preparation of software engineers, against SWECOM, which represents the employer's needs.

Keywords

Competency Model; Software Engineering Curriculum Guideline; Software Engineering Education

Introduction

The computing field is a very dynamic environment, and as such, the industry needs to respond to these changes on a regular basis; however, computing curriculum cannot change as fast. There is a great workforce demand for people who have the appropriate knowledge and experience in the computing field. The industries that employ graduates of computing degree programs aim to hire those who are familiar with the latest technical traits, tools and methodologies, and in order to meet these needs, the computing curriculum needs to respond quickly to these needs. However, this is not a realistic expectation because modifications to the curriculum, to better serve the needs of industry, in an academic setting is a long and tedious process. By the time a curriculum modification is proposed, submitted through the approval process, and accepted, the industry needs could have changed. To further complicate this process, there are multiple computing curriculum recommendations that have been endorsed by the computing professional organizations that provide guidelines for curriculum design.

The Software Engineering (SE) curriculum guideline (SE2014) is widely used by the software engineering educator for the design and modification of undergraduate software engineering programs[1]. SE2014 defines the Software Engineering Education Knowledge (SEEK), which are set of knowledge areas. Each knowledge area is broken down into several units and each unit are further divided into a set of topics, which are the smallest level of knowledge that can be

attained. On the other hand, the IEEE Software Engineering Competency Model (SWECOM) presents a competency model that represents a set of competencies that a software engineering professional should possess and could potentially be used by educators to develop a software engineering curriculum that meets the needs of industry[2].

This paper represents our initial analysis of a study, where we treat the SE2014 as input to guide the curriculum development, and SWECOM as an expected characteristic of output of the curriculum (i.e., the graduates of the degree). For this analysis, we treated the curriculum as a gray box were the syllabi for the required courses were used in order to conduct the curriculum coverage of software engineering topics.

Background

SE2014 is an update to a previous effort, SE2004, whose main goal was to provide a curriculum guideline for software engineering education[1]. SE2014 describes three main key components that support desired outcomes for undergraduate students, foundational ideas and beliefs about software engineering and the goals for the curriculum. The main expected outcomes for students are:

- Professional knowledge: students must show mastery of software engineering knowledge and skills.
- Technical knowledge: student must demonstrate an understanding and apply appropriate theories, techniques, and models.
- Teamwork: student shall be able to work both, as individuals and as a part of a team.

The main SE2014 principles for students are:

- SE in the Computing Spectrum: student has to understand that computing is a broad field that extends beyond any computing discipline.
- Reference Disciplines: student has to understand that Software Engineering is founded based on a variety of disciplines.
- Curriculum Evolution: Software Engineering is always evolving in different aspect such as technology, applications, pedagogy and requires a review of the curriculum.

The main SE2014 goals for the guidelines are:

- International Relevance: SE2014 has to be approachable in an international scope.
- Range of Perspectives: SE2014 has to be broadly based.
- Professionalism: SE2014 has to include strategies and tactics for implementation with high-level recommendations.

SE2014 describes the main core for a software engineering curriculum, it defines a unit of time for each subject to be covered in curriculum. In SE2014 an “hour” corresponds to the time, in which the material must be presented to a student, and it is presented in a traditional lecture-oriented format, such as a 50-minute lecture. There are ten main knowledge areas defined for software engineering, such as computing essentials, mathematical and engineering fundamentals, professional practice, software modeling and analysis, requirements analysis and specification, software design, software verification and validation, software process, software quality, and security. Each knowledge area contains specific topic. For each, the SE2104 specifies one of the skill levels:

- Knowledge (k): Remembering previously learned the material.
- Comprehension (c): Understanding information and the meaning of material presented.
- Application (a): Using learned material in new and concrete situations.

Each topic in each knowledge has its own relevance and it is assigned as follows:

- Essential (E): Topics that are part of the core.
- Desirable (D): Topics that are recommended to be included in the core of a specific program if possible; otherwise, it should be considered as a part of elective materials

In 2014, the IEEE Computer Society published the Software Engineering Competency Model (SWECOM) that provides a competency model framework that identifies activities that software engineers will participate in throughout their career[2]. SWECOM defines competency in the form of skill areas, skills and the set of activities associated with those skills. There are five competency levels defined by SWECOM, these are (technician, entry level, practitioner, technical leader, and senior software engineer). While developing SWECOM the authors took into consideration the existing competency models such as the engineering competency model[3], information technology[4], systems engineering[5], and software assurance[6]. In addition to the above competency models, the Software Engineering Body of Knowledge (SWEBOK)[7], ISO/IEEE Standard 12207 (software engineering processes)[8], Graduate Software Engineering Curriculum Guidelines (GswE2009)[9], and Undergraduate Software Engineering Curriculum Guidelines (SE2004)[10] had major influence on the development of the SWECOM.

The primary focus of SWECOM is in the technical skills of the software engineer. The SWECOM framework is based on thirteen different skill areas, five skill areas directly related to the different software development phases, these are; requirements, design, construction, testing, and sustainment. The remaining eight skill areas are cross-cutting across all phases. These are; process and life cycle, systems engineering, quality, configuration management, security, safety, measurement, and human computer interaction. Each skill area is comprised of number of skills, and number of activities defines each skill. Almost all these activities are related to the technical capability of the individual.

In addition to the skill areas, skills and activities, SWECOM identifies five levels of competencies. The competency level represents the extent in which a person can perform a specific task. The following is the list of the competency levels with the associated details:

- Technician, an individual who is competent to follow instructions while performing an activity.
- Entry Level Practitioner, an individual who is competent to assist in performing of an activity or performs an activity with supervision.
- Practitioner, an individual who is competent to perform an activity with little or no supervision.
- Technical Leader, an individual who is competent to lead and direct individuals and teams in the performance of activities in a skill or skill area.
- Senior Software Engineer, an individual who is competent to develop new or modify existing policies, procedures, methods, tools and guidelines for the technical activities and work products within an organizational unit that is engaged in software engineering.

In addition to the competency level, the framework includes additional notations to distinguish the role of a person in a given competency level. These roles include; Following (F), Assisting (A), Performing (P), Leading (L) and Creating (C).

Previous Study

Previous work presented an initial result surveying four universities that offer an undergraduate software engineering degree[11]. A survey was created based on the structure of SWECOM. The survey asked participants to compare SWECOM skills and activities against their curriculum. It is important to point out that the representatives of the software engineering programs completed the survey; therefore, the results presented were a self-evaluation of the program. The results of the survey showed that SWECOM is a good starting point for the software engineering competency, and it is obvious that universities are enabling their students to gain the knowledge that is needed to perform tasks that are associated with the different skills. The results of the analysis of the survey showed that there may be a need to either adjust the expected level of competencies that are identified by SWECOM or make appropriate modifications the curriculum offering to ensure that the graduates of SE programs are well prepared to enter the workforce.

Approach

The first step in this analysis was to conduct a comparison between SE2014 and SWECOM to identify any similarities. A table was created capturing the intersection between SWECOM skill areas with skill sets, and SE2014 knowledge areas. Then it was decided to apply this to the software engineering curriculum at Embry-Riddle Aeronautical University (ERAU). The skills that were identified as the intersection between both guidelines were used to guide in the selection of what software engineering degree classes were to be used in the study. The course syllabi for these classes were examined in order to identify the topics and activities that are covered in these courses.

Thus, the syllabi were used to identify what topics are covered in the course and how many hours were spent on each topic by dividing the topics over the hours the course had. In addition, since SWECOM does not prescribe the knowledge level or years of experience with these competency levels, the assumption was made that undergraduate students graduates from a software engineering program is at the Entry Level Practitioner level. The resulting data was plotted into a radar graph to facilitate the understanding of the data.

Results

This section provides the results of the analysis for each skill area for ERAU. Each level refers to a specific value of attainment such as 0 for a no attainment, 1 is a Technician level, 2 is an Entry Level Practitioner and 3 is a Practitioner level. Each graph shows the comparison of assessed level of required attainment by SE2014 and by ERAU compared to the assumed SWECOM attainment, which is set at the constant level of 2 (Practitioner).

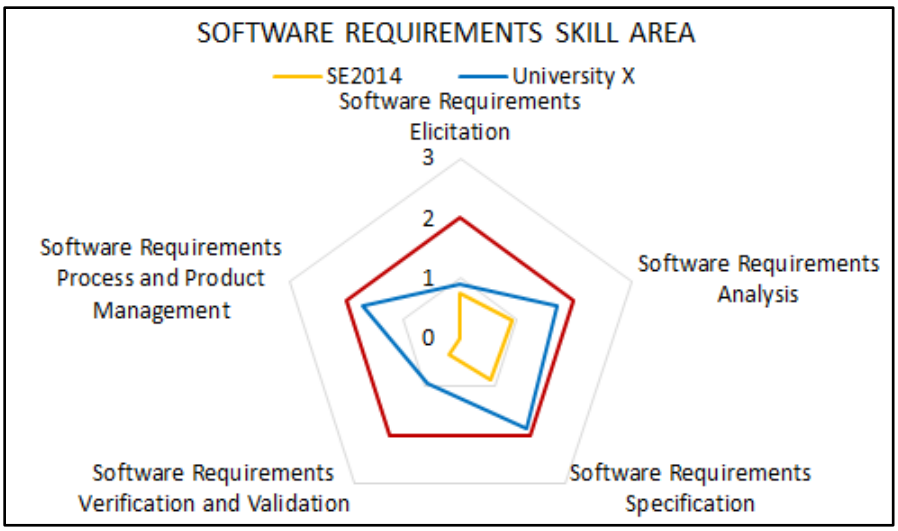

Figure 1. Gray-box analysis for Software Requirements.

Fig. 1 shows that in the Software Requirements skill area, as it is shown, the SE2014 recommendation is not enough to meet SWECOM's recommendations, and only meeting four of the five skills at the Technician level. Likewise, ERAU, although doing better than SE2014, is also not meeting SWECOM's recommendations in two of the five skill sets. It is worth noting that SE2014 required knowledge attainment for the Software Requirements area is significantly smaller than what SWECOM requires.

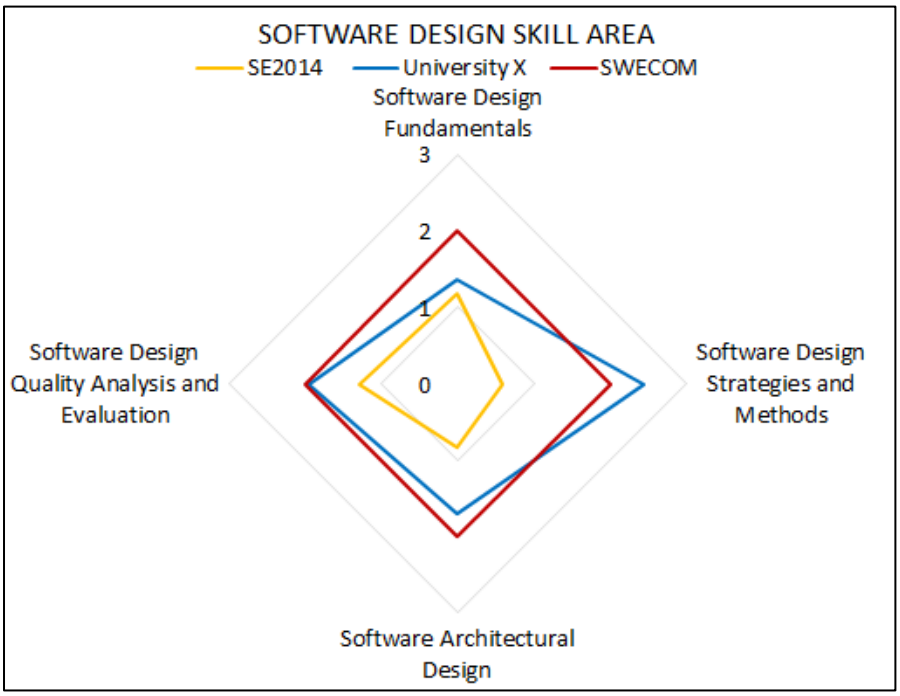

Figure 2. Gray-box analysis for Software Design.

Fig. 2 shows that SWECOM requires a higher attainment level of knowledge in the Software Design skill area. Like the previous skill area, SE2014 requires less knowledge attainment than SWECOM and the analysis shows that ERAU is closer to requirements of SWECOM.

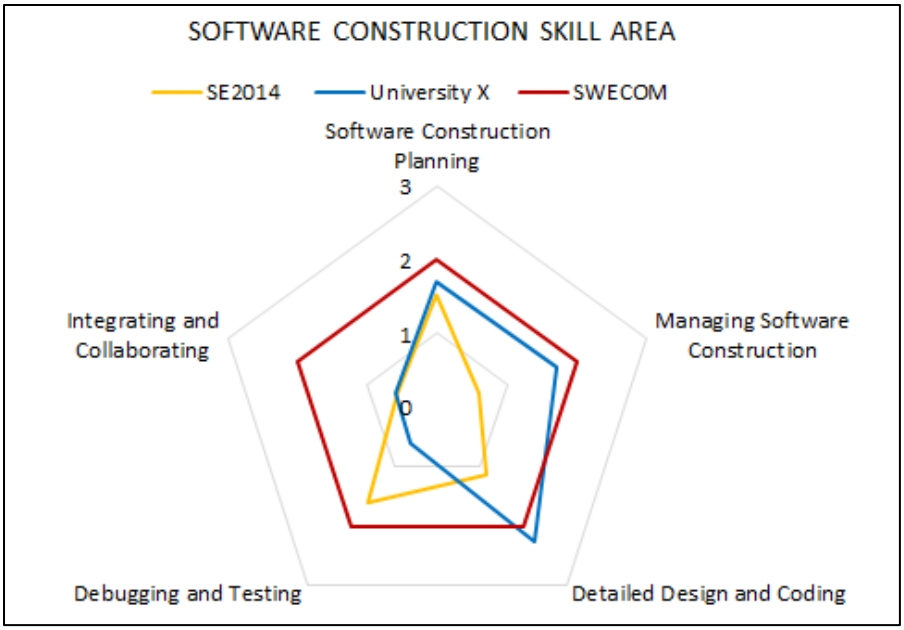

Figure 3. Gray-box analysis for Software Construction

Fig. 3 shows that in the Software Construction skill area SE2014 is close to the suggested levels that SWECOM requires, except for the detailed design and coding skill set. The gray-box analysis of ERAU reveals a close alignment with SWECOM.

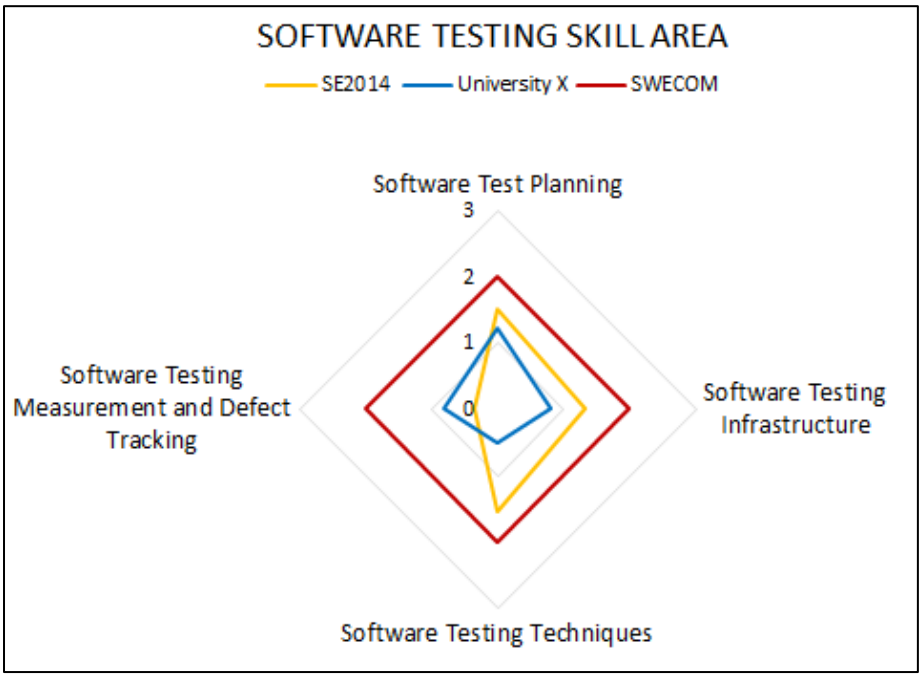

Figure 4. Gray-box analysis for Software Testing

Fig. 4 shows that in the Software Testing skill area SE2014 is not matching up to the requirements of SWECOM. Moreso, for the skill set of software testing mesurement and defect tracking, SE2014 is very far from SWECOM. On the other hand, for ERAU the gray-box analysis of the curriculum shows that ERAU is barely meeting the SE2014 criteria.

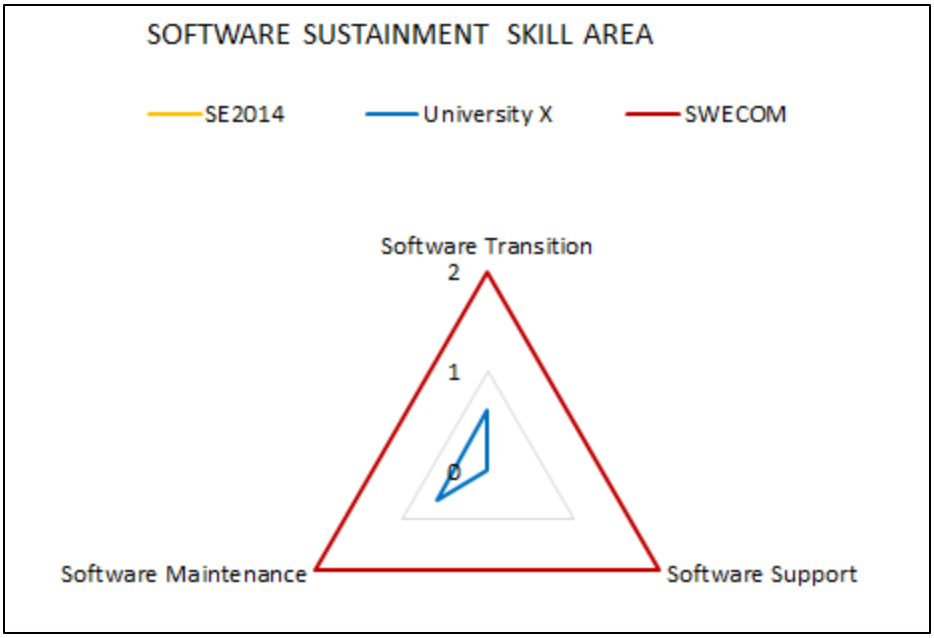

Figure 5. Gray-box analysis for Software Sustainment.

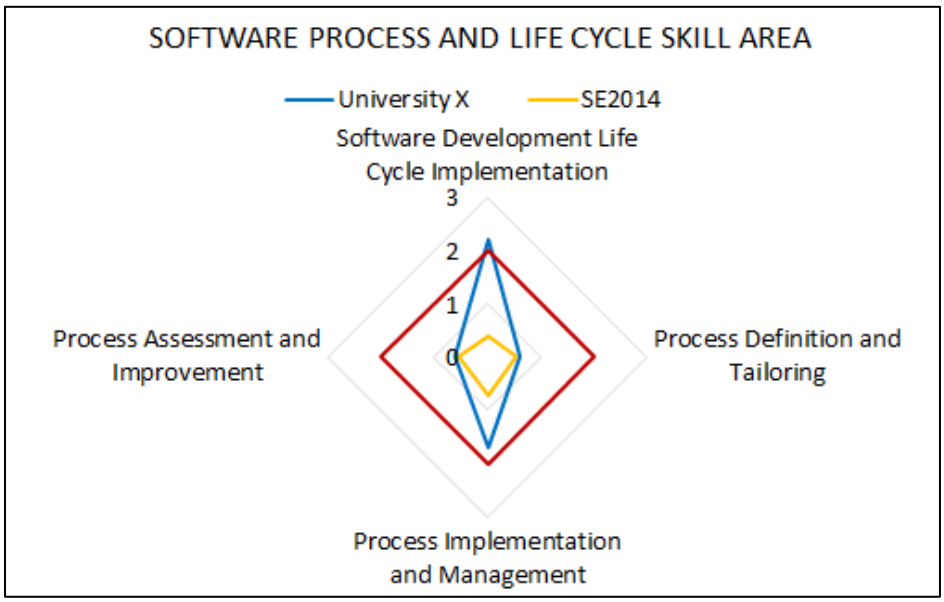

Figure 6. Gray-box analysis for Software Process.

Fig. 5 shows that in the Software Sustainment skill area SE2014 has no required attainment for this area. In addition, ERAU is missing the Software Support skill set.

Fig. 6 shows that SE2014 is not attaining the same knowledge requirements that SWECOM prescribes for the Software Process and Life Cycle skill area, for two of the four areas SE2014 is far from SWECOM. On the other hand the gray-box analysis shows that ERAU is attaining the knowledge required by SWECOM in this skill area.

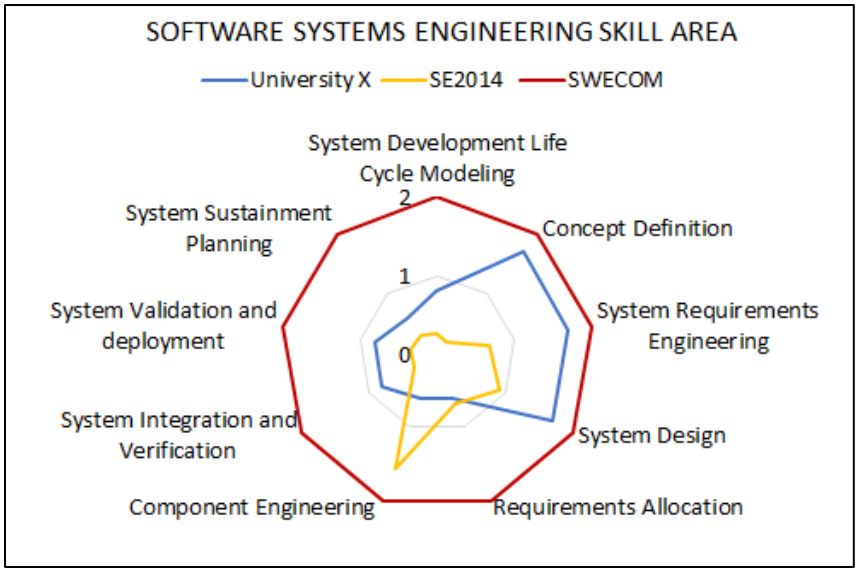

Figure 7. Gray-box analysis for Systems Engineering.

Fig. 7 shows that in Systems Engineering skill area, SE2014 meets SWECOM's required knowledge for two, and almost three, of the skill sets. However, it is very far off in the rest of the skill sets. Univeristy X is doing slightly better, meeting SWECOM's required knowledge levels in two, and almost four, of the skill sets. It is however far off from both SE2014 and SWECOM in three skill areas, System Sustainment Planning, Component Engineering, and Requirements Allocation.

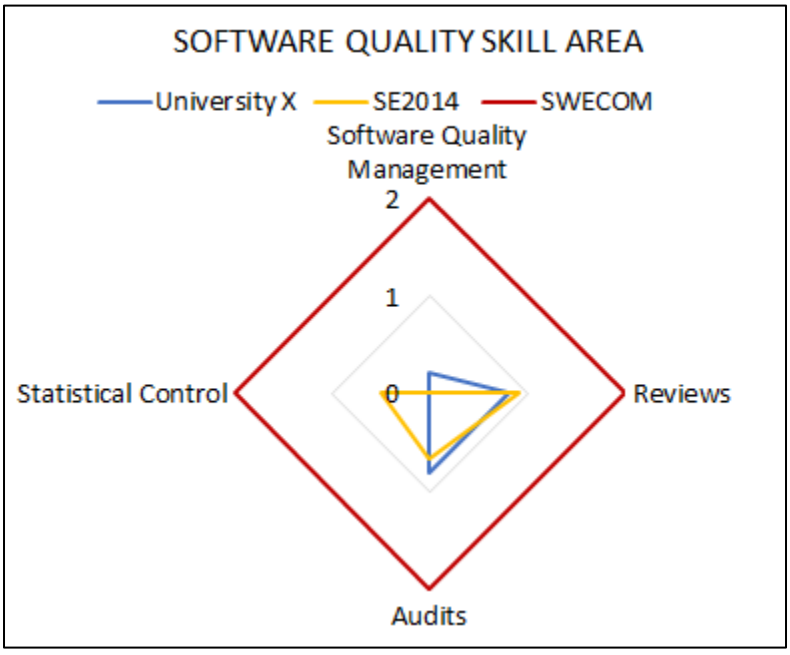

Figure 8. Gray-box analysis for Software Quality

In Fig. 8 the SE2014 guideline is far back from the attainment requirements of SWECOM in the Software Quality skill area. SE2014 only meet one of SWECOM's required skill set, i.e. Audits. Likewise, ERAU is also failing to match both SWECOM and SE2014.

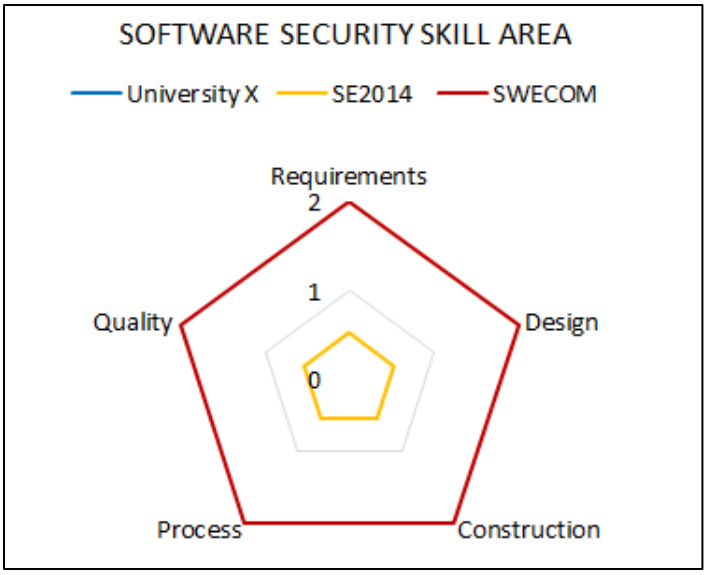

Figure 9. Gray-box analysis for Software Security.

In Fig 9, for the Software Security Skill area, SE2014 is shown to still be behind the knowledge attained requirement of SWECOM. Although ERAU does provide some knowledge attainment for the Software Security skill area, it is still far from meeting the Technician level.

The analysis of the Software Measurements, Safety, Configuration Management, and Human-Computer Interaction skill areas; are postponed at this time. The main reason behind this decision, was the fact that these areas are typically covered under some other topic, for example, safety is typically covered under the security coverage, and human computer interactions are typically covered under design and testing. Therefore, it is almost impossible to conduct an accurate analysis using gray-box analysis approach.

Conclusions

The initial goal of this project was to evaluate the alignment of the SE2014 undergraduate software engineering curriculum with the recommended competency of an entry-level software engineer professional. Our initial hypothesis was that a student graduating with a bachelor's degree in software engineering should meet all the required core knowledge areas under the SE2014, plus additional recommended knowledge areas defined by the guideline. On the other hand, SWECOM identifies the competency expectation of an entry-level software engineer practitioner. The authors of the SWECOM never intended to claim that an entry-level practitioner must possess all the skill sets defined by the SWECOM at the same level. The initial results from our analysis have shown that there are some differences between the SE2014 guideline and SWECOM requirements. Tab. 1 summarizes the results found through the gray-box analysis of ERAU and SE2014 in comparison against SWECOM.

As it is shown in Tab. 1, there are some variation between the SE2014, SWECOM, and ERAU coverage of the skill/knowledge areas, and on the surface some of these differences are troublesome. However, it is important to recognize that there are several assumptions made by the authors of this paper in order to complete this analysis, and some of these assumptions may not be valid and/or accurate. For example, the authors have assumed that what is presented in the

course syllabi is an accurate representation of what is being covered in the class. As it is known, this is a big assumption, and as such, a couple of instructors at ERAU were contacted to verify the accuracy of the syllabi. Based on the further analysis of the course material (i.e., lecture slides, assignments, exam questions, etc.) and discussion with instructors, it became clear that there are several activities associated with the topic of design for security, in multiple classes; however, based on the gray-box analysis no coverage was identified. Leading to the conclusion that a white-box analysis of the course coverage will be necessary.

SE Skill Area	SE2014 vs SWECOM	ERAU vs SWECOM
Software Requirement	Below SWECOM's Entry Level Practitioner, and only meeting the Technician level for four out of five skill sets.	Only meets SWECOM for three of the five skill sets.
Software Design	Below SWECOM's Entry Level Practitioner, and only meeting or exceeding the Technician level for four out of five skill sets.	Meets or exceeds SWECOM for three of the five skill sets.
Software Construction	Slightly below SWECOM's Entry Level Practitioner but missing significantly on one skill set.	Very close to meeting SWECOM for four of the five skill sets. Slightly exceeding for the fifth one.
Software Testing	Below SWECOM's Entry Level Practitioner in all four skill sets.	Only meeting SWECOM's Entry Level Practitioner in three of four skill sets.
Software Sustainment	SE2014 does not provide any guidelines for this area.	Very close to meeting SWECOM for two of the three areas.
Software Process Life Cycle	SE2014 is not matching SWECOM's Entry Level Practitioner, in particular for two of the four areas SE2014 is far from SWECOM.	Meets SWECOM in all four skill sets.
Software Systems Engineering	SE2014 is only matching SWECOM's Entry Level Practitioner in two skill sets, and not meeting the Technician level for one of the skill sets.	Only meets SWECOM for two of the skill sets, close to meeting two other kill sets. Very far behind in one of the skill sets.
Software Quality Management	SE2014 is only matching SWECOM's Entry Level Practitioner in one skill set, and not meeting the Technician level for one of the skill sets.	Does not match SWECOM's Entry Level Practitioner. Meets Technician level for two of the skill sets.
Software Security Skills	SE2014 is behind requirement of SWECOM for this software skill area.	Provides little SE curriculum content on this area.

Table 1 Summary of Conclusions per Skill Area

Future Work

The future direction planned for this research work is twofold: 1) Expand the analysis into a true white-box approach by interviewing all the SE curriculum professors associated with teaching the classes in ERAU to assess if a finer look at the class content uncovers more matching points with SE2014 or SWECOM. 2) Expand the work to apply the gray-box, and possibly white-box approach to other universities to assess if the results are in congruence with the results found in the analysis of ERAU's SE curriculum.

References

1 Ardis, Mark A., et al. "SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering." *IEEE Computer* 48.11 (2015): 106-109.

2 Fairley, Richard E. "A Software Engineering Competency Model (SWECOM)." *IEEE Computer Society* (2014).

3 Engineering Competency Model, United States Department of Labor, 2015; http://www.aaes.org/sites/default/files/Engineering_Competency_Model_draft-0115.pdf

4 Skill Framework for Information Age (SFIA), SFIA Foundation, 2015; https://www.sfia-online.org/en/reference-guide

5 Systems Engineering Competency Model, INCOSE, 2010; http://www.incose.org/ProductsPublications/techpublications/secompetencies

6 Hilburn, Thomas., et al., Software Assurance Competency Model in Technical Note CMU/SEI-2013-TN-004 Software Engineering Institute, Carnegie Mellon University, March 2013.

7 Bourque, Pierre, and Richard E. Fairley. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

8 IEEE Std. 12207-2008, IEEE Standard for Systems and Software Engineering—Software Life Cycle Processes IEEE.

9 Pyster, Art. "Graduate software engineering 2009 (GSwE2009) curriculum guidelines for graduate degree programs in software engineering." *Integrated Software and Systems Engineering Curriculum (iSSEc) series* (2009).

10 Lethbridge, Timothy C., et al. "SE2004: Recommendations for undergraduate software engineering curricula." *IEEE software* 23.6 (2006): 19-25.

11 Towhidnejad, Massood, and Mouza Al Balooshi. "Determining Degree of Alignment of Undergraduate Software Engineering Program with SWECOM." *Proceedings of the International Conference on Software Engineering Research and Practice* (2017): 137-142.

Massood Towhidnejad

Massood Towhidnejad is a Professor of Software Engineering at Embry-Riddle Aeronautical University. His research interest includes; STEM education, software engineering, software quality assurance and testing, autonomous systems, and Air Traffic Management. He has secured about six million dollars in research funding from various government agencies including, the National Science Foundation, the Federal Aviation Administration, the National Oceanic and Atmospheric Administration, and NASA.

Omar Ochoa

Omar Ochoa is an Assistant Professor of Software Engineering at Embry-Riddle Aeronautical University. Omar has over ten years of experience working in industry with companies such as IBM and Hewlett Packard Enterprise, and with the Army Research Lab. Dr. Ochoa has acted as a software technical leader with responsibilities that included mentoring, leadership and client-facing operations. His research interest includes software engineering education, the semantic web, machine learning and cybersecurity.

Anton Kiselev

Anton Kiselev is an undergraduate software engineering student at Embry-Riddle Aeronautical University.