2015

Integrating Security into the Undergraduate Software Engineering Curriculum

Robert Evans
University of North Florida, n00099761@ospreys.unf.edu

INTEGRATING SECURITY INTO THE UNDERGRADUATE SOFTWARE ENGINEERING CURRICULUM

by

Robert E. Evans

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Systems

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

December, 2015

The thesis “Integrating Security into the Undergraduate Software Engineering Curriculum” submitted by Robert E. Evans in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Systems has been

Approved by the thesis committee:	Date
______________________________	____________________
Dr. Robert F. Roggio Thesis Advisor and Committee Chairperson	
______________________________	____________________
Dr. Sandeep Reddivari	
______________________________	____________________
Dr. Swapnoneel Roy	

Accepted for the School of Computing:

______________________________ ____________________

Dr. Sherif Elfayoumy
Director of the School

Accepted for the College of Computing, Engineering and Construction:

______________________________ ____________________

Dr. Mark A. Tumeo
Dean of the College

Accepted for the University:

______________________________ ____________________

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEDGEMENT

I wish to especially thank my wife for her support and understanding during the long process of creating this paper and my continuing educational endeavors. Additionally I would like to thank Dr. Roggio for his direction and guidance in completing this thesis.

CONTENTS

FIGURES

TABLES

ABSTRACT

This research included a thorough examination of the existing software assurance or what is commonly called software security knowledge, methodologies and what information security technologies is currently being recommended by the information technology community. Finally it is demonstrated how this security knowledge could be incorporated into the curriculum for undergraduate software engineering.

Since the birth of the Internet in 1969 when the first ARPANET nodes were created there has been exponential growth in the amount of data available and collected on the Internet. Users are creating 2.5 quintillion (a billion billion, 10^{18}) bytes of data every day. Ninety percent (90%) of the data in the world today has been created in just the last two years. Every day in all sectors of our economy there are reported security breaches to this data, e.g. (Retail) Target, Home Depot and Supervalu, (Entertainment) Sony, (Insurance) Anthem, (Banking) JP Morgan and Bank of America, (Internet) Apple, Yahoo and eBay, (Government) Veterans Administration. This is just a small sample of just the “reported” breaches.

Technology has contributed to tremendous productivity and quality-of-life gains in our society. Technology has been incorporated into virtually every aspect of our nation’s infrastructure, and we have become critically reliant upon it. Our reliance has made it extremely important that more emphasis needs to be placed on its protection.

The field of software engineering education needs to evolve to where it recognizes the necessity to produce software that is free from vulnerabilities, either intentionally designed or accidentally inserted during its lifetime. Software assurance principles and practices need to be explicitly incorporated into the software engineering curriculum. The education of software assurance must no longer be an elective. It has become imperative that our software engineering workforce must become more adept at producing secure software.

Chapter 1

INTRODUCTION

Today's software is interconnected, far-reaching and complex, and because of these attributes modern software applications are highly vulnerable to disclosure, theft, destruction and misuse. The importance of protecting our national infrastructures and systems has been acknowledged by the U.S. Department of Homeland Security (DHS). The DHS acknowledges that there is a growing need to produce more skilled practitioners of software assurance. The DHS began the Software Assurance (SwA) program as a result of a National Strategy to Secure Cyberspace which states: “DHS will facilitate a national public-private effort to promulgate best practices and methodologies that promote integrity, security, and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, malicious code, or trap doors that could be introduced during development” [DHS10].

1.1 Problem Statement

Software engineering students must possess the necessary software assurance knowledge to produce secure software if they are to be prepared to enter the workforce. The problem is clear and continues to be recognized: companies and organizations place a premium on prospective employees that possess some background in security [BLS15, McDonald15, Gartner14, Monster15, Dubie08], and the dated curriculum guidelines do not adequately address the subject of security. Consequently, it is hypothesized that these incomplete, outdated guidelines have contributed to the woefully under-prepared graduating software engineering students, who are ill-equipped to produce secure software.

1.2 Background

In the fall of 2001 a joint task force on Computing Curricula was formed by the Association for Computing Machinery (ACM) and IEEE Computer Society (IEEE-CS). The result of this task force was the publishing on August 23 2004 of "Software Engineering 2004 (SE2004), Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series." The stated purpose of these guidelines is to provide guidance to academic institutions and accreditation agencies about what should constitute an undergraduate software engineering education [IEEE-CS04A]. When work on this thesis started the 2004 guidelines were not revised and they remain the guidelines in use today in most organizations.

It should be noted that very recently, late in the development of this thesis, the ACM and IEEE have jointly approved a revised version of the Curriculum Guidelines dated 23 February 2015. These newly revised guidelines validate a great deal of the premises put forth by this paper [IEEE-CS15]. An analysis of the newly released guidelines will be given in Chapter 5.

1.3 Overview of the Research Methodology

Given the stated problem, my research will proceed as follows:

- First, it will be determined if the problem exists by researching current thinking as to what the software engineering community is saying about the current SE2004 curriculum guidelines.
- Secondly, based on current research, scholarly comments and practitioner experiences in the software engineering field of what a generally agreed upon software assurance body of knowledge is, A knowledge base of software assurance knowledge applicable to an undergraduate software engineering student will be developed.
- Thirdly, a side by side analysis of the software assurance knowledge areas and the SE2004 curriculum guidelines will be presented to show which if any software assurance knowledge areas are not covered.
- And finally, the research will culminate in recommendations as to how if necessary software assurance knowledge areas can be integrated into the SE2004 curriculum guidelines.

To accomplish these activities this thesis presents:

1. A review of the current literature to obtain a sense of the current software engineering environment as it pertains to the education of software engineering, software assurance and the guidelines contained in SE2004.

2. Utilizing the software assurance knowledge base accumulated by the Software Engineering Institute (SEI), the IEEE Computer Society (IEEE-CS) and the Association for Computing Machinery (ACM) and various others as indicated, investigation and aggregation of what the generally agreed upon software assurance knowledge areas of an undergraduate software engineering student should consists.
3. An analysis of the knowledge areas in the SE2004 curriculum guidelines for identification of those knowledge areas that may be suitable for software assurance incorporation.
4. An assessment of the current guidelines for logical coalescing of the excluded software assurance knowledge areas.

Chapter 2

LITERATURE REVIEW

2.1 Computing Curricula Series

The curriculum guidelines for an undergraduate degree in software engineering were originally outlined in the "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (SE2004)" [IEEE-CS04A]. This document was developed as part of the "Computing Curricula Series" of curriculum guidelines for several computing disciplines. Early in their process of creating the curricula series it became obvious to the task force that because of the incredible advances in technology that a set of curriculum guidelines were needed to cover a more expanded family of computer-related disciplines [IEEE-CS05]. The complete series included the following disciplines:

- Computer Engineering
- Computer Science
- Information Systems
- Information Technology
- Software Engineering

The Curricula Series is illustrated in Figure 1.

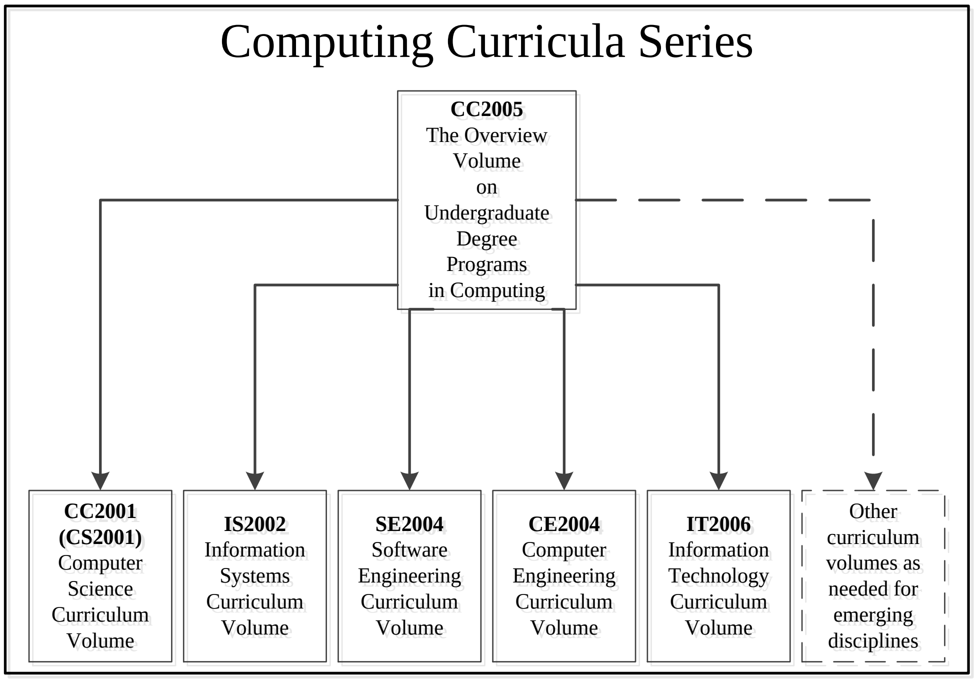
Computing Curricula Series
CC2005 The Overview Volume on Undergraduate Degree Programs in Computing
CC2001 (CS2001) Computer Science Curriculum Volume
IS2002 Information Systems Curriculum Volume
SE2004 Software Engineering Curriculum Volume
CE2004 Computer Engineering Curriculum Volume
IT2006 Information Technology Curriculum Volume
Other curriculum volumes as needed for emerging disciplines

Figure 1: Structure of the Computing Curricula Series [IEEE-CS05]

The stated purpose of the guidelines was "to explain the character of the various undergraduate degree programs in computing and to help one determine which of the programs are most suited to one's particular goals and circumstances" [IEEE-CS05]. Table 1 shows the curricula series knowledge areas and the relative emphasis (0=Lowest 5=Highest) that each discipline places on it. Software Engineering (SE) is the highlighted column. (CE) is Computer Engineering, (CS) is Computer Science, (IS) is Information Systems, and (IT) is Information Technology [IEEE-CS05].

Area	*Performance Capability*	*CE*	*CS*	*IS*	*IT*	*SE*
Algorithms	Prove theoretical results	3	5	1	0	**3**
	Develop solutions to programming problems	3	5	1	1	**3**
	Develop proof-of-concept programs	3	5	3	1	**3**
	Determine if faster solutions possible	3	5	1	1	**3**
Application programs	Design a word processor program	3	4	1	0	**4**
	Use word processor features well	3	3	5	5	**3**
	Train and support word processor users	2	2	4	5	**2**
	Design a spreadsheet program (e.g., Excel)	3	4	1	0	**4**
	Use spreadsheet features well	2	2	5	5	**3**
	Train and support spreadsheet users	2	2	4	5	**2**
Computer programming	Do small-scale programming	5	5	3	3	**5**
	Do large-scale programming	3	4	2	2	**5**
	Do systems programming	4	4	1	1	**4**
	Develop new software systems	3	4	3	1	**5**
	Create safety-critical systems	4	3	0	0	**5**
	Manage safety-critical projects	3	2	0	0	**5**
Hardware and devices	Design embedded systems	5	1	0	0	**1**
	Implement embedded systems	5	2	1	1	**3**
	Design computer peripherals	5	1	0	0	**1**
	Design complex sensor systems	5	1	0	0	**1**
	Design a chip	5	1	0	0	**1**
	Program a chip	5	1	0	0	**1**
	Design a computer	5	1	0	0	**1**
Human-computer interface	Create a software user interface	3	4	4	5	**4**
	Produce graphics or game software	2	5	0	0	**5**
	Design a human-friendly device	4	2	0	1	**3**
Information systems	Define information system requirements	2	2	5	3	**4**
	Design information systems	2	3	5	3	**3**
	Implement information systems	3	3	4	3	**5**
	Train users to use information systems	1	1	4	5	**1**
	Maintain and modify information systems	3	3	5	4	**3**
Information management (Database)	Design a database mgmt. system (e.g., Oracle)	2	5	1	0	**4**
	Model and design a database	2	2	5	5	**2**
	Implement information retrieval software	1	5	3	3	**4**
	Select database products	1	3	5	5	**3**
	Configure database products	1	2	5	5	**2**
	Manage databases	1	2	5	5	**2**
	Train and support database users	2	2	5	5	**2**
IT resource planning	Develop corporate information plan	0	0	5	3	**0**
	Develop computer resource plan	2	2	5	5	**2**
	Schedule/budget resource upgrades	2	2	5	5	**2**
	Install/upgrade computers	4	3	3	5	**3**
	Install/upgrade computer software	3	3	3	5	**3**
Intelligent systems	Design auto-reasoning systems	2	4	0	0	**2**
	Implement intelligent systems	2	4	0	0	**4**
Networking and communications	Design network configuration	3	3	3	4	**2**
	Select network components	2	2	4	5	**2**
	Install computer network	2	1	3	5	**2**
	Manage computer network	3	3	3	5	**3**
	Implement communication software	5	4	1	1	**4**
	Manage communication resources	1	0	3	5	**0**
	Implement mobile computing system	5	3	0	1	**3**
	Manage mobile computing resources	3	2	2	4	**2**
Systems Development Through Integration	Manage and organization's web presence	2	2	4	5	**2**
	Configure & integrate e-commerce software	2	3	4	5	**4**
	Develop multimedia solutions	2	3	4	5	**3**
	Configure & integrate e-learning systems	1	2	5	5	**3**
	Develop business solutions	1	2	5	3	**2**
	Evaluate new forms of search engine	2	4	4	4	**4**

Table 1: Computing Curricula Knowledge Areas [IEEE-CS05]

At the time the series was created, the discipline of software engineering had emerged from the discipline of computer science. The emergence was triggered by the complex nature that software had become and the desire to apply traditional engineering techniques and ideas to the questions rose in the construction of quality software [IEEE-CS05].

2.2 Software Engineering Curriculum Guidelines

The curriculum guidelines for software engineering began in 2001 as a task force convened by the (ACM) and the Computer Society of the (IEEE). The task force volunteers were first charged with developing the "Software Engineering Education Knowledge" (SEEK) areas [IEEE-CS04A]. Knowledge areas represent the significant knowledge categories that a software engineering education should contain. These knowledge areas illustrate the top level structural element that organizes, classifies and describes software engineering knowledge. Each area is also identified by an abbreviation. The resulting ten areas were:

- Computing Essentials (CMP)
- Mathematical and Engineering Fundamentals (FND)
- Professional Practice (PRF)
- Software Modeling and Analysis (MAA)
- Software Design (DES)
- Software Verification and Validation (VAV)
- Software Evolution (EVL)

- Software Process (PRO)
- Software Quality (QUA)
- Software Management (MGT)

Each of these knowledge areas is then broken down into knowledge units, which further sub-divide the area into more specific thematic modules. Each knowledge unit is identified by a two or three-letter suffix which is added to the knowledge area abbreviation. The SEEK knowledge areas and knowledge units are illustrated in Table 2. The indicated hours represent the minimum number of classroom hours required to adequately present the material [IEEE-CS04A]. As can be seen, software assurance or security was not included as an area or unit.

Knowledge				Knowledge			
Area	Unit	Title	Hours	Area	Unit	Title	Hours
CMP		Computing Essentials	172	MAA		Software Modeling and Analysis	53
	cf	Computer Science Foundations	140		md	Modeling Foundations	19
	ct	Construction Technologies	20		tm	Types of Models	12
	tl	Construction Tools	4		sf	Analysis Fundamentals	6
	fm	Formal Construction Methods	8		rfd	Requirements Fundamentals	3
VAV		Software V & V	42		er	Eliciting Requirements	4
	fnd	V & V Terminology and Foundations	5		rsd	Requirements Specification and Documentation	6
	rev	Reviews	6		rv	Requirements Validation	3
	tst	Testing	21	QUA		Software Quality	16
	hct	Human Computer UI Testing and Evaluation	6		cc	Software Quality Concepts and Culture	2
	par	Problem Analysis and Reporting	4		std	Software Quality Standards	2
FND		Mathematical and Engineering Fundamentals	89		pro	Software Quality Processes	4
	mf	Mathematical Foundations	56		pca	Process Assurance	4
	ef	Engineering Foundations for Software	23		pda	Product Assurance	4
	ec	Engineering Economics for Software	10	DES		Software Design	45
EVL		Software Evolution	10		con	Design Concepts	3
	pro	Evolution Processes	6		str	Design Strategies	6
	ac	Evolution Activities	4		ar	Architectural Design	9
PRF		Professional Practice	35		hci	Human Computer Interface Design	12
	psy	Group Dynamics/Psychology	5		dd	Detailed Design	12
	com	Communication Skills (Specific to SwE)	10		ste	Design Support Tools and Evaluation	3
	pro	Professionalism	20	MGT		Software Management	19
PRO		Software Process	13		con	Management Concepts	2
	con	Process Concepts	3		pp	Project Planning	6
	imp	Process Implementation	10				

Table 2: Computing Curricula Knowledge Areas and Units [IEEE-CS04A]

The SEEK units are further broken down into one or more SEEK topics, these SEEK topics represent the lowest level in the SEEK hierarchy. These topics are described utilizing Bloom's taxonomy dated 1956 [Bloom56]. Figure 2 describes the Bloom taxonomy used.

BLOOM'S TAXONOMY

- **Knowledge (k)** - Remembering previously learned material. Test observation and recall of information; that is, "bring to mind the appropriate information" (e.g. dates, events, places, knowledge of major ideas, mastery of subject matter).
- **Comprehension (c)** - Understanding information and the meaning of material presented. For example, be able to translate knowledge to a new context, interpret facts, compare, contrast, order, group, infer causes, predict consequences, etc.
- **Application (a)** - Ability to use learned material in new and concrete situations. For example, using information, methods, concepts, and theories to solve problems requiring the skills or knowledge presented.

A topic's relevance to the core is represented as follows:

- **Essential (E)** - The topic is part of the core.
- **Desirable (D)** - The topic is not part of the core, but it should be included in the core of a particular program if possible; otherwise, it should be considered as part of elective materials.
- **Optional (O)** - The topic should be considered as elective only.

Figure 2: Bloom's Taxonomy [IEEE-CS04A]

An example of this SEEK hierarchy is illustrated in Figure 3. This SEEK taxonomy and hierarchical structure would be essential in knowing when attempting to integrate additional SEEK topics into the SE2004 curriculum guidelines.

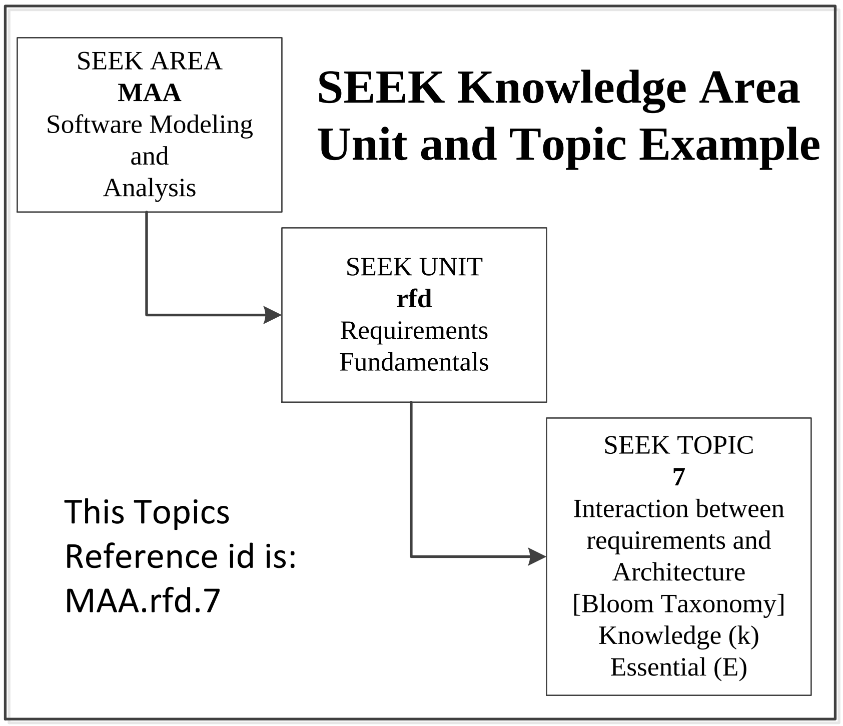
SEEK AREA
MAA
Software Modeling and Analysis
SEEK Knowledge Area Unit and Topic Example
SEEK UNIT
rfd
Requirements Fundamentals
SEEK TOPIC
7
Interaction between requirements and Architecture
[Bloom Taxonomy]
Knowledge (k)
Essential (E)
This Topics Reference id is: MAA.rfd.7

Figure 3: SEEK Knowledge Hierarchy [IEEE-CS04A]

The complete SEEK listing by knowledge area, knowledge unit and topic is listed in Appendix A.

It should be noted that when the draft SE2004 was submitted for review, there were numerous comments regarding the omission of security. The SE2004 task force response to such comments was:
"While security issues have been increasing in importance, little is found in current undergraduate curricula on it (which most likely explains why little to no reference has been found in other KAs [Knowledge Areas]). As the exposure to security issues increases in the classrooms, it will also be found in more KAs" [IEEE-CS03A, IEEE-CS04B].

It should also be noted that the newly revised curriculum guidelines maintain the same SEEK nomenclature and structure as the SE2004 guidelines, which is knowledge area, unit and topic, and in addition incorporates the same Bloom's taxonomy [IEEE-CS15].

2.3 Software Assurance

In February 2005, the President's Information Technology Advisory Committee (PITAC) issued a report titled "Cyber Security: A Crisis of Prioritization" [PITAC05]. This report identified the top ten areas requiring attention in securing our nation's critical technology infrastructure. One of the areas identified was 'secure software engineering and software assurance'. The report stated: "Commercial software engineering today lacks the scientific underpinnings and rigorous controls needed to produce high-quality, secure products at acceptable cost. Commonly used software engineering practices permit dangerous errors, such as improper handling of buffer overflows, which enable hundreds of attack programs to compromise millions of computers every year. In the future, the nation may face even more challenging problems as adversaries – both foreign and domestic –become increasingly sophisticated in their ability to insert malicious code into critical software" [PITAC05]. As a result of this report the US Department of Homeland Security (DHS) launched the Software Assurance Program whose stated goal is: "Facilitate a national public private effort to promulgate best practices and methodologies that promote integrity, security and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, or trap doors that could be introduced during development" [Jarzombek06].

The official United States government definition of software assurance is:
"Level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle and that the software functions in the intended manner" [CNSS10]. Software assurance best practices and knowledge areas are a growing list of processes techniques and tools being developed and utilized by technology professionals.

The DHS enlisted the resources of the Computer Emergency Response Team (CERT) program at the Software Engineering Institute (SEI) to develop and maintain software assurance related information. The SEI oversaw the development of a Software Assurance Core Body of Knowledge CorBok [Redwine06] and a corresponding Software Assurance Competency Model [Hilburn13]. The SEI Software Assurance Competency Model breaks out the knowledge areas into levels of competency which are distinguished by different levels of professional capabilities relative to knowledge, skills and effectiveness. The five levels of software assurance competency are described in Table 3 [Hilburn13].

L1	Technician	
		Possesses technical knowledge and skills, typically gained through a certificate or an associate degree program, or equivalent knowledge and experience May be employed in a system operator, implementer, tester, or maintenance position with specific individual tasks assigned by someone at a higher level Main areas of competency: System Operational Assurance, System Functionality Assurance, and System Security Assurance (see CorBok) Major tasks: tool support, low-level implementation, testing and maintenance
L2	Professional Entry Level	
		Possesses application-based knowledge and skills and entry-level professional effectiveness, typically gained through a bachelor's degree in computing or through equivalent professional experience May perform all tasks of L1. May also manage a small internal project; supervise and assign sub-tasks for L1 personnel; supervise and assess system operations; and implement commonly accepted assurance practices. Main areas of competency: System Functionality Assurance, System Security Assurance, and Assurance Assessment (see CorBok) Major tasks: requirements fundamentals, module design, and implementation
L3	Practitioner	
		Possesses breadth and depth of knowledge, skills, and effectiveness beyond L2, and typically has two to five years of professional experience May perform all tasks of L2. May also set plans, tasks, and schedules for in-house projects; define and manage such projects and supervise teams on the enterprise level; report to management; assess the assurance quality of a system; implement and promote commonly accepted software assurance practices Main areas of competency: Risk Management, Assurance Assessment, and Assurance Management (see CorBok) Major tasks: requirements analysis, architectural design, tradeoff analysis, and risk assessment
L4	Senior Practitioner	
		Possesses breadth and depth of knowledge, skills, and effectiveness and a variety of work experiences beyond L3, with 5 to 10 years of professional experience and advanced professional development at the master's level or with equivalent education/training May perform all tasks of L3. May also identify and explore effective software assurance practices for implementation, manage large projects, interact with external agencies, and so forth Main areas of competency: Risk Management, Assurance Assessment, Assurance Management, and Assurance Across Lifecycles (see CorBok) Major tasks: assurance assessment, assurance management, and risk management across the lifecycle
L5	Expert	
		Possesses competency beyond L4; advances the field by developing, modifying, and creating methods, practices, and principles at the organizational level or higher; has peer/industry recognition; typically includes a low percentage of an organization's workforce within the Software Assurance profession (e.g., 2% or less)

Table 3: Software Assurance Competency Levels [Hilburn13]

As can be seen, the level of software assurance knowledge and skill for a bachelor's degree would be L2, Professional Entry Level [Hilburn13]. The Software Assurance Competency Model is based on extensive review by the DHS, IEEE-CS and the SEI and covers the entire spectrum of software assurance practices [Hilburn13]. Table 4 describes the knowledge areas of the Software Assurance Competency Model [Hilburn13]. The complete listing of the Software Assurance Competency Model core body of knowledge is listed in Appendix B.

Knowledge Area (KA)	KA Competency
Assurance Across Lifecycles	The ability to incorporate assurance technologies and methods into lifecycle processes and development models for new or evolutionary system development, and for system or service acquisition
Risk Management	The ability to perform risk analysis and tradeoff assessment, and to prioritize security measures
Assurance Assessment	The ability to analyze and validate the effectiveness of assurance operations and create auditable evidence of security measures
Assurance Management	The ability to make a business case for software assurance, lead assurance efforts, understand standards, comply with regulations, plan for business continuity, and keep current in security technologies
System Security Assurance	The ability to incorporate effective security technologies and methods into new and existing systems
System Functionality Assurance	The ability to verify new and existing software system functionality for conformance to requirements and help reveal malicious content
System Operational Assurance	The ability to monitor and assess system operational security and respond to new threats

Table 4: Software Assurance Core Knowledge Areas [Hilburn13]

2.4 Software Engineering Body of Knowledge (SWEBOK)

When the SE2004 curriculum guidelines were created in 2001 and published in 2004 by the IEEE-CS and the ACM, the task force stated that one of their primary sources in creating the SE2004 SEEK areas, units and topics was the SWEBOK [Bourque01]. The SEEK structure of area, unit and topic is similar to the SWEBOK which uses area and topic.

Additionally, the SEEK topic Bloom's taxonomy was derived from the SWEBOK [IEEE-CS04A]. The SE2004 task force viewed the SWEBOK as the entire "Body of Knowledge" for the practicing of the software engineering profession. The competency models for the SWEBOK were not developed until 2014. The task force therefore extracted from the SWEBOK only the applicable undergraduate software engineering knowledge resulting in what is referred to as Software Engineering Education Knowledge (SEEK) [IEEE-CS04A].

The current SWEBOK version is 3.0 dated 2014 [Bourque14]. One of the stated objectives of the current SWEBOK is to provide a foundation for curriculum development.

There are fifteen (15) knowledge areas in the SWEBOK they are listed in Table 5 [Bourque14].

SWEBOK 2014 Version 3 Knowledge Areas (KAs)
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

Table 5: SWEBOK Knowledge Areas [Bourque14]

The knowledge areas in the current SWEBOK are decomposed into topics. A complete listing of the SWEBOK knowledge areas topics is listed in Appendix C.

The current version of the SWEBOK incorporates software assurance with the inclusion in Chapter 13 Computing Foundations of topic 17 titled "Secure Software Development and Maintenance" [Bourque14]. The topics are:

- Software Requirements Security
- Software Design Security
- Software Construction Security
- Software Testing Security
- Build Security into Software Engineering Process
- Software Security Guidelines

A complete description of the secure software development and maintenance topics is given in Table 6. It should be noted that the current SWEBOK does not include the Bloom's taxonomy from the 2001 version used by the SE2004.

Secure Software Development and Maintenance Topics (Chapter 13 SWEBOK)
Software Requirements Security
Software requirements security deals with the clarification and specification of security policy and objectives into software requirements, which lays the foundation for security considerations in the software development. Factors to consider in this phase include software requirements and threats/risks. The former refers to the specific functions that are required for the sake of security; the latter refers to the possible ways that the security of software is threatened.
Software Design Security
Software Design security deals with the design of software modules that fit together to meet the security objectives specified in the security requirements. This step clarifies the details of security considerations and develops the specific steps for implementation. Factors considered may include frameworks and access modes that set up the overall security monitoring/enforcement strategies, as well as the individual policy enforcement mechanisms.
Software Construction Security
Software construction security concerns the question of how to write actual programming code for specific situations such that security considerations are taken care of. The term “Software Construction Security” could mean different things for different people. It can mean the way a specific function is coded, such that the coding itself is secure, or it can mean the coding of security into software.
Software Testing Security
Software testing security determines that software protects data and maintains security specification as given.
Build Security into Software Engineering Process
Software is only as secure as its development process goes. To ensure the security of software, security must be built into the software engineering process. One trend that emerges in this regard is the Secure Development Lifecycle (SDL) concept, which is a classical spiral model that takes a holistic view of security from the perspective of software lifecycle and ensures that security is inherent in software design and development, not an afterthought later in production. The SDL process is claimed to reduce software maintenance costs and increase reliability of software concerning software security related faults.
Software Security Guidelines
Although there are no bulletproof ways for secure software development, some general guidelines do exist that can be used to aid such effort. These guidelines span every phase of the software development lifecycle. Some reputable guidelines are published by the Computer Emergency Response Team (CERT) and below are its top 10 software security practices. 1. Validate input. 2. Heed compiler warnings. 3. Architect and design for security policies. 4. Keep it simple. 5. Default deny. 6. Adhere to the principle of least privilege. 7. Sanitize data sent to other software. 8. Practice defense in depth. 9. Use effective quality assurance techniques. 10. Adopt a software construction security standard.

Table 6: Secure Software Development and Maintenance Topics [Bourque14]

2.5 Software Engineering Competency Model (SWECOM)

Competency as defined by the dictionary is "an ability or skill" [Merriam-Webster15]. Competency levels of a given profession are measured as different stages of a career path. As has previously been mentioned, the SWEBOK constitutes the entire body of knowledge of the software engineering profession (see section 2.4), and this section describes the software engineering competency model SWECOM for the software engineering profession. The Software Engineering Competency Model SWECOM began in 2014 as a project of the IEEE Computer Society [IEEE-CS14]. The SWECOM is based on the SWEBOK and its stated purpose is to present a competency model for use by those who develop software, their managers, human-resource personnel, curriculum designers, and others [IEEE-CS14]. The SWECOM was also developed using the Software Assurance Competency Model and the SE2004 among others [IEEE-CS14]. This relationship is illustrated in Figure 4.

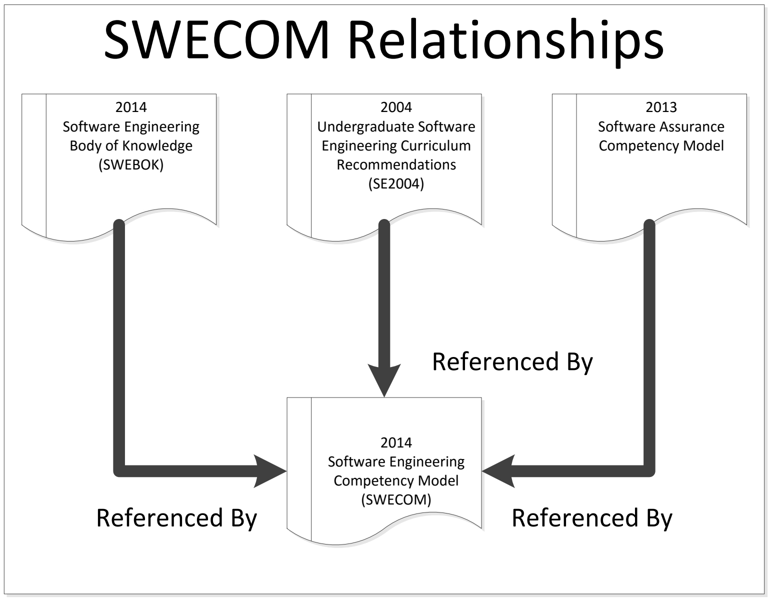

Figure 4: SWECOM Relationships Illustrated [IEEE-CS14]

The applicable software assurance content can be found in Chapter 19 of the SWECOM, titled "Software Security Skills". The SWECOM describes software security skills as applicable across all areas of software engineering [IEEE-CS14]. Table 7 illustrates the software security skill area and skill sets from Chapter 19 of the SWECOM and their corresponding activities.

Software Security Skill Area (Chapter 19 SWECOM)
Requirements Skill Set
Identifies security risks (such as misuse cases).
Creates requirements that capture security issues.
Performs initial threat modeling.
Creates or proposes new methods for recognizing security vulnerabilities.
Design Skill Set
Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data).
Uses appropriate, secure design patterns.
Models threats and associated risks of new and modified systems.
Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems.
Construction Skill Set
Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation).
Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems).
Selects or establishes project coding standards to avoid security vulnerabilities.
Reviews and approves coding standards to avoid security vulnerabilities.
Establishes organization coding standards to avoid security vulnerabilities.
Creates new coding standards to avoid security vulnerabilities.
Process Skill Set
Assists in the collection of metrics for security assessment processes.
Follows project standards in the collection of security assessment metrics.
Establishes organization standards for security assessment processes.
Quality Skill Set
Assists in the installation of static analysis tools.
Performs code reviews to identify security vulnerabilities.
Uses static analysis methods to identify security vulnerabilities.
Selects appropriate static analysis tools to identify security vulnerabilities.
Creates new static analysis methods or tools.

Table 7: SWECOM Software Security Skills [IEEE-CS14]

The SWECOM is organized by skill area, skill sets within skill areas, and activities within skill sets. Similar to the software assurance competency levels, the activities are specified at five levels of competency they are:

- Technician

- Entry Level Practitioner
- Practitioner
- Technical Leader
- Senior Software Engineer

The activities performed at the specific competency levels are described as:

- Follows (F)
- Assists (A)
- Participates (P)
- Leads (L)
- Creates (C)

An example of this SWECOM Skill hierarchy is illustrated in Figure 5.

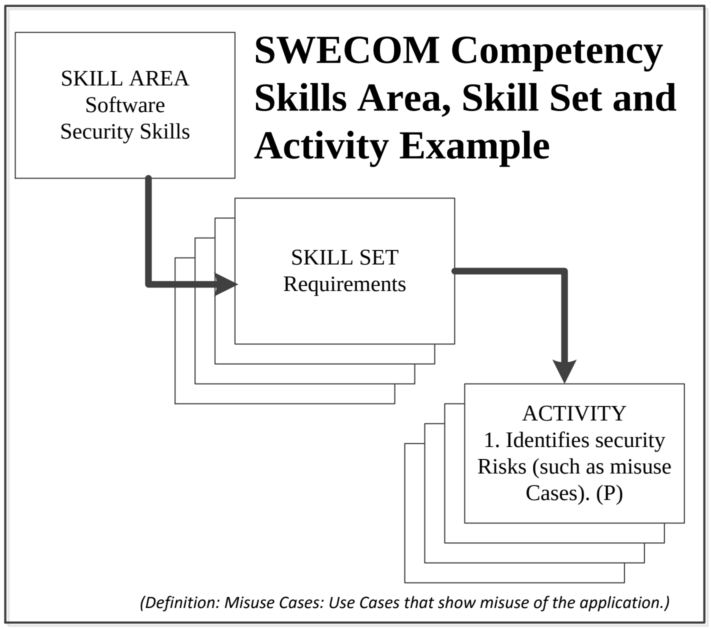
SWECOM Competency Skills Area, Skill Set and Activity Example
SKILL AREA Software Security Skills
SKILL SET Requirements
ACTIVITY 1. Identifies security Risks (such as misuse Cases). (P)
(Definition: Misuse Cases: Use Cases that show misuse of the application.)

Figure 5: SWECOM Skill Hierarchy Example [IEEE-CS14]

The SWECOM, like the SEEK, is based off the knowledge in the SWEBOK [IEEE-CS14]. The SWECOM however includes the entire "Body of Knowledge" and collates the knowledge into categories called competency levels. When the SE2004 task force collated out the undergraduate knowledge in 2001 for the SEEK there wasn't any security knowledge to include in the SEEK. Additionally the SWECOM does not include the SWEBOK Bloom's taxonomy that was included in the SEEK. This thesis will later demonstrate, utilizing the SWECOM, how to integrate the undergraduate SWEBOK security knowledge into the SEEK.

The SWECOM competency level that equates to graduating undergraduate software engineering students is "Entry Level Practitioner", which is equivalent to an accredited software engineering degree program or equivalent and zero to four years of relevant experience [IEEE-CS14]. Because the SWECOM is based on the SWEBOK, the software assurance competency model and the SE2004, it would be an excellent source for integrating software assurance into the curriculum guidelines. Appendix D provides a complete listing of the SWECOM skills and activities by competency level.

2.6 IT Workforce

The Bureau of Labor Statistics, U.S. Department of Labor reports: "Cyberattacks have grown in frequency and sophistication over the last few years, and many organizations are behind in their ability to detect these attacks. Analysts will be needed to come up with innovative solutions to prevent hackers from stealing critical information or creating havoc on computer networks” [BLS15]. The Bureau of Labor Statistics also reports that employment of information security analysts is projected to grow 37 percent from 2012 to 2022, much faster than the average for all occupations [BLS15].

In addition, according to the Computing Technology Industry Association (CompTIA) a non-profit trade association: “Currently, and likely for the foreseeable future, mobile security and big data will drive IT hiring. Candidates with the right experience in areas like security and software development often entertain several job offers at one time” [McDonald15].

While it has always been an important field, IT security is growing even more critical as businesses place more reliance on digital processes and the field is also showing healthy growth, with expectations for final 2014 revenue for global security to hit $71.1 billion and grow a further 8.2 percent in 2015 [Gartner14].

It is obvious that security has been catapulted to the forefront of sought after IT skills. According to Monster.com the online job search site some might think that 2014 was the year of the hacks, with a number of major companies alerting customers of potential data breaches, and some companies experiencing internal hacks [Monster15]. Moving into 2015, companies will be looking to save themselves from any potential fiascos around sensitive data being released, whether its internal memos, credit card information, or consumer data.

Another job search site, Dice.com, reported to Information Week magazine that IT security is the number one fastest-growing tech skills area [Casey15]. Information Week also reports that security is one of the hottest fields for IT recruiters in 2015. Execusource, another online job recruiter site, posted a report titled "Top 4 Sought after Tech Skills for 2014" within which security ranked number four [Execusource14]. Network World magazine reports: Security as a must-have skill is a no-brainer. And finally, the Network World reports states: “going forward, IT professionals will need to be able to incorporate their security savvy into network, wireless, application, operating system and other IT areas to best compete” [Dubie08].

2.7 Summary

It is clear from the literature review that security skills are highly sought after in the IT industry. It has also been demonstrated that the need for more secure software is receiving more and more attention. Industry has responded by developing software assurance programs like Microsoft’s Security Development Lifecycle (SDL), [Microsoft15], Oracle’s Software Security Assurance [Oracle15], and Department of Homeland Security’s Build Security In [DHS15]. Several scholarly papers have undertaken efforts to define and describe what software security skills are and what their competencies should be. Until the very recent publication of SE2014 in February 2015, the Software Engineering undergraduate degree curriculum has received little or no attention since its original publication in 2004.

Research indicates that the SE2004 curricula guidelines for undergraduate software engineering has not been brought current with the latest version of the Software Engineering Body of Knowledge SWEBOK, Software Assurance Body of Knowledge CorBOK, or any other current bodies of knowledge.

It should be noted that the recently revised curriculum guidelines do reference the current version of the Software Engineering Body of Knowledge SWEBOK. However the Software Engineering Competency Model SWECOM was not cited in the revised 2014 guidelines [IEEE-CS15]. This is only stated to point out that, both the SE2004 and the SE2014 state that the SWEBOK covers the entire spectrum of software engineering knowledge from novice to expert [IEEE-CS04A, IEEE-CS15]. It is hypothesized that a competency model would be required to extract undergraduate software engineering knowledge from the SWEBOK, and there is no evidence in the updated guidelines to indicate how the SE2015 determined the applicable level of SWEBOK knowledge [IEEE-CS15].

Chapter 3

RESEARCH METHODOLOGY

3.1 Hypotheses

The hypothesis underlying this research is that current graduating baccalaureate software engineering students entering the workforce is are not adequately prepared to define, design, implement or deploy secure software.

This hypothesis is supported by the Department of Homeland Security, which has stated:

> *"It is clear that to produce, acquire, and sustain secure software, a framework that identifies workforce needs for competencies, leverages sound practices, and guides curriculum development for education and training relevant to software assurance is inevitable. Because software quality assurance and software engineering have evolved bodies of knowledge that do not explicitly address security as a quality attribute, a workforce education and training framework must also identify the integration point of secure software development techniques and practices in the existing programs nationwide"* [DHS11].

"Given the scope and potential impact of software defects, it is important to ensure the workforce follows proper software development and sustainment practices. The problem is that there is currently no authoritative point of reference to define what those practices are" [Woody12].

The SE2004 curriculum guidelines were developed by the IEEE-CS and ACM task force over ten years ago. Those guidelines, however, failed to include security knowledge. Since then there has been a demonstrated urgency in producing secure software. There have been a few papers proposing the integration of software assurance into the curriculum guidelines most notably the work done by Dan Shoemaker, Jeff Ingalsbe, Nancy Mead and Antonio Drommi [Shoemaker07] [Shoemaker08] [Shoemaker11]. The software engineering body of knowledge SWEBOK has also been updated to include software security knowledge. Along with the updated core body of knowledge a corresponding software engineering competency model SWECOM has recently been created that describes what software engineering knowledge an undergraduate software engineering student should possess upon graduation. The curriculum guidelines require updating, to include the software security knowledge found in the SWECOM.

This research builds on Shoemaker et al, the SWEBOK and the SWECOM by analyzing the knowledge areas of the curriculum guidelines and the knowledge skills of the SWECOM Chapter 19. Utilizing the IEEE Computer Society and the Department of Homeland Security definitions and descriptions of software assurance knowledge, it is shown where in the curriculum guidelines software assurance knowledge should logically be incorporated. The use of the SWECOM is the logical source of the software assurance knowledge to be integrated into the curriculum guidelines due to the stated goal of the SWECOM to "to prepare an academic or training curriculum for one or more of the SWECOM skills or skill areas to achieve the desired level of competency for each or skill area" [IEEE-CS14].

Because the SE2004 curriculum guidelines do not define specific courses but rather only what should constitute an overall undergraduate software engineering education, [IEEE-CS04A] the analysis is focused primarily on the integration of software security skills and activities of the SWECOM into the SEEK knowledge areas and units. By not focusing on specific courses (of which there are none in the SE2004), academic institutions have the freedom to incorporate and test for, the software security knowledge into their own course materials. This could conceivably be one course or several. In fact many institutions have already integrated security courses, tracks, or full programs into their curriculum.

The methodology is illustrated in Figure 6.

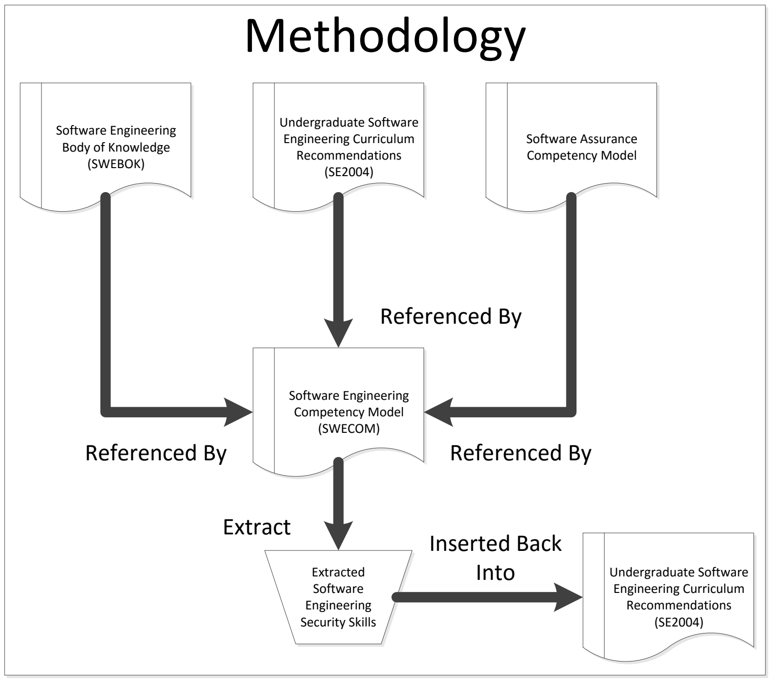
Methodology
Software Engineering Body of Knowledge (SWEBOK)
Undergraduate Software Engineering Curriculum Recommendations (SE2004)
Software Assurance Competency Model
Referenced By
Software Engineering Competency Model (SWECOM)
Referenced By
Referenced By
Extract
Extracted Software Engineering Security Skills
Inserted Back Into
Undergraduate Software Engineering Curriculum Recommendations (SE2004)

Figure 6: Illustrated Methodology

3.2 Methods

The security skill area of the SWECOM are examined and based on the competency level skill-sets and skill-set activities that apply to an undergraduate software engineering education are determined. Because the curriculum guidelines knowledge area topics are described utilizing Bloom's taxonomy, any new topics inserted into the curriculum guidelines would also require a Bloom's taxonomy description, and because the SWECOM does not contain the taxonomy, the Bloom's taxonomy will be determined for any skill set activity to be selected for insertion.

Utilizing the selected competency level skill set activities found in Chapter 19 of the SWECOM, the following will be determined:

- What the SEEK Bloom’s taxonomy of the skill set activity should logically be. i.e.; knowledge, comprehension or application (see Chapter 2 Figure 2).
- What the SEEK Bloom’s taxonomy relevance of the skill set activity should logically be. i.e.; essential, desirable or optional (see Chapter 2 Figure 2).
- Conclude by determining where in the undergraduate software engineering curriculum guidelines SE2004 the resulting security skills should or might logically be incorporated.

Once completed, a method to incorporate the security knowledge from the SWECOM into the curriculum guidelines will be demonstrated.

3.3 Evaluation

It would not be feasible to fully evaluate the outcomes of integrating software assurance into the software engineering curriculum without first performing the integration and measuring and monitoring the vulnerabilities of software produced by graduating software engineering students who have gone through the curricula. It is possible that current software engineering student development projects could be updated to include software security knowledge and then measure the results. This is being done here at the University of North Florida. It should be noted that in response to the continually growing threat, many organizations are implementing more and more vulnerability and security testing measures. These measures should assist companies and organization to effectively evaluate the effectiveness of newly hired software engineering graduates.

Until the software engineering curriculum guidelines are updated with the software assurance knowledge areas, it is recommended that academic institutions and accreditation agencies update their software engineering core courses with the software assurance knowledge areas of the SWECOM.

Academic institutions should determine themselves how best to incorporate the knowledge of software assurance into software engineering courses. These institutions could develop new courses or integrate the knowledge areas into existing courses.
It is recommended that the IEEE-CS and ACM consider updating the curriculum guidelines with the DHS, SWEBOK, SWECOM software assurance knowledge areas and skills.

It is further hypothesized that because producing more secure software has become a national imperative increased introduction of software assurance into the software engineering curricula would significantly better prepare students entering the workforce to produce less vulnerable software.

3.4 Related Work

The proposition of integrating software assurance into a college level curriculum is not new. In their 2008 paper to the 21st Conference on Software Engineering Education and Training Dan Shoemaker, Antonio Drommi, Jeff Ingalsbe and Nancy Mead presented "Integrating Secure Software Assurance Content with SE 2004 Recommendations". The stated purpose of this paper was to identify places in the SE2004 where the Department of Homeland Security's common body of knowledge software assurance knowledge areas [Redwine06], would best fit in a software engineering curricula [Shoemaker08]. This thesis, however, differs in that this paper proposes using the IEEE-CS SWECOM instead of the DHS common body of knowledge.

Dan Shoemaker, Jeff Ingalsbe and Nancy Mead researched the idea again in their 2011 paper "Integrating Software Assurance Knowledge into Conventional Curricula" [Shoemaker11]. This time their paper proposed integrating software assurance into the entire Computing Curricula Series [IEEE-CS05]. Again this thesis differs in in that the scope in this thesis addresses only the software engineering undergraduate degree program.

Elizabeth Hawthorne's paper "Infusing Software Assurance in Computing Curricula" proposed infusing software assurance principles not a specific body of knowledge into a community college associate degree curricula in computer science [Hawthorne12].

It should also be noted that several academic institutions have been and are developing graduate level degree programs in “Master of Software Assurance” [Mead10] such as Carnegie Mellon University, Embry-Riddle Aeronautical University and the University of Nebraska just to name a few.

Chapter 4

RESULTS

4.1 SWECOM

It has been demonstrated that the SE2004 lacks the software assurance guidelines to enable graduating software engineering students to produce secure software, and enable them to compete in the job market. The research showed that competency levels may measure professional knowledge in a discipline such as software engineering; that is, what software engineering knowledge should a technician possess, what software engineering knowledge an entry level practitioner possess. So the logical extension is, what level of software engineering knowledge should a graduating undergraduate software engineering student possess? The SWECOM answers this question by stating an undergraduate software engineering student should possess the equivalent knowledge of an entry level practitioner. The focus of this thesis is software assurance, therefore only the SWECOM software engineering knowledge that deals with software security is addressed; more precisely, the research addresses how only software security knowledge should be included in software engineering curriculum such that a graduating student would possess the competency level tagged, entry level practitioner.

To review, the SWECOM in Chapter 13 Topic 17 breaks down the software security skill area into five skill sets:

- Requirements.
- Design.
- Construction.
- Process.
- Quality.

For each skill set, (such as requirements, design, construction, process and quality) a list of activities is given for each software engineering competency level (see Chapter 2 Table 7). These SWECOM skill set activities will become the SEEK knowledge topics to be inserted into the SE2004 SEEK. As noted in Chapter 2 the SWECOM does not include the Bloom's taxonomy that was included in the SEEK. Therefore the Bloom's taxonomy of the activity will need to be determined before inserting it into the SE2004 SEEK. This thesis will logically incorporate these activities into the curriculum guidelines by determining three things:

- Determine the SEEK Bloom's taxonomy of the activity (See Chapter 2, Figure 2).
- Determine the SEEK Bloom's taxonomy relevance of the activity (See Chapter 2, Figure 2).
- Determine the SEEK knowledge area and unit the activity logically should be inserted (See Chapter 2, Table 2).

4.2 Determine the curriculum guidelines SEEK Bloom's taxonomy

The curriculum guidelines SEEK describes its taxonomy utilizing the Bloom's taxonomy as reiterated here (see Chapter 2, Figure 2) as knowledge, comprehension and application. Knowledge is described as remembering previously learned material. Comprehension is understanding information and the meaning of material presented. Application is the ability to use learned material in a new and concrete situation [IEEE-CS04A].

The SWECOM within the skill sets (see Chapter 2 Table 7) uses the following activity notations to specify what is expected at the specific competency level (See Appendix D). They are:

- Follows (abbreviated as F)
- Assists (abbreviated as A)
- Participates (abbreviated as P)
- Leads (abbreviated as L)
- Creates (abbreviated as C)

Equating the SWECOM activity notations to a SEEK taxonomy the following assertions were made:

1. The SWECOM activity notation of "Follows" indicates the knowledge but not necessarily comprehension or independent application. Thus "Follows" subscribes to the SEEK knowledge taxonomy.

2. The SWECOM activity notation of "Assists" indicates knowledge and comprehension but not independent application, and therefore “Assists” subscribes to the SEEK comprehension taxonomy.

3. The SWECOM activity notation of "Participates" indicates knowledge, comprehension and independent application, and thus “Participates” subscribes the SEEK application taxonomy.

4. The activity notations of "Leads and Creates" indicates knowledge, comprehension and application, and thus this activity subscribes to the SEEK application taxonomy.

5. The asserted activity notation of "Observes" indicates awareness of the knowledge by participation of the next higher competency level, and thus the activity subscribes to the SEEK knowledge taxonomy.

Each SWECOM security skill set will be examined and integrated separately starting with the software security requirements skill set. Table 8 illustrates what the expected activities are at the indicated competency level.

SWECOM Software Security Skills (Appendix D) LEVEL (T=Technician, EL=Entry Level (target), P=Practitioner, TL=Technical Leader, SE=Senior Software Engineer) ACTIVITY (F=Follows, A=Assists, P=Participates, L=Leads, C=Creates, O=Observes)					
Requirements Skill Set	T	EL	P	TL	SE
Identifies security risks (such as misuse cases).		O	P		
Creates requirements that capture security issues.		O	P		
Performs initial threat modeling.		O	P		
Creates or proposes new methods for recognizing security vulnerabilities.					C

Table 8: Security Requirements Skill Set Activities (SWECOM)

The entry level practitioner is not expected to participate in any of these activities. Because the next higher level, practitioner, has activities, it is not unreasonable to expect the entry level practitioner to be an observer to this knowledge. The assertion that "Observes" equates to an activity where the next higher level competency has activities will be taken with all the skill set activities. As can also be seen, the activity of "Creates or proposes new methods for recognizing security vulnerabilities" has no activities or observance at the entry level practitioner level therefore it would be eliminated from inclusion into the SE2004 SEEK. Using the taxonomy assertions made before the resulting SWECOM activities applicable to an entry level practitioner would have a SEEK taxonomy illustrated in Table 9.

SWECOM Software Security Skills SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)	
Requirements Skill Set	**SEEK**
Identifies security risks (such as misuse cases).	K
Creates requirements that capture security issues.	K
Performs initial threat modeling.	K

Table 9: Security Requirements Skills (SWECOM w SEEK Taxonomy)

Table 10 illustrates the SWECOM software security design skill set and what the activities are at the indicated competency level.

SWECOM Software Security Skills (Appendix D) LEVEL (T=Technician, EL=Entry Level (target), P=Practitioner, TL=Technical Leader, SE=Senior Software Engineer) ACTIVITY (F=Follows, A=Assists, P=Participates, L=Leads, C=Creates, O=Observes)					
Design Skill Set	**T**	**EL**	**P**	**TL**	**SE**
Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data).		**F**			
Uses appropriate, secure design patterns.		**F**			
Models threats and associated risks of new and modified systems.		**O**	**P**		
Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems.		**O**	**P**		

Table 10: Security Design Set Activities (SWECOM)

The SWECOM activities with the derived SEEK taxonomy is illustrated in Table 11.

SWECOM Software Security Skills SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)	
Design Skill Set	**SEEK**
Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data).	A
Uses appropriate, secure design patterns.	A
Models threats and associated risks of new and modified systems.	K
Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems.	K

Table 11: Security Design Skills (SWECOM w SEEK Taxonomy)

Table 12 illustrates the SWECOM software security construction skill set and what the activities are at the indicated competency level.

SWECOM Software Security Skills (Appendix D) LEVEL (T=Technician, EL=Entry Level (target), P=Practitioner, TL=Technical Leader, SE=Senior Software Engineer) ACTIVITY (F=Follows, A=Assists, P=Participates, L=Leads, C=Creates, O=Observes)					
Construction Skill Set	**T**	**EL**	**P**	**TL**	**SE**
Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation).		**F**			
Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems).		**F**			
Selects or establishes project coding standards to avoid security vulnerabilities.		**O**	**P**		
Reviews and approves coding standards to avoid security vulnerabilities.		**O**	**P**		
Establishes organization coding standards to avoid security vulnerabilities.				**L**	
Creates new coding standards to avoid security vulnerabilities.					**C**

Table 12: Security Construction Skill Set Activities (SWECOM)

The SWECOM activities with the derived SEEK taxonomy is illustrated in Table 13.

SWECOM Software Security Skills SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)	
Construction Skill Set	**SEEK**
Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation).	C
Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems).	C
Selects or establishes project coding standards to avoid security vulnerabilities.	K
Reviews and approves coding standards to avoid security vulnerabilities.	K

Table 13: Security Construction Skills (SWECOM w SEEK Taxonomy)

Table 14 illustrates the SWECOM software security process skill set and what the activities are at the indicated competency level.

SWECOM Software Security Skills (Appendix D) LEVEL (T=Technician, EL=Entry Level (target), P=Practitioner, TL=Technical Leader, SE=Senior Software Engineer) ACTIVITY (F=Follows, A=Assists, P=Participates, L=Leads, C=Creates, O=Observes)					
Process Skill Set	**T**	**EL**	**P**	**TL**	**SE**
Assists in the collection of metrics for security assessment processes.	**A**				
Follows project standards in the collection of security assessment metrics.		**F**			
Establishes organization standards for security assessment processes.				**L**	

Table 14: Security Process Skill Set Activites (SWECOM)

The SWECOM activities with the derived SEEK taxonomy is illustrated in Table 15.

SWECOM Software Security Skills SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)	
Process Skill Set	**SEEK**
Assists in the collection of metrics for security assessment processes.	C
Follows project standards in the collection of security assessment metrics.	C

Table 15: Security Process Skills (SWECOM w SEEK Taxonomy)

Table 16 illustrates the SWECOM software security quality skill set and what the activities are at the indicated competency level.

SWECOM Software Security Skills (Appendix D) LEVEL (T=Technician, EL=Entry Level (target), P=Practitioner, TL=Technical Leader, SE=Senior Software Engineer) ACTIVITY (F=Follows, A=Assists, P=Participates, L=Leads, C=Creates, O=Observes)					
Quality Skill Set	**T**	**EL**	**P**	**TL**	**SE**
Assists in the installation of static analysis tools.	**A**				
Performs code reviews to identify security vulnerabilities.		**P**			
Uses static analysis methods to identify security vulnerabilities.		**P**			
Selects appropriate static analysis tools to identify security vulnerabilities.		**O**	**P/L**		
Creates new static analysis methods or tools.					**C**

Table 16: Security Quality Skill Set Activities (SWECOM)

The SWECOM activities with the derived SEEK taxonomy is illustrated in Table 17.

SWECOM Software Security Skills SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)	
Quality	**SEEK**
Assists in the installation of static analysis tools.	K
Performs code reviews to identify security vulnerabilities.	C
Uses static analysis methods to identify security vulnerabilities.	A
Selects appropriate static analysis tools to identify security vulnerabilities.	A

Table 17: Security Quality Skills (SWECOM w SEEK Taxonomy)

4.3 Determine the curriculum guidelines SEEK Bloom's relevance

To review the SWECOM skill set activities do not have the SEEK Bloom's taxonomy, which consists of the taxonomy and its relevance. The previous section has determined what the taxonomy should be of all the security skill activities. This section will determine the activities taxonomy relevance to the core.

The curriculum guidelines definition of core is: the core consists of the essential material that professionals teaching software engineering agree is necessary for anyone to obtain an undergraduate degree in this field [IEEE-CS04A].

The SE2004 SEEK describes its Bloom's taxonomy relevance reiterated here (see Chapter 2 Figure 2) as:

- Essential (E) the topic is part of the core.
- Desirable (D) the topic is not part of the core, but it should be included in the core of a particular program if possible; otherwise, it should be considered as part of elective materials.
- Optional (O) the topic should be considered as elective only.

In an effort to determine the relevance of software security skill set activities, the simplest path would have been to determine that all the activities should be desirable or optional. The question as to whether the security knowledge was essential, desired or optional as it related to the core knowledge of software engineering was not researched and only hypothesis that it was at a minimum desired. The resulting effort is illustrated in Table 18.

SWECOM Skills with SE2004 SEEK Taxonomy and Relevance		
SEEK TAXONOMY (K=knowledge, C=comprehension, A=application)		
SEEK TAXONOMY RELAVANCE (E=Essential, D=Desired, O=Optional)		
SWECOM	SEEK	
Software Security Skills	**K/C/A**	**E/D/O**
Requirements Skill Set		
Identifies security risks (such as misuse cases). (P)	**K**	**D**
Creates requirements that capture security issues. (P)	**K**	**D**
Performs initial threat modeling. (P)	**K**	**D**
Design Skill Set		
Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data). (F)	**A**	**D**
Uses appropriate, secure design patterns. (F)	**A**	**D**
Models threats and associated risks of new and modified systems. (P)	**K**	**D**
Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems. (P)	**K**	**D**
Construction Skill Set		
Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). (F)	**C**	**D**
Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems). (F)	**C**	**D**
Selects or establishes project coding standards to avoid security vulnerabilities. (P)	**K**	**D**
Reviews and approves coding standards to avoid security vulnerabilities. (P)	**K**	**D**
Process Skill Set		
Assists in the collection of metrics for security assessment processes. (A)	**C**	**D**
Follows project standards in the collection of security assessment metrics. (F)	**C**	**D**
Quality Skill Set		
Assists in the installation of static analysis tools. (A)	**C**	**D**
Performs code reviews to identify security vulnerabilities. (P)	**A**	**D**
Uses static analysis methods to identify security vulnerabilities. (P)	**A**	**D**
Selects appropriate static analysis tools to identify security vulnerabilities. (P/L)	**K**	**D**

Table 18: Software Security Skills (With SEEK Taxonomy and Relevance)

4.4 Determine the SEEK knowledge unit.

Having determined the SWECOM security knowledge skill activity taxonomy and relevance, the next step is to determine where in the curriculum guidelines this knowledge should logically be inserted.

At this point randomly inserting the SWECOM software security skill sets activities into the SE2004 SEEK would successfully integrate software assurance knowledge into the SE2004 curriculum guidelines. This paper asserts a realistic location in the SE2004 SEEK in which to insert each SWECOM software security skill set activity.

4.4.1 Software Security Requirements Skills

The most logical placement of the SWECOM security requirements skill set activities would be in the SEEK knowledge area that deals with gathering, analyzing, documenting, eliciting or validating requirements. It is asserted that the SEEK knowledge area where these units are found is in the software modeling and analysis SEEK area. More specifically these SEEK units:

- Requirements Fundamentals
- Eliciting Requirements
- Requirements Specification and Documentation
- Requirements Validation

The SE2004 SEEK curriculum guidelines with the inserted SWECOM software security requirements skill set activities is illustrated in Table 19. Note only the inserted topics are shown for each SEEK unit.

Integrated Curriculum Guidelines with Software Assurance	
Software Security Requirements Skills (SWECOM SKILL SET)	
Software Modeling & Analysis (SEEK AREA)	
Requirements fundamentals (SEEK UNIT)	
	Creates requirements that capture security issues. (SWECOM ACTIVITY)
Eliciting requirements (SEEK UNIT)	
	Identifies security risks (such as misuse cases, which are Use Cases that show possible misuse of the system). (SWECOM ACTIVITY)
Requirements specification & documentation (SEEK UNIT)	
	Creates requirements that capture security issues. (SWECOM ACTIVITY)
Requirements validation (SEEK UNIT)	
	Performs initial threat modeling. (SWECOM ACTIVITY)

Table 19: Integrated Security Requirements Skills

4.4.2 Software Security Design Skills

The most logical placement of SWECOM security design skill set activities would be in the SEEK knowledge area that deals with design concepts, strategies, interface design and architectures. It is asserted that the SEEK knowledge area where these units are found is in the software design SEEK area. More specifically these SEEK units:

- Design Concepts
- Design Strategies
- Architectural Design
- Human Computer Interface Design
- Detailed Design

The SE2004 SEEK curriculum guidelines with the inserted SWECOM software security design skill set activities are illustrated in Table 20. Note only the inserted topics are shown for each SEEK unit.

Integrated Curriculum Guidelines with Software Assurance	
Software Security Design Skills (SWECOM SKILL SET)	
Software Design (SEEK AREA)	
Design concepts (SEEK UNIT)	
	Uses appropriate, secure design patterns. (SWECOM ACTIVITY)
Design strategies (SEEK UNIT)	
Architectural design (SEEK UNIT)	
	Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data). (SWECOM ACTIVITY)
Human computer interface design (SEEK UNIT)	
	Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems. (SWECOM ACTIVITY)
Detailed design (SEEK UNIT)	
	Models threats and associated risks of new and modified systems. (SWECOM ACTIVITY)
Design support tools and evaluation (SEEK UNIT)	

Table 20: Integrated Security Design Skills

4.4.3 Software Security Construction Skills

The most logical placement of SWECOM security construction skill set activities would be in the SEEK knowledge area that deals with construction tools and construction methods. It is asserted that the SEEK knowledge area where these units are found is in the computing essentials SEEK area. More specifically these SEEK units:

- Construction Technologies

- Construction Tools

The SE2004 SEEK curriculum guidelines with the inserted SWECOM software construction skill set activities are illustrated in Table 21. Note only the inserted topics are shown for each SEEK unit.

Integrated Curriculum Guidelines with Software Assurance	
Software Security Construction Skills (SWECOM SKILL SET)	
Computing Essentials (SEEK AREA)	
Computer science foundations (SEEK UNIT)	
	Selects or establishes project coding standards to avoid security vulnerabilities. (SWECOM ACTIVITY)
Construction technologies (SEEK UNIT)	
	Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems). (SWECOM ACTIVITY)
Construction tools (SEEK UNIT)	
Formal construction methods (SEEK UNIT)	
	Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). (SWECOM ACTIVITY)
	Reviews and approves coding standards to avoid security vulnerabilities. (SWECOM ACTIVITY)

Table 21: Integrated Security Construction Skills

4.4.4 Software Security Process Skills

The most logical placement of SWECOM security process skill set activities would be in the SEEK knowledge area that deals with process concepts, testing, analyzing and reporting. This thesis asserts that the SEEK knowledge area where these units are found is in the software verification and validation SEEK area. More specifically these SEEK units:

- Testing
- Problem Analysis and Reporting
- Reviews
- Human Computer UI Testing and Evaluation
- V & V Terminology and Foundations

The SE2004 SEEK curriculum guidelines with the inserted SWECOM software process skill set activities are illustrated in Table 22. Again, note only the inserted topics are shown for each SEEK unit.

Integrated Curriculum Guidelines with Software Assurance	
Software Security Process Skills (SWECOM SKILL SET)	
Software Verification and Validation (SEEK AREA)	
V & V Terminology and Foundations (SEEK UNIT)	
	Follows project standards in the collection of security assessment metrics. (SWECOM ACTIVITY)
Reviews (SEEK UNIT)	
	Assists in the collection of metrics for security assessment processes. (SWECOM ACTIVITY)
Testing (SEEK UNIT)	
Human Computer UI Testing and Evaluation (SEEK UNIT)	
Problem Analysis and Reporting (SEEK UNIT)	

Table 22: Integrated Security Process Skills

4.4.5 Software Security Quality Skills

The most logical placement of SWECOM security quality skill set activities would be in the SEEK knowledge area that deals with quality, assurance and standards. It is asserted that the SEEK knowledge area where these units are found is in the software quality SEEK area. More specifically these SEEK units:

- Software Quality Concepts and Culture
- Software Quality Standards
- Software Quality Processes
- Process Assurance
- Product Assurance

The SE2004 SEEK curriculum guidelines with the inserted SWECOM software quality skill set activities are illustrated in Table 23. Only the inserted topics are shown for each SEEK unit.

Integrated Curriculum Guidelines with Software Assurance	
Software Security Quality Skills (SWECOM SKILL SET)	
Software Quality (SEEK AREA)	
Software Quality Concepts and Culture (SEEK UNIT)	
	Selects appropriate static analysis tools to identify security vulnerabilities. (SWECOM ACTIVITY)
Software Quality Standards (SEEK UNIT	
	Performs code reviews to identify security vulnerabilities. (SWECOM ACTIVITY
Software Quality Processes (SEEK UNIT)	
	Uses static analysis methods to identify security vulnerabilities. (SWECOM ACTIVITY)
	Assists in the installation of static analysis tools. (SWECOM ACTIVITY)
Process Assurance (SEEK UNIT)	
Product Assurance (SEEK UNIT)	

Table 23: Integrated Security Quality Skills

This completes the integration of the SWECOM software security skill set activities into the SE2004 curriculum guidelines. The combined listings of Tables 19 through 23 are shown in Appendix E.

To review how integrating software assurance into the SE2004 curriculum guidelines was accomplished the following steps were taken:

1. Determined the source for the software assurance knowledge. The best source was determined to be the SWECOM (see Chapter 2, section 2.5)

2. Integrated only the software security area skill sets;

3. Used the SEEK Bloom’s taxonomy and relevance in the placement of SWECOM skill set activities into the SE2004 SEEK;

4. Inserted the SWECOM software security area skill set activities into the SE2004 SEEK.

An illustration of these steps is given in Figure 7.

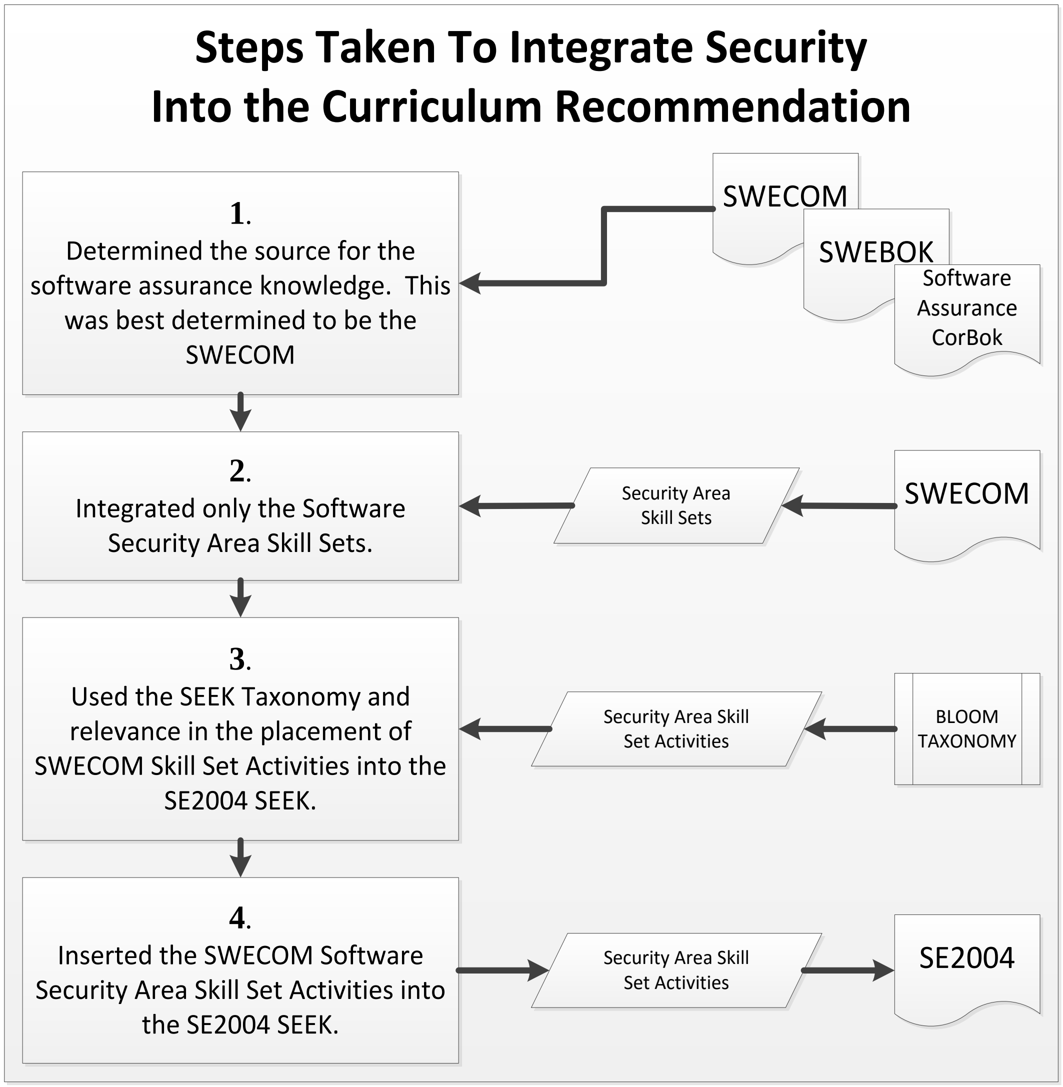
Steps Taken To Integrate Security Into the Curriculum Recommendation
1. Determined the source for the software assurance knowledge. This was best determined to be the SWECOM
SWECOM
SWEBOK
Software Assurance CorBok
2. Integrated only the Software Security Area Skill Sets.
Security Area Skill Sets
SWECOM
3. Used the SEEK Taxonomy and relevance in the placement of SWECOM Skill Set Activities into the SE2004 SEEK.
Security Area Skill Set Activities
BLOOM TAXONOMY
4. Inserted the SWECOM Software Security Area Skill Set Activities into the SE2004 SEEK.
Security Area Skill Set Activities
SE2004

Figure 7: Steps Taken To Integrate Security into the SE2004

Chapter 5

CONCLUSION

Integrating the software assurance topics proved to be fairly straight forward.
The logical association method did indicate possible additional SEEK knowledge area units where the SWECOM activities could be inserted. And the goal of incorporating the software assurance knowledge would still be achieved with any other variation of insertions, as long as the activities were inserted.

This research did not specify lecture lengths of any inserted SE2004 SEEK topic. As indicated in the SWECOM, a competency level of entry level would indicate an introductory level of instruction, and the subject matter would determine the length of time required. Because the SE2004 does not specify individual topic lecture lengths this was not deemed necessary for integration.

Any quantifiable results of incorporating software assurance into the software engineering curriculum could only be measured by actual graduating software engineering students. It is hypothesized that more secure software would be produced.

It should be noted that software assurance is being taught to undergraduate and graduate software engineering students and students in other disciplines at a variety of institutions including the University of North Florida. Advanced degrees are also a possibility as indicated by the Software Engineering Institute (SEI) development of a master's degree in software assurance.

Applying this research by incorporating security skills into existing courses (and new courses) is currently underway within the senior-level computer science software engineering course at UNF. Integrating the security knowledge into other courses within the School of Computing in related majors, such as Information Systems, Information Science, and Information Technology, would entail either supplementing existing security topics or inserting new topics and lessons into existing courses considered "core" for these degree programs.

Where creating an additional core course may not be feasible, it is strongly recommended that software security topics be judiciously inserted into the syllabi of existing core courses to better prepare the student for real world software assurance concerns.

For completeness, current school of computing course offerings within which security components might well logically fit, include:

<u>CEN 4010 Software Engineering:</u> inclusion of topics such as:

Identification of security risks (such as misuse cases i.e. Use-Cases that show application misuse).

Creation of requirements that capture security issues.

Performance of code reviews to identify security vulnerabilities.

Performance of initial threat modeling.

COP 4813 Internet Programming: Inclusion of topics to ensure the developed software:

Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation), and

Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems.

CNT 4504 Computer Networks/Distributed Processing: Include topics that consider:

Model threats and associated risks of new and modified systems.

CDA 4010 Human Factors/Collaborative Computing: Inclusion of topics that ensure the interface design topics

Follow recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation).

CEN 4535C Development of Gaming and Mobile Apps: Include topics that ensure development practices

Use appropriate, secure design patterns, and

Includes code reviews to identify security vulnerabilities.

While undertaking this research with recommendations as articulated, it is significant that the author of this thesis was asked to provide another UNF graduate student and faculty member the recommendations for the incorporation of software assurance knowledge into a developing manuscript on “Secure Software Development”. A few of the methods and techniques suggested were:

<u>Gathering and Validating Security Requirements</u>:

1. Develop misuse cases.
2. Legal ethical obligations.
3. Encryption Requirements.
4. Data at rest requirements (DAR).
5. Data in motion requirements.
6. User Roles.
7. Authentication requirements.
8. Evaluate security risks.

<u>Designing with security in mind</u>:

1. Use threat modeling.
2. Attack surface analysis.
3. Establish secure design requirements.

<u>Developing secure applications</u>:

1. Validate input.

2. Heed compiler warnings.
3. Architect and design for security policies.
4. Keeping it simple.
5. Default deny.
6. Adhere to the principle of least privilege.
7. Sanitize data sent to other software.
8. Practice defense in depth.
9. Adopt a software construction security standard.
10. Perform code reviews.
11. Use approved tools.

Testing for security requirements:

1. Penetration testing.
2. Access control testing.
3. Input validation testing.
4. SQL injection.
5. Cross-site scripting.
6. Check for buffer overflows.
7. Maintain data protection, integrity and availability.
8. Perform dynamic analysis, run-time verification.
9. Fuzz testing.
10. Attack surface reviews.
11. Use effective quality assurance techniques.

<u>Releasing secure applications</u>:

1. Plan and implement a defect tracking process.
2. Conduct final security review.

<u>Maintaining secure applications</u>:

1. Keep application configurations current.

2. Continuously monitor application security.

2. Thoroughly test updates.

3. Analyze attack surface regularly.

There are proven development methodologies than can provide specific methods and processes for example Microsoft’s "Secure Development Lifecycle" (SDL) initiative [Microsoft15].

As stated in Chapter 1, recently late in the development of this thesis, the ACM and IEEE have jointly approved a revised version of the curriculum guidelines. The following is an abbreviated analysis of the differences and similarities between the 2004 and 2014 versions.

As stated in SE2014 "This new version of the curriculum guidelines shares much of the original structure of the previous version" [IEEE-CS15]. A side by side comparison of the changes made to the software engineering educational knowledge is illustrated in Table 24.

OLD		NEW	
SE2004 SEEK		SE2014 SEEK	
Knowledge Area	Title	Knowledge Area	Title
CMP	Computing Essentials	CMP	Computing Essentials
VAV	Software V & V	VAV	Software V & V
FND	Mathematical and Engineering Fundamentals	FND	Mathematical and Engineering Fundamentals
PRF	Professional Practice	PRF	Professional Practice
PRO	Software Process	PRO	Software Process
MAA	Software Modeling and Analysis	MAA	Software Modeling and Analysis
QUA	Software Quality	QUA	Software Quality
DES	Software Design	DES	Software Design
Removed		Added	
EVL	Software Evolution	REQ	Requirements analysis and specification
MGT	Software Management	SEC	Security

Table 24: SE2004 and SE2014 SEEK Comparison

As can be seen the major changes include the removal of the two knowledge areas "Software Evolution" and "Software Management" and the insertion of two new knowledge areas "Security" and "Requirements Analysis and Specification".

The stated source for these newly revised knowledge areas is the Software Engineering Body of Knowledge SWEBOK version 3.0 [IEEE-CS15]. Using the SWEBOK directly as the source of the security knowledge differs from the approach of this paper, which utilized the Software Engineering Competency Model SWECOM. The new SEEK "Security" knowledge area is expanded in Table 25 to show the knowledge units and topics.

SE2014 SEEK Knowledge Areas, Units and Topics						
Knowledge			***Title***	***k,c,a***	***E,D,O***	***Hrs***
Area	***Unit***	***Topic***				
SEC			Security			20
	sfd		Security Fundamentals			4
		1	Information assurance concepts (confidentiality, integrity, and availability)	k	E	
		2	Nature of threats (e.g., natural, intentional, and accidental)	k	E	
		3	Encryption, digital signatures, message authentication, and hash functions	c	E	
		4	Common cryptographic protocols (applications, strengths, and weaknesses)	a	E	
		5	Nontechnical security issues (e.g., social engineering)	c	E	
	net		Computer and Network Security			8
		1	Network security threats and attacks	k	E	
		2	Use of cryptography for network security	k	E	
		3	Protection and defense mechanisms and tools	c	E	
	dev		Developing secure software			8
		1	Building security into the software development life cycle	c	E	
		2	Security in requirements analysis and specification	a	E	
		3	Secure design principles and patterns	a	E	
		4	Secure software construction techniques	a	E	
		5	Security-related verification and validation	a	E	

Table 25: SE2014 Security Knowledge Area [IEEE-CS15]

Chapter 6

FURTHER RESEARCH

Further research into integrating software assurance into the other disciplines of the curricula series information systems, information technology, computer science should be continued. It is indicated in the software assurance competency model that a great deal of the software assurance core body of knowledge overlaps into all computing disciplines. There should also be research into developing a curricula guidelines review and revision process that includes security considerations as a fundamental criterion. Additionally, accreditation of software engineering programs by ABET should be expanded to clearly include security considerations as criteria in assessing software engineering courses.

Ten plus years is too many years to wait to update critical guidelines of a fast-paced discipline such as software engineering. It is beyond time when the infusion of security concerns is essential to software engineering education.

REFERENCES

Print Publications:

[Bloom56]
Bloom, B. S., Ed., *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain*. Longmans, 1956.

[Bourque01]
Bourque, P., R. Dupuis, eds. *Guide to the Software Engineering Body of Knowledge*, IEEE CS Press, 2001.

[Bourque14]
Bourque, P., R.E. Fairley, eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society, 2014; www.swebok.org.

[CNSS15]
Committee on National Security Systems, *Committee on National Security Systems (CNSS) Glossary*, CNSSI No. 4009, April 6, 2015

[DHS11]
Department of Homeland Security, *Software Assurance in Education, Training & Certification Life Cycle Support*, Volume I – (Version 2.2, March 16, 2011)

[Hawthorne12]
E. Hawthorne, *Infusing Software Assurance in Computing Curricula*, ACM Inroads, Volume 3, Number 2, June 2012

[Hilburn13]
Hilburn, T., et al, *Software Assurance Competency Model*, (CMU/SEI-2013-TN-004), Software Engineering Institute, Carnegie Mellon University, March 2013.

[IEEE-CS03A]
IEEE Computer Society (IEEE-CS) & the Association for Computing Machinery (ACM), "First General Review Comments", "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.", Computing Curriculum Series, http://sites.computer.org/ccse/FirstReviewComments7-17-03.pdf (2003).

[IEEE-CS04A]
IEEE Computer Society (IEEE-CS) & the Association for Computing Machinery (ACM), "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.", Computing Curriculum Series, http://sites.computer.org/ccse/SE2004Volume.pdf (2004) (visited December 09, 2015).

[IEEE-CS04B]
IEEE Computer Society (IEEE-CS) & the Association for Computing Machinery (ACM), "Second & Third General Review Comments", "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.", Computing Curriculum Series, http://sites.computer.org/ccse/Review2&3Comments5-1-04.pdf (2004).

[IEEE-CS05]
IEEE Computer Society (IEEE-CS) & the Association for Computing Machinery (ACM), "Computing Curricula 2005: The Overview Report" Computing Curriculum Series. http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf (visited December 09, 2015).

[IEEE-CS14]
IEEE Computer Society (IEEE-CS), Software Engineering Competency Model, (SWECOM), 2014, http://www.computer.org/web/peb/swecom (visited December 09, 2015).

[IEEE-CS15]
IEEE Computer Society (IEEE-CS) & the Association for Computing Machinery (ACM), "Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.", Computing Curriculum Series, February 23, 2015, http://www.acm.org/education/se2014.pdf (visited December 09, 2015).

[Jarzombek06]
Jarzombek, Joe, "Forward" Ed. Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software, Version 1.1. Washington: U.S. DHS, 2006: xiii-xv.

[Mead10]
N. Mead, J. Allen, M. Ardis, T. Hilburn, A. Kornecki, R. Linkger, J. McDonald, (2010) Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum (CMU/SEI-2010-TR-005). Software Engineering Institute, Carnegie Mellon University. available from www.sei.cmu.edu/library/abstracts/reports/10tr005.cfm.

[PITAC05]
President's Information Technology Advisory Committee (PITAC) Report, Cyber Security: A Crisis of Prioritization, February 2005

[Redwine06]
S. T. Redwine, Ed., “Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software, Version 1.1,” U.S. Department of Homeland Security, Washington, 2006.

[Shoemaker07]
D. Shoemaker, A. Drommi, J. Ingalsbe, and N. Mead, “A Comparison of the software Assurance Common Body of Knowledge to Common Curricular Standards” 20th Conference on Software Engineering Education & Training (CSEET’07), Dublin, Ireland, 2007, pp. 149-156.

[Shoemaker08]
D. Shoemaker, A. Drommi, J. Ingalsbe, and N. Mead, “Integrating Secure Software Assurance Content with SE 2004 Recommendations” in Proc.
21st Conf. Software Engineering Education & Training (CSEET’08), Charleston, South Carolina, 2008, pp. 59-66.

[Shoemaker11]
D. Shoemaker, J. Ingalsbe, and N. Mead, Integrating Software Assurance Knowledge into Conventional Curricula, Carnegie Mellon University, CrossTalk: The Journal of Defense Software Engineering February 2011.

[Woody12]
C. Woody, N. Mead, and D. Shoemaker, Foundations for Software Assurance, In proceedings of the 45th Hawaii International Conference on System Science (HICSS), pages 5368-5374, 2012

Electronic Sources:

[BLS15]
Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, 2014-15 Edition, Information Security Analysts, on the Internet at http://www.bls.gov/ooh/computer-and-information-technology/information-security-analysts.htm (visited December 03, 2015).

[Casey15]
Kevin Casey, Information Week: *15 Hot Skill Sets for IT Pros In 2015*, February 17, 2015, http://www.informationweek.com/it-life/15-hot-skill-sets-for-it-pros-in-2015/d/d-id/1319100 (visited December 09, 2015).

[DHS15]
Department of Homeland Security National Cyber Security Division, Build Security In Home, https://buildsecurityin.us-cert.gov/, 2015 (visited December 09, 2015).

[Dubie08]
Dubie, D., Network World: *Wanted: 10 IT skills employers need today,* April 17, 2008, http://www.networkworld.com/article/2278046/computers/wanted--10-it-skills-employers-need-today.html (visited December 09, 2015).

[Execusource14]
“Top 4 Sought after Tech Skills for 2014”, 2014, http://www.execusource.com/top-4-sought-tech-skills-2014 (visited December 09, 2015).

[Gartner14]
“Gartner Says Worldwide Information Security Spending Will Grow Almost 8 Percent in 2014 as Organizations Become More Threat-Aware”, 2014, http://www.gartner.com/newsroom/id/2828722 (visited December 09, 2015).

[McDonald15]
McDonald, Natalie Hope, CompTIA: “Job Outlook: A Candid Conversation about IT Careers, Certifications, Salary and Social Media”, March 15, 2015, https://certification.comptia.org/it-career-news/post/view/2015/03/25/job-outlook-a-candid-conversation-about-it-careers-certifications-salary-and-social-media (visited December 09, 2015).

[Merriam-Webster15]
"Competency." *Merriam-Webster.com*. Merriam-Webster, n.d. Web. 25 Sept. 2015. http://www.merriam-webster.com/dictionary/competency (visited December 09, 2015).

[Microsoft15]
Microsoft, “Security Development Lifecycle”, http://www.microsoft.com/en-us/sdl/default.aspx (visited December 09, 2015).

[Monster15]
Monster.com, “New Year, New Skills: Tech Hiring Trends for 2015”, http://www.monster.com/technology/a/Tech-Hiring-Trends-2015 (visited December 09, 2015).

[Oracle15]
Oracle, “Software Security Assurance”, http://www.oracle.com/us/support/assurance/overview/index.html (visited December 09, 2015).

APPENDIX A

SE2004 Software Engineering Education Knowledge (SEEK).

Knowledge			***Title***	***k,c,a*[1]**	***E,D,O*[2]**	***Hrs*[3]**
Area	***Unit***	***Topic***				
CMP			Computing Essentials			172
	cf		Computer Science Foundations			140
		1	Programming Fundamentals (Control & data, typing, recursion)	a	E	
		2	Algorithms, Data Structures/Representation (static & dynamic) and Complexity	a	E	
		3	Problem solving techniques	a	E	
		4	Abstraction - use and support for (encapsulation, hierarchy, etc.)	a	E	
		5	Computer organization	c	E	
		6	Basic concept of a system	c	E	
		7	Basic user human factors (I/O, error messages, robustness)	c	E	
		8	Basic developer human factors (comments, structure, readability)	c	E	
		9	Programming language basics	a	E	
		10	Operating system basics	c	E	
		11	Database basics	c	E	
		12	Network communication basics	c	E	
		13	Semantics of programming languages		D	
	ct		Construction Technologies			20
		1	API design and use	a	E	
		2	Code reuse and libraries	a	E	
		3	Object-oriented run-time issues (e.g. polymorphism, dynamic binding, etc.)	a	E	
		4	Parameterization and generics	a	E	
		5	Assertions, design by contract, defensive programming	a	E	
		6	Error handling, exception handling, and fault tolerance	a	E	
		7	State-based and table driven construction techniques	c	E	
		8	Run-time configuration and internationalization	a	E	
		9	Grammar-based input processing (parsing)	a	E	
		10	Concurrently primitives (e.g. semaphores, monitors, etc.)	a	E	

	11	Middleware (components and containers)	c	E	
	12	Construction methods for distributed software	a	E	
	13	Constructing heterogeneous (hardware and software) systems; hardware-software design	c	E	
	14	Performance analysis and tuning	k	e	
	15	Platform standards (POSIX etc.)		D	
	16	Test-first programming		D	
tl		Construction Tools			4
	1	Development environments	a	E	
	2	GUI builders	c	E	
	3	Unit testing tools	c	E	
	4	Application oriented languages (e.g. scripting, visual, domain-specific, markup, macros, etc.)	c	E	
	5	Profiling, performance analysis and slicing tools		D	
fm		Formal Construction Methods			8
	1	Application of abstract machines (e.g. SDL, Paisley, etc.)	k	E	
	2	Application of specification languages and methods (e.g. ASM, B, CSP, VDM, Z)	a	E	
	3	Automatic generation of code from a specification	k	E	
	4	Program derivation	c	E	
	5	Analysis of candidate implementations	c	E	
	6	Mapping of a specification to different implementations	k	E	
	7	Refinement	c	E	
VAV		Software V & V			42
fnd		V & V Terminology and Foundations			5
	1	Objectives and constraints of V&V	k	E	
	2	Planning the V&V effort	k	E	
	3	Documenting V&V strategy, including tests and other artifacts	a	E	
	4	Metrics & Measurement (e.g. reliability, usability, performance, etc.)	k	E	
	5	V&V involvement at different points in the lifecycle	k	E	
rev		Reviews			6
	1	Desk Checking	a	E	
	2	Walkthroughs	a	E	
	3	Inspections	a	E	
tst		Testing			21
	1	Unit testing tools	a	E	
	2	Exception handling (writing test cases to trigger exception handling; designing good handling)	a	E	
	3	Coverage analysis and Structure Based Testing (e.g. statement, branch, basis path, multi-condition, dataflow, etc.)	a	E	
	4	Black-box functional testing techniques	a	E	

	5	Integration testing	c	E	
	6	Developing test cases based on use cased and/or customer stories	a	E	
	7	Operation profile-based testing	k	E	
	8	System and acceptance testing	a	E	
	9	Testing across quality attributes (e.g. usability, security, compatibility, accessibility, etc.)	a	E	
	10	Regression Testing	c	E	
	11	Testing tools	a	E	
	12	Deployment process		D	
hct		Human Computer UI Testing and Evaluation			6
	1	The variety of aspects of usefulness and usability	k	E	
	2	Heuristic evaluation	a	E	
	3	Cognitive walkthroughs	c	E	
	4	User testing approaches (observation sessions etc.)	a	E	
	5	Web usability; testing techniques for web sites	c	E	
	6	Formal experiments to test hypotheses about specific HCI controls		D	
par		Problem Analysis and Reporting			4
	1	Analyzing failure reports	c	E	
	2	Debugging/fault isolation techniques	a	E	
	3	Defect analysis	k	E	
	4	Problem tracking	c	E	
FND		Mathematical and Engineering Fundamentals			89
mf		Mathematical Foundations			56
	1	Functions, Relations and Sets	a	E	
	2	Basic Logic (propositional and predicate)	a	E	
	3	Proof Techniques (direct, contradiction, inductive)	a	E	
	4	Basic Counting	a	E	
	5	Graphs and Trees	a	E	
	6	Discrete Probability	a	E	
	7	Finite State Machines, regular expressions	c	E	
	8	Grammars	c	E	
	9	Numerical precision, accuracy and errors	c	E	
	10	Number Theory		D	
	11	Algebraic Structures		O	
ef		Engineering Foundations for Software			23
	1	Empirical methods and experimental techniques (e.g., computer-related measuring techniques for CPU and memory usage)	c	E	
	2	Statistical analysis (including simple hypothesis testing, estimating, regression, correlation etc.)	a	E	
	3	Measurement and metrics	k	E	

		4	Systems development (e.g. security, safety, performance, effects of scaling, feature interaction, etc.)	k	E	
		5	Engineering design (e.g. formulation of problem, alternative solutions, feasibility, etc.)	c	E	
		6	Theory of measurement (e.g. criteria for valid measurement	c	E	
		7	Engineering science for other engineering disciplines (strength of materials, digital system principles, logic design, fundamentals of thermodynamics, etc.)		O	
	ec		Engineering Economics for Software			10
		1	Value considerations throughout the software lifecycle	k	E	
		2	Generating system objectives (e.g. participatory design, stakeholder win-win, quality function deployment, prototyping, etc.)	c	E	
		3	Evaluating cost-effective solutions (e.g. benefits realization, tradeoff analysis, cost analysis, return on investment, etc.)	c	E	
		4	Realizing system value (e.g. prioritization, risk resolution, controlling costs, etc.)	k	E	
EVO			Software Evolution			10
	pro		Evolution Processes			6
		1	Basic concepts of evolution and maintenance	k	E	
		2	Relationship between evolving entities (e.g. assumptions, requirements, architecture, design, code, etc.)	k	E	
		3	Models of software evolution (e.g. theories, laws, etc.)	k	E	
		4	Cost models of evolution		D	
		5	Planning for evolution (e.g. outsourcing, in-house etc.)		D	
	ac		Evolution Activities			4
		1	Working with legacy systems (e.g. use of wrappers, etc.)	k	E	
		2	Program comprehension and reverse engineering	k	E	
		3	System and process re-engineering (technical and business)	k	E	
		4	Impact analysis	k	E	
		5	Migration (technical and business)	k	E	
		6	Refactoring	k	E	
		7	Program transformation		D	
		8	Data reverse engineering		D	
PRF			Professional Practice			35
	psy		Group Dynamics/Psychology			5
		1	Dynamics of working in teams/groups	a	E	
		2	Individual cognition (e.g. limits)	k	E	
		3	Cognitive problem complexity	k	E	
		4	Interacting with stakeholders	c	E	

		5	Dealing with uncertainty and ambiguity	k	E	
		6	Dealing with multicultural environments	k	E	
	com		Communication Skills (Specific to Software Engineering)			10
		1	Reading, understanding and summarizing reading (e.g. source code, documentation)	a	E	
		2	Writing (assignments, reports, evaluations, justifications, etc.)	a	E	
		3	Team and group communication (both oral and written, email, etc.)	a	E	
		4	Presentation skills	a	E	
	pro		Professionalism			20
		1	Accreditation, certification, and licensing	k	E	
		2	Codes of ethics and professional conduct	c	E	
		3	Social, legal, historical and professional issues and concerns	c	E	
		4	The nature and role of professional societies	k	E	
		5	The nature and role of software engineering standards	k	E	
		6	The economic impact of software	c	E	
		7	Employment contracts	k	E	
PRO			Software Process			13
	con		Process Concepts			3
		1	Themes and terminology	k	E	
		2	Software engineering process infrastructure (e.g. personnel, tools, training, etc.)	k	E	
		3	Modeling and specification of software processes	c	E	
		4	Measurement and analysis of software processes	c	E	
		5	Software engineering process improvement (individual, team)	c	E	
		6	Quality analysis and control (e.g. defect prevention, review processes, quality metrics, root cause analysis, etc.)	c	E	
		7	Analysis and modeling of software process models		D	
	imp		Process Implementation			10
		1	Levels of process definition (e.g. organization, project, team, individual, etc.)	k	E	
		2	Life cycle models (agile, heavyweight, waterfall, spiral, V-Model, etc.)	c	E	
		3	Life cycle process models and standards (e.g. IEEE, ISO, etc.)	c	E	
		4	Individual software process (model, definition, measurement, analysis, improvement)	c	E	
		5	Team process (model, definition, organization, measurement, analysis, improvement)	c	E	
		6	Process tailoring	k	E	
		7	Requirements for software life cycle process (e.g., ISO/IEEE Standard 12207)	k	E	
MAA			Software Modeling and Analysis			53

md		Modeling Foundations			19
	1	Modeling principles (e.g. decomposition, abstraction, generalization, projection/views, explicitness, use of formal approaches, etc.)	a	E	
	2	Pre & post conditions, invariants	c	E	
	3	Introduction to mathematical models and specification languages (X, VDM, etc.)	c	E	
	4	Properties of modeling languages	k	E	
	5	Syntax vs. semantics (understanding model representations)	c	E	
	6	Explicitness (make no assumptions, or state all assumptions)	k	E	
tm		Types of Models			12
	1	Information modeling (e.g. entity-relationship modeling, class diagrams, etc.)	a	E	
	2	Behavioral modeling (e.g. structured analysis, state diagrams, use case analysis, interaction diagrams, failure modes and effects analysis, fault tree analysis etc.)	a	E	
	3	Structure modeling (e.g. architectural, etc.)	c	E	
	4	Domain modeling (e.g. domain engineering approaches, etc.)	k	E	
	5	Functional modeling (e.g. component diagrams, etc.)	c	E	
	6	Enterprise modeling (e.g. business processes, organization, goals, etc.)		D	
	7	Modeling embedded systems (e.g. real-time schedule ability analysis, external interface analysis, etc.)		D	
	8	Requirements interaction analysis (e.g. feature interaction, house of quality, viewpoint analysis, etc.)		D	
	9	Analysis Patterns (e.g. problem frames, specification re-use, etc.)		D	
sf		Analysis Fundamentals			6
	1	Analyzing well-form-ness (e.g. completeness, consistency, robustness, etc.)	a	E	
	2	Analyzing correctness (e.g. static analysis, simulation, model checking, etc.)	a	E	
	3	Analyzing quality (non-functional) requirements (e.g. safety, security, usability, performance, root cause analysis, etc.)	a	E	
	4	Prioritization, trade-off analysis, risk analysis, and impact analysis	c	E	
	5	Traceability	c	E	
	6	Formal analysis	k	E	
rfd		Requirements Fundamentals			3
	1	Definition of requirements (e.g. product, project, constraints, system boundary, external, internal, etc.)	c	E	
	2	Requirements process	c	E	

	3	Layers/levels of requirements (e.g. needs, goals, user requirements, system requirements, software requirements, etc.)	c	E	
	4	Requirements characteristics (e.g. testable, non-ambiguous, consistent, correct, traceable, priority, etc.)	c	E	
	5	Managing changing requirements	c	E	
	6	Requirements management (e.g. consistency management, release planning, reuse, etc.)	k	E	
	7	Interaction between requirements and architecture	k	E	
	8	Relationship of requirements to systems engineering, human-centered design, etc.		D	
	9	Wicked problems (e.g. ill-structured; problems with many solutions; etc.)		D	
	10	COTS as a constraint		D	
er		Eliciting Requirements			4
	1	Elicitation Sources (e.g. stakeholders, domain experts, operational and organization environments, etc.)	c	E	
	2	Elicitation Techniques (e.g. interviews, questionnaires/surveys, prototypes, use cases, observation, participatory techniques, etc.)	c	E	
	3	Advanced techniques (e.g. ethnographic, knowledge elicitation, etc.)		O	
rsd		Requirements Specification and Documentation			6
	1	Requirement documentation basics (e.g. types, audience, structure, quality, attributes, standards, etc.)	k	E	
	2	Software requirements specification	a	E	
	3	Specification languages (e.g. structured English, UML, formal languages such as Z, VDM, SCR, RSML, etc.)	k	E	
rv		Requirements Validation			3
	1	Reviews and inspection	a	E	
	2	Prototyping to validate requirements (Summative prototyping)	k	E	
	3	Acceptance test design	c	E	
	4	Validating product quality attributes	c	E	
	5	Formal requirements analysis		D	
QUA		Software Quality			16
cc		Software Quality Concepts and Culture			2
	1	Definitions of quality	k	E	
	2	Society's concern for quality	k	E	
	3	The costs and impacts of bad quality	k	E	
	4	A cost of quality model	c	E	
	5	Quality attributes for software (e.g. dependability, usability, etc.)	k	E	
	6	The dimensions of quality engineering	k	E	

	7	Roles of people, processes, methods, tools, and technology	k	E	
std		Software Quality Standards			2
	1	The ISO 9000 Quality Management Systems	k	E	
	2	ISO/IEE Standard 12207 Software Life Cycle Processes	k	E	
	3	Organizational implementation of standards	k	E	
	4	IEEE software quality-related standards		D	
pro		Software Quality Processes			4
	1	Software quality models and metrics	c	E	
	2	Quality-related aspects of software process models	k	E	
	3	Introduction/overview of ISO 15504 and the SEI CMMs	k	E	
	4	Quality-related process areas of ISO 15504	k	E	
	5	Quality-related process areas of the SQ-CMM and the CMMIs	k	E	
	6	The Baldrige Award criteria as applied to software engineering		O	
	7	Quality aspects of other process models		O	
pca		Process Assurance			4
	1	The nature of process assurance	k	E	
	2	Quality planning	a	E	
	3	Organizing and reporting for process assurance	a	E	
	4	Techniques of process assurance	c	E	
pda		Product Assurance			4
	1	The nature of product assurance	k	E	
	2	Distinctions between assurance and V&V	k	E	
	3	Quality product models	k	E	
	4	Root cause analysis and defect prevention	c	E	
	5	Quality product metrics and measurement	c	E	
	6	Assessment of product quality attributes (e.g. usability, reliability, availability, etc.)	c	E	
DES		Software Design			45
con		Design Concepts			3
	1	Definition of design	c	E	
	2	Fundamental design issues (e.g. persistent data, storage management, exceptions, etc.)	c	E	
	3	Context of design within multiple software development life cycles	k	E	
	4	Context of design principles (information hiding, cohesion and coupling)	a	E	
	5	Interactions between design and requirements	c	E	
	6	Design for quality attributes (e.g. reliability, usability, maintainability, performance, testability, security, fault tolerance, etc.)	k	E	
	7	Design trade-offs	k	E	

	8	Architectural styles, patterns, reuse	c	E	
str		Design Strategies			6
	1	Function-oriented design	a, c	E	
	2	Object-oriented design	c, a	E	
	3	Data-structure centered design		D	
	4	Aspect oriented design		O	
ar		Architectural Design			9
	1	Architectural styles (e.g. pipe-and-filter, layered, transaction-centered, peer-to-peer, publish-subscribe, event-based, client-server, etc.)	a	E	
	2	Architectural trade-offs between various attributes	a	E	
	3	Hardware issues in software architecture	k	E	
	4	Requirements traceability in architecture	k	E	
	5	Domain-specific architectures and product-lines	k	E	
	6	Architectural notations (e.g. architectural structure viewpoints & representations, component diagrams, etc.)	c	E	
hci		Human Computer Interface Design			12
	1	General HCI design principles	a	E	
	2	Use of modes, navigation	a	E	
	3	Coding techniques and visual design (e.g. color, icons, fonts, etc.)	c	E	
	4	Response time and feedback	a	E	
	5	Design modalities (e.g. menu-driven, forms, question-answering, etc.)	a	E	
	6	Localization and internationalization	c	E	
	7	Human computer interface design methods	c	E	
	8	Multi-media (e.g. I/O techniques, voice, natural language, web-page, sound, etc.)		D	
	9	Metaphors and conceptual models		D	
	10	Psychology of HCI		D	
dd		Detailed Design			12
	1	One selected design method (e.g. SSA/SD, JSD, OOD, etc.)	a	E	
	2	Design patterns	a	E	
	3	Component design	a	E	
	4	Component and system interface design	a	E	
	5	Design notations (e.g. class and object diagrams, UML, state diagrams, etc.)	c	E	
ste		Design Support Tools and Evaluation			3
	1	Design support tools (e.g. architectural, static analysis, dynamic evaluation, etc.)	a	E	
	2	Measures of design attribute (e.g. coupling, cohesion, information-hiding, separation of concerns, etc.)	k	E	
	3	Design metrics (e.g. architectural factors, interpretation, metric sets in common use, etc.)	a	E	

	4	Formal design analysis		O	
MGT		Software Management			19
con		Management Concepts			2
	1	General project management	k	E	
	2	Classic management models	k	E	
	3	Project management roles	k	E	
	4	Enterprise/Organizational management structure	k	E	
	5	Software management types (e.g. acquisition, project, development, maintenance, risk, etc.)	k	E	
pp		Project Planning			6
	1	Evaluation and planning	c	E	
	2	Work breakdown structure	a	E	
	3	Task scheduling	a	E	
	4	Effort estimation	a	E	
	5	Resource allocation	c	E	
	6	Risk management	a	E	
per		Project Personnel and Organization			2
	1	Organization structures, positions, responsibilities, and authority	k	E	
	2	Formal/informal communication	k	E	
	3	Project staffing	k	E	
	4	Personnel training, career development, and evaluation	k	E	
	5	Meeting management	a	E	
	6	Building and motivating teams	a	E	
	7	Conflict resolution	a	E	
ctl		Project Control			4
	1	Change control	k	E	
	2	Monitoring and reporting	c	E	
	3	Measurement and analysis of results	c	E	
	4	Correction and recovery	k	E	
	5	Reward and discipline		O	
	6	Standards of performance		O	
cm		Software Configuration Management			5
	1	Revision control	a	E	
	2	Release management	c	E	
	3	Tool support	c	E	
	4	Builds	c	E	
	5	Software configuration management processes	k	E	
	6	Maintenance issues	k	E	
	7	Distribution and backup		D	

APPENDIX B

SEI Software Assurance Competency Model.

KA	Unit		Competency Activities
Assurance Across Lifecycles	Software Lifecycle Processes	L1	Understand and execute the portions of a defined process applicable to the assigned tasks.
		L2	Manage the application of a defined lifecycle software process for a small internal project
		L3	Lead and assess process application for small and medium-sized projects over a variety of lifecycle phases, such as new development, acquisition, operation, and evolution.
		L4	Manage the application of a defined lifecycle software process for a large project, including selecting and adapting existing SwA practices by lifecycle phase.
		L5	Analyze, design, and evolve lifecycle processes that meet the special organizational or domain needs and constraints.
	Software Assurance Processes and Practices	L1	Possess general awareness of methods, procedures, and tools used to assess assurance processes and practices.
		L2	Apply methods, procedures, and tools to assess assurance process and practices
		L3	Manage integration of assurance practices into typical lifecycle phases.
		L4	Lead the selection and integration of lifecycle assurance processes and practices in all projects across an organization.
		L5	Analyze assurance assessment results to determine best practices for various lifecycle phases.
Risk Management	Risk Management Concepts	L1	Understand the basic elements of risk management, including threat modeling.
		L2	Explain how risk analysis is performed.
		L3	Determine the models, process, and metrics to be used in risk management for small internal projects.
		L4	Develop the models, processes, and metrics to be used in risk management of projects of any size.
		L5	Analyze the effectiveness of the use and application of risk management concepts across an organization.
		L1	Describe an organizational risk management process.

	Risk Management Processes	L2	Identify and describe the risks associated with a project.
		L3	Analyze the likelihood, impact, and severity of each identified risk for a project. Plan and monitor risk management for small to medium-sized projects.
		L4	Plan and monitor risk management for a large project.
		L5	Develop a program for analyzing and enhancing risk management practices across an organization.
	Software Assurance Risk Management	L1	Describe risk assessment techniques for threats.
		L2	Apply risk assessment techniques to identified threats.
		L3	Analyze and plan for mitigation of software assurance risks for small systems.
		L4	Analyze and plan for mitigation of software assurance risks for both new and existing systems.
		L5	Assess software assurance processes and practices across an organization and propose improvements.
Assurance Assessment	Assurance Assessment Concepts	L1	Provide tool and documentation support for assurance assessment activities.
		L2	Support assurance assessment activities.
		L3	Apply various assurance assessment methods (such as validation of security requirements, risk analysis, threat analysis, vulnerability assessments and scans, and assurance evidence) to determine if the software/system being assessed is sufficiently secure within tolerances.
		L4	Establish and specify the required or desired level of assurance for a specific software application, set of applications, or software-reliant system.
		L5	Research, analyze, and recommend best practices for assurance assessment methods and techniques.
	Measurement for Assessing Assurance	L1	Provide tool and documentation support for assurance assessment measurement.
		L2	Support assurance assessment measurement activities.
		L3	Implement assurance assessment measurement activities.
		L4	Determine and then analyze the key product and process measurements, and performance indicators that can be used to validate the required level of software assurance; determine which software assurance measurement processes and frameworks might be used to accomplish software assurance integration into lifecycle phases.
		L5	Research, analyze, and recommend best practices for assurance assessment measurement.

Assurance Management	Making the Business Case for Assurance	L1	Understand the need for business case analysis.
		L2	Apply a business case tradeoff analysis to existing data and determine the validity of the case.
		L3	Identify and gather data needed, and produce the business case.
		L4	Perform sophisticated business case analysis.
		L5	Perform research to develop new business case analysis approaches.
	Managing Assurance	L1	Understand the importance of assurance in the lifecycle.
		L2	Support software assurance management tasks.
		L3	Manage small software assurance projects, building in software assurance practices and measurement.
		L4	Manage medium-sized to large projects, building in software assurance practices and measurement.
		L5	Develop new methods for managing assurance.
	Compliance Considerations for Assurance	L1	Understand the importance of compliance and possess awareness of laws and regulations.
		L2	Apply known compliance considerations, laws, and policies to a project.
		L3	Lead compliance activities for a conventional project.
		L4	Lead compliance activities for a complex project, and participate in standards and policy activities.
		L5	Lead standard and policy development activities. Propose new standards and policies.
System Security Assurance	For Newly Developed and Acquired Software for Diverse Applications	L1	Possess knowledge of safety and security risks associated with critical infrastructure systems (e.g., banking and finance, energy production and distribution, telecommunications, and transportation systems).
		L2	Describe the variety of methods by which attackers can damage software or data associated with that software by exploiting weaknesses in the system design or implementation.
		L3	Apply software assurance countermeasures such as layers, access controls, privileges, intrusion detection, encryption, and code review checklists.
		L4	Analyze the threats to which software is most likely to be vulnerable in specific operating environments and domains.
		L5	Perform research on security risks and attack methods, and use it to support modification or creation of techniques used to counter such risks and attacks.
	For Diverse Operational	L1	Possess knowledge of the attacks that have been used to interfere with an application's or system's operations.

	(Existing) Systems	L2	Possess knowledge of how gates, locks, guards, and background checks can address risks.
		L3	Design and plan for access control and authentication.
		L4	Analyze the threats to which software is most likely to be vulnerable in specific operating environments and domains.
		L5	Perform research on security risks and attack methods, and use it to support modification or creation of techniques used to counter such risks and attacks.
	Ethics and Integrity in Creation, Acquisition, and Operation of Software Systems	L1	Possess knowledge of how people who are knowledgeable about attack and prevention methods are obligated to use their abilities, both legally and ethically.
		L2	Possess knowledge of the legal and ethical considerations involved in analyzing a variety of historical events and investigations.
		L3	Follow legal and ethical guidelines in the creation and maintenance of software systems.
		L4	Play a leadership role in the practice of ethical behavior for software security.
		L5	Create new case studies for use in education about ethical and legal issues.
System Functionality Assurance	Assurance Technology	L1	Possess general awareness of technologies used for system functionality assurance.
		L2	Apply assurance technology to determine system functionality assurance.
		L3	Manage integration of selected technology in the functionality assurance process.
		L4	Select and guide decisions on the use of selected technologies for specific projects.
		L5	Analyze assurance technologies and contribute to the development of new ones.
	Assured Software Development	L1	Understand the importance of assurance in the development process.
		L2	Engage in the development tasks contributing to functionality assurance.
		L3	Lead the development of a functionality assurance process in small projects.
		L4	Select and guide decisions on the use of a specific assurance process in large projects.
		L5	Analyze assured development processes and contribute to the development of new ones.

	Assured Software Analytics	L1	Understand the need for using an analytical approach to software development and the use of supporting tools.
		L2	Apply specific selected methods for structured and functional analysis "in the small."
		L3	Lead projects applying specific selected methods for structured and functional analysis "in the large."
		L4	Lead development teams in testing assurance and developing auditable assurance evidence.
		L5	Develop new methods and techniques allowing for testing assurance, and develop auditable assurance evidence.
	Assurance in Acquisition	L1	Understand the need to identify risks in internal software, contracted software, commercial, off-the-shelf (COTS) software, and software as a service (SaaS).
		L2	Define and analyze risks in the acquisition of contracted software, COTS software, and SaaS; employ mitigation tactics to test and identify risks prior to integration.
		L3	Manage multiple supply chains and employ measures to reduce risk in acquisition, and require vendors to employ security measures equal to or greater than internal policy.
		L4	Lead acquisition teams by providing policy, process, tools, and language to prevent the acquisition of insecure software.
		L5	Establish comprehensive policies, plans, and education to L1-L4 personnel, all software development lifecycle stakeholders, and procurement teams to protect against the acquisition of insecure software.
System Operational Assurance	Operational Procedures	L1	Understand the role of business objectives and strategic planning in system assurance.
		L2	Support the creation of security policies and procedures for system operations.
		L3	Create security policies and procedures for the operation of a designated system.
		L4	Define the process and procedures for creating security policies and procedures for the operation of a designated system.
		L5	Research, analyze, and recommend best practices for determining security policies and procedures for system operations.
	Operational Monitoring	L1	Provide support for the installation and use of tools for monitoring and controlling system operation.
		L2	Support the installation and configuration or acquisition of monitors and controls for systems, services, and personnel.

		L3	Evaluate operational monitoring results with respect to system and service functionality and security, and malicious content and application of countermeasures.
		L4	Lead maintenance and evolution of operational systems while preserving assured functionality and security.
		L5	Research, analyze, and recommend best practices for operational monitoring with respect to system and service functionality and security.
	System Control	L1	Provide support for the installation and use of tools for monitoring and controlling system operation.
		L2	Support the implementation of effective responses to operational system accidents, failures, and intrusions.
		L3	Implement effective responses to operational system accidents, failures, and intrusions.
		L4	Lead and plan for effective responses to operational system accidents, failures, and intrusions, including maintenance of business survivability and continuity of operations in adverse environments.
		L5	Research, analyze, and recommend best practices for system control with respect to operational system accidents, failures, and intrusions, including business survivability and continuity of operations in adverse environments.

SEI Software Assurance Competency Model [Hilburn13]

APPENDIX C

Software Engineering Body of Knowledge (SWEBOK)

SWEBOK 2014 Version 3 **Knowledge Areas (KAs)**		
Software Requirements		
Software Design		
Software Construction		
Software Testing		
Software Maintenance		
Software Configuration Management		
Software Engineering Management		
Software Engineering Process		
Software Engineering Models and Methods		
Software Quality		
Software Engineering Professional Practice		
Software Engineering Economics		
Computing Foundations		
Mathematical Foundations		
Engineering Foundations		
Software Requirements		
1	Software Requirements Fundamentals	
	1	Definition of a Software Requirement
	2	Product and Process Requirements
	3	Functional and Non-functional Requirements
	4	Emergent Properties
	5	Quantifiable Requirements
	6	System Requirements and Software Requirements
2	Requirements Process	
	1	Process Models
	2	Process Actors
	3	Process Support and Management
	4	Process Quality and Improvement
3	Requirements Elicitation	
	1	Requirements Sources

	2	Elicitation Techniques
4	Requirement Analysis	
	1	Requirements Classification
	2	Conceptual Modeling
	3	Architectural Design and Requirements Allocation
	4	Requirements Negotiation
	5	Formal Analysis
5	Requirements Specification	
	1	System Definition Document
	2	System Requirements Specification
	3	Software Requirements Specification
6	Requirements Validation	
	1	Requirements Reviews
	2	Prototyping
	3	Model Validation
	4	Acceptance Tests
7	Practical Considerations	
	1	Iterative Nature of Requirements Process
	2	Change Management
	3	Requirements Attributes
	4	Requirements Tracking
	5	Measuring Requirements
8	Software Requirements Tools	
Software Design		
1	Software Design Fundamentals	
	1	General design concepts
	2	Context of software design
	3	Software design process
	4	Software Design Principles
2	Key Issued in Software Design	
	1	Concurrency
	2	Control and handling of Events
	3	Data Persistence
	4	Distribution of Components
	5	Error and exception handling and fault tolerance
	6	Interaction and Presentation
	7	Security
3	Software Structure and Architecture	
	1	Architectural structures and viewpoints

	2	Architectural Styles
	3	Design Patterns
	4	Architecture Design Decisions
	5	Families of Programs and Frameworks
4	User Interface Design	
	1	General User Interface Design Principles
	2	User Interface Design Issues
	3	The Design of User Interaction Modalities
	4	The Design of Information Presentation
	5	User Interface Design Process
	6	Localization and Internationalization
	7	Metaphors and Conceptual Models
5	Software Design Quality Analysis and Evaluation	
	1	Quality Attributes
	2	Quality Analysis and Evaluation Techniques
	3	Measures
6	Software Design Notations	
	1	Structural Descriptions (Static View)
	2	Behavior Descriptions (Dynamic View)
7	Software Design Strategies and Methods	
	1	General Strategies
	2	Function-Oriented (Structured) Design
	3	Object-Oriented Design
	4	Data-Structure Centered Design
	5	Component-Based Design (CBD)
	6	Other Methods
8	Software Design Tools	
Software Construction		
1	Software Construction Fundamentals	
	1	Minimizing Complexity
	2	Anticipating Change
	3	Constructing for Verification
	4	Reuse
	5	Standards in Construction
2	Managing Construction	
	1	Construction in Life Cycle Models
	2	Construction Planning
	3	Construction Measurement
3	Practical Considerations	

	1	Construction Design
	2	Construction Languages
	3	Coding
	4	Construction Testing
	5	Construction for Reuse
	6	Construction with Reuse
	7	Construction Quality
	8	Integration
4	Construction Technologies	
	1	API Design and Use
	2	Object-Oriented Runtime Issues
	3	Parameterization and Generics
	4	Assertions, Design by Contract, and Defensive Programming
	5	Error Handling, Exception Handling, and Fault Tolerance
	6	Executable Models
	7	State-Based and Table-Driven Construction Techniques
	8	Runtime Configuration and Internalization
	9	Grammar-Based Input Processing
	10	Concurrency Primitives
	11	Middleware
	12	Construction Methods for Distributed Software
	13	Constructing Heterogeneous Systems
	14	Performance Analysis and Tuning
	15	Platform Standards
	16	Test-First Programming
5	Software Construction Tools	
	1	Development Environments
	2	GUI Builders
	3	Unit Testing Tools
	4	Profiling, Performance Analysis, and Slicing Tools
Software Testing		
1	Software Testing Fundamentals	
	1	Testing-Related Terminology
	2	Key Issues
	3	Relationships of Testing to Other Activities
2	Test Levels	
	1	The Target of the Test
	2	Objectives of Testing
3	Test Techniques	
	1	Based on the Software Engineer's Intuition and Experience

	2	Input Domain-Based Techniques
	3	Code-Based Techniques
	4	Fault-Based Techniques
	5	Usage-Based Techniques
	6	Model-Based Techniques
	7	Techniques Based on the Nature of the Application
	8	Selecting and Combining Techniques
4	Test Related Measures	
	1	Evaluation of the Program Under Test
	2	Evaluation of the Tests Performed
5	Test Process	
	1	Practical Considerations
	2	Test Activities
6	Software Testing Tools	
	1	Testing Tool Support
	2	Categories of Tools
Software Maintenance		
1	Software Maintenance Fundamentals	
	1	Definitions and Terminology
	2	Majority of Maintenance Costs
	3	The Nature of Maintenance
	4	Evolution of Software
	5	Need for Maintenance
	6	Categories of Maintenance
3	Maintenance Process	
	1	Maintenance Processes
	2	Maintenance Activities
2	Key Issues in Software Maintenance	
	1	Technical Issues
	2	Management Issues
	3	Maintenance Cost Estimation
	4	Software Maintenance Measurement
4	Techniques for Maintenance	
	1	Program Comprehension
	2	Re-engineering
	3	Reverse Engineering
	4	Migration
	5	Retirement
5	Software Maintenance Tools	

Software Configuration Management		
1	Management of the SCM Process	
	1	Organizational Context for SCM
	2	Constraints and Guidance for SCM
	3	Planning for SCM
	4	SCM Plan
	5	Surveillance of SCM
2	Software Configuration Identification	
	1	Identifying Items to be Controlled
	2	Software Library
3	Software Configuration Control	
	1	Requesting, Evaluating and Approving Software Changes
	2	Implementing Software Changes
	3	Deviations and Waivers
4	Software Configuration Status Accounting	
	1	Software Configuration Status Information
	2	Software Configuration Status Reporting
5	Software Configuration Auditing	
	1	Software Functional Configuration Audit
	2	Software Physical Configuration Audit
	3	In-Process Audits of a Software Baseline
6	Software Release Management and Delivery	
	1	Software Building
	2	Software Release Management
7	Software Configuration Management Tools	
Software Engineering Management		
1	Initiation and Scope Definition	
	1	Determination and Negotiation of Requirements
	2	Feasibility Analysis
	3	Process for the Review and Revision of Requirements
2	Software Project Planning	
	1	Process Planning
	2	Determine Deliverables
	3	Effort, Schedule and Cost Estimation
	4	Resource Allocation
	5	Risk Management
	6	Quality Management
	7	Plan Management
3	Software Project Enactment	

	1	Implementation of Plans
	2	Software Acquisition and Supplier Contract Management
	3	Implementation of Measurement Process
	4	Monitor Process
	5	Control Process
	6	Reporting
4		Review and Evaluations
	1	Determining Satisfaction of Requirements
	2	Reviewing and Evaluating Performance
5		Closure
	1	Determining Closure
	2	Closure Activities
6		Software Engineering Measurement
	1	Establish and Sustain Measurement Commitment
	2	Plan the Measurement Process
	3	Perform Measurement Process
	4	Evaluate Measurement
7		Software Engineering Management Tools
Software Engineering Process		
1		Software Process Definition
	1	Software Process Management
	2	Software Process Infrastructure
2		Software Life Cycles
	1	Categories of Software Processes
	2	Software Life Cycle Models
	3	Software Process Adaptation
	4	Practical Considerations
3		Software Process Assessment and Improvement
	1	Software Process Assessment Models
	2	Software Process Assessment Methods
	3	Software Process Improvement Models
	4	Continuous and Staged Software Process Ratings
4		Software Measurement
	1	Software Process and Product Measurement
	2	Quality of Measurement Results
	3	Software Information Models
	4	Software Process Measurement Techniques
5		Software Engineering Process Tools
Software Engineering Models and Methods		

1		Modeling
	1	Modeling Principles
	2	Properties and Expression of Models
	3	Syntax, Semantics, and Pragmatics
	4	Preconditions, Post conditions, and Invariants
2		Types of Models
	1	Information Modeling
	2	Behavioral Modeling
	3	Structure Modeling
3		Analysis of Models
	1	Analyzing for Completeness
	2	Analyzing for Consistency
	3	Analyzing for Correctness
	4	Traceability
	5	Interaction Analysis
4		Software Engineering Methods
	1	Heuristic Methods
	2	Formal Methods
	3	Prototyping Methods
	4	Agile Methods
Software Quality		
1		Software Quality Fundaments
	1	Software Engineering Culture and Ethics
	2	Value and Costs of Quality
	3	Models and Quality Characteristics
	4	Software Quality Improvement
	5	Software Safety
2		Software Quality Management Processes
	1	Software Quality Assurance
	2	Verification and Validation
	3	Reviews and Audits
3		Practical Considerations
	1	Software Quality Requirements
	2	Defect Characterization
	3	Software Quality Management Techniques
	4	Software Quality Measurement
4		Software Quality Tools
Software Engineering Professional Practice		
1		Professionalism

	1	Accreditation, Certification, and Licensing
	2	Codes of Ethics and Professional Conduct
	3	Nature and Role of Professional Societies
	4	Nature and Role of Software Engineering Standards
	5	Economic Impact of Software
	6	Employment Contracts
	7	Legal Issues
	8	Documentation
	9	Tradeoff Analysis
2	Group Dynamics and Psychology	
	1	Dynamics of Working in Teams/Groups
	2	Individual Cognition
	3	Dealing with Problem Complexity
	4	Interacting with Stakeholders
	5	Dealing with Uncertainty and Ambiguity
	6	Dealing with Multicultural Environments
3	Communication Skills	
	1	Reading, Understanding and Summarizing
	2	Writing
	3	Team and Group Communication
	4	Presentation Skills
Software Engineering Economics		
1	Software Engineering Economics Fundamentals	
	1	Finance
	2	Accounting
	3	Controlling
	4	Cash Flow
	5	Decision-Making Process
	6	Valuation
	7	Inflation
	8	Depreciation
	9	Taxation
	10	Time-Value of Money
	11	Efficiency
	12	Effectiveness
	13	Productivity
2	Life Cycle Economics	
	1	Product
	2	Project
	3	Program

	4	Portfolio
	5	Product Life Cycle
	6	Project Life Cycle
	7	Proposals
	8	Investment Decisions
	9	Planning Horizon
	10	Price and Pricing
	11	Cost and Costing
	12	Performance Measurement
	13	Earned Value Management
	14	Termination Decisions
	15	Replacement and Retirement Decisions
3	Risk and Uncertainty	
	1	Goals, Estimates and Plans
	2	Estimation Techniques
	3	Addressing Uncertainty
	4	Prioritization
	5	Decisions Under Risk
	6	Decisions Under Uncertainty
4	Economic Analysis Methods	
	1	For-Profit Decision Analysis
	2	Minimum Acceptable Rate of Return
	3	Return on Investment
	4	Return on Capital Employed
	5	Cost-Benefit Analysis
	6	Cost-Effectiveness Analysis
	7	Break-Even Analysis
	8	Business Case
	9	Multiple Attribute Evaluation
	10	Optimization Analysis
5	Practical Considerations	
	1	The "Good Enough" Principle
	2	Friction-Free Economy
	3	Ecosystems
	4	Offshoring and Outsourcing
Computing Foundations		
1	Problem Solving Techniques	
2	Abstraction	
3	Programming Fundamentals	
4	Programming Language Basics	

5	Debugging Tools and Techniques
6	Data Structure and Representation
7	Algorithms and Complexity
8	Basic Concept of a System
9	Computer Organization
10	Operating System Basics
11	Compiler Basics
12	Database Basics and Data Management
13	Network Communication Basics
14	Parallel and Distributed Computing
15	Basic User Human Factors
16	Basic Developer Human Factors
17	Secure Software Development and Maintenance
Mathematical Foundations	
1	Set, Relations, Functions
2	Basic Logic
3	Proof Techniques
4	Basics of Counting
5	Graphs and Trees
6	Discrete Probability
7	Finite State Machines
8	Grammars
9	Numerical Precision, Accuracy and Errors
10	Number Theory
11	Algebraic Structures
Engineering Foundations	
1	Empirical Methods and Experimental Techniques
2	Statistical Analysis
3	Measurement
4	Engineering Design
5	Modeling, Simulation and Prototyping
6	Standards
7	Root Cause Analysis

Software Engineering Body of Knowledge (SWEBOK) [Bourque14]

APPENDIX D

The SWECOM Skill Areas, Sets and Activities by competency level.

SOFTWARE REQUIREMENTS SKILLS	
Software Requirements Elicitation	
Technician	
1	Assists requirements engineers with preparation of surveys and other elicitation instruments. (F/A)
Entry Level	
1	Assists in engaging different stakeholders to determine needs and requirements. (A)
2	Assists in applying different methods to the project as appropriate to elicit requirements. (A)
Practitioner	
1	Identifies important stakeholders. (P/L)
2	Engages different stakeholders to determine needs and requirements. (P)
3	Applies different methods to the project as appropriate to elicit requirements. (P)
4	Assists in negotiating conflicts between stakeholders in requirements elicitation. (A)
Technical Leader	
1	Selects appropriate methods to engage and communicate with stakeholders in requirements activities. (L)
2	Negotiates conflicts between stakeholders in requirements elicitation. (P/L)
Senior Software Engineer	
1	Creates new ways to engage and communicate with stakeholders, the management team, and developers in requirements activities. (C)
Software Requirements Analysis	
Technician	
Entry Level	
1	Assists in domain analysis. (A)
Practitioner	
1	Selects the most appropriate domain analysis methods. (P/L)
2	Identifies emergent properties and requirements throughout the software development life cycle. (P)
Technical Leader	
1	Leads identification of emergent properties and requirements throughout the softwre development life cycle. (P)
Senior Software Engineer	

1	Creates new domain analysis methods. (C)
Software Requirements Specification	
Technician	
1	Assists with preparation of requirements for consistency with internal and published standards. (F/A)
Entry Level	
1	Prepares requirements documentation including descriptions of interfaces and functional and non-functional requirements. (P)
Practitioner	
1	Selects the most appropriate formal and informal notations for describing interfaces and functional and non-functional requirements. (P/L)
Technical Leader	
1	Leads development of the SRS. (L)
2	Selects the most appropriate formal and informal notations for describing interfaces and functional and non-functional requirements. (L)
Senior Software Engineer	
1	Creates new requirements specification methods. (C)
Software Requirements Verification and Validation	
Technician	
1	Assists in prototype construction and testing. (F/A)
Entry Level	
2	Reviews specifications of requirements for errors and omissions. (P)
Practitioner	
1	Reviews specifications of requirements for errors and omissions. (L)
2	Creates prototypes of different types as needed. (P)
3	Assists in negotiating conficts between stakeholders in requirements verification. (A)
Technical Leader	
1	Selects the most appropriate formal and informal requirements validation and verification techniques. (L)
2	Negotiates conflicts between stakeholders in requirements verification. (P/L)
Senior Software Engineer	
1	Creates new requirements validation and verification techniques. (C)
Software Requirements Elicitation	
Technician	
1	Follows and applies defined processes for requirements engineering with guidance. (F/A)
Entry Level	
1	Assists in applying defined processes for requirements engineering. (A)
Practitioner	
1	Performs tradeoff analysis of requirements activities. (P/L)
Technical Leader	
Senior Software Engineer	

1	Sets strategy and direction for the requirements process across projects and functional units of an organization. (L)
SOFTWARE DESIGN SKILLS	
Software Design Fundamentals	
Technician	
1	Assists software designers with tools and techniques for gathering information about applications and use of software design fundamentals. (F/A)
Entry Level	
1	Assists in the application of enabling techniques in the design of software components and modules. (A)
2	Assists in the application of design techniques in the areas of concurrency, event handling, data persistence, or distributed software. (A)
3	Assists in the application of exception handling and tault tolerance techniques in the design of software components and modules. (A)
4	Assists in the use of restructuring and refactoring methods in the design of sfotware components and modules. (A)
Practitioner	
1	Applies enabling techniques (such as abstraction, coupling/cohesion, information hiding, and so forth) to the design of software components and modules. (P)
2	As appropriate in the domain of application, applies appropriate design techniques in the areas of concurrency, event handling, data persistence, or distributed software. (P)
3	Applies exception handling and fault tolerance techniques in the design of software components and modules. (P)
4	Uses restructuring and refactoring methods in the design of software components and modules. (P)
Technical Leader	
1	Evaluates the effectiveness of the application of software design enabling techniques. (P/L)
2	Provides direction and advice on methods and techniques to be used in the areas of concurrency, event handling, or distributed software. (L)
Senior Software Engineer	
1	Analyzes and makes recommendations related to organization-wide application of software design fundamentals. (C)
Software Design Strategies and Methods	
Technician	
1	Provides assistance in the installation and use of tools appropriate for a project's designated design strategy and methodology (such as an incremental object-oriented approach). (F/A)
Entry Level	
1	Assists in the application of the designeated software design strategy and methodology to create a software design (such as an incremental object oriented approach). (A/P)

Practitioner	
1	Applies the designated software design strategy and methodology to create a software design (such as an incremental object-oriented approach). (P)
2	Determines design alternatives and performs trade-off analysis. (P/L)
Technical Leader	
1	Determines the processes and strategy to be used in software design at the project level (such as top-down or bottom-up, step-wise refinement, use of patterns and pattern languages, iterative and incremental processes, and so forth). (L)
2	Selects the appropriate design methodology (such as object-oriented, function-oriented, component based) and strategies to be used at the project level. (L)
3	Provides guidance and advice on the use of software design strategies and methods. (L)
4	Evaluates the effectiveness of the application of the selected software design methodology. (P/L)
5	Determines design alternatives and performs trade-off analysis. (L)
Senior Software Engineer	
1	Examines and assesses the effectiveness, across an organization, of the application of software design strategies and methods. (C)
2	Analyzes and makes recommendations related to organization-wide software design strategies and methodologies. (C)
3	Creates new techniques evaluating software design quality. (C)
Software Architectural Design	
Technician	
1	Provides assistance in the installation and use of software architecture tools. (F/A)
Entry Level	
1	Assists in architectural design tasks associated with use of standard notations, diagramming techniques, models, and patterns. (A)
2	Applies a selected software design pattern to the design of software component or module. (A/P)
Practitioner	
1	Applies standard notations, diagramming techniques, models, and patterns (such as architectural styles, structural and behavioral models, GoF patterns, structured systems design models, and UML) to model the high-level organization of a software system. (P/L)
2	Creates multiple views of the software system. (P/L)
3	Uses design patterns and frameworks to design mid-level software components or modules. (P)
Technical Leader	
1	Provides direction and advice on standard notations, diagramming techniques, models, and patterns to be applied. (L)
2	Evaluates the effectiveness of the creation of software architecture. (P/L)
Senior Software Engineer	

1	Analyzes and makes recommendations related to organization-wide software architectural design. (C)
2	Determines new methods and techniques to be used in architectural design. (C)
Software Design Quality Analysis and Evaluation	
Technician	
1	Assists software designers with tools and techniques for collecting design metrics and evaluating software design quality. (F/A)
Entry Level	
1	Participates in software design reviews. (P)
2	Carries out static analysis tasks to evaluate design quality. (P)
3	Assists in development and use of simulation and prototypes to evaluate software design quality. (A)
Practitioner	
1	Facilitates software design reviews. (P/L)
2	Leads static analysis tasks to evaluate design quality. (P/L)
3	Develops and uses simulation and prototypes to evaluate software design quality. (P)
Technical Leader	
1	Selects appropriate tools and techniques (such as design reviews, static analysis, simulation and prototyping, design metrics) to ensure a software design's quality. (L)
2	Uses the results of software design quality evaluation activities to assess the quality of the design and to decide on corrective action, if needed. (P/L)
3	Provides guidance and direction related to the need for requirements change resulting from design review. (P/L)
Senior Software Engineer	
1	Analyzes and makes recommendations related to organization-wide design quality evaluation and analysis techniques. (C)
2	Creates new techniques for evaluating software design quality. (C)
SOFTWARE CONSTRUCTION SKILLS	
Software Construction Planning	
Technician	
Entry Level	
Practitioner	
1	Assists in the selection of appropriate processes and models for software construction. (F/P)
2	Assists in the selection of appropriate languages and tools for software construction. (F/P)
3	Assists in the selection of appropriate frameworks, platforms, and environments for software construction. (F/P)
Technical Leader	
1	Selects appropriate processes and models for constructing software on individual projects (such as compilation, generation from domain-specific languages). (L)

2	Selects appropriate languages and tools for software construction on individual projects. (L)
3	Selects appropriate frameworks, platforms, and environments for individual projects. (L)
Senior Software Engineer	
1	Creates or proposes new processes and models for software construction. (C)
2	Creates new languages and tools for software construction. (C)
3	Creates or proposes new frameworks, platforms, and environments. (C)
Managing Software Construction	
Technician	
1	Assists in the installation of tools and repositories for version control and configuration management. (A)
Entry Level	
1	Uses standard tools and processes for version control and configuration management. (P)
2	Collects standard measures of code quality and size. (P)
Practitioner	
1	Monitors standard measures of code quality and size. (P)
Technical Leader	
1	Establishes project standards for version control and configuration management. (L)
Senior Software Engineer	
1	Establishes organization standards for version control and configuration management. (L)
2	Sets organization standards for code quality and size. (L)
3	Creates new tools and processes for version control and configuration management. (C)
Detailed Design and Coding	
Technician	
1	Assists in the installation of tools and repositories for design and coding. (A)
Entry Level	
1	Creates code to implement detailed designs. (P)
2	Refactors code when needed. (P)
3	Follows project and organization standards for designs and code. (F)
4	Uses appropriate design patterns. (P)
5	Uses defensive coding techniques to minimize propagation of errors and threats. (P)
6	Documents code through comments to support software maintenance. (P)
7	Generates code and systems from models (such as UML) as appropriate. (P)
Practitioner	
1	Creates and reviews detailed designs and code that meet quality requirements. (P)
2	Suggests and performs appropriate code refactoring when needed. (L)
3	Selects or establishes project standards for designs and code. (P)

4	Suggests and uses appropriate design patterns. (L)
5	Suggests and uses defensive coding techniques to minimize propagation of errors and threats. (L)
6	Writes executable models suitable for code generation as appropriate. (P/L)
Technical Leader	
1	Measures and monitors an organization's use of design patterns. (L)
2	Plans and initiates model-driven development processes as appropriate. (L)
Senior Software Engineer	
1	Establishes organization standards for detailed designs and code. (L)
2	Creates new design patterns. (C)
Debugging and Testing	
Technician	
1	Assists in the installation of tools for debugging and testing. (A)
Entry Level	
1	Uses appropriate tools and techniques for debugging. (P)
2	Creates and executes unit tests for all delivered code. (P)
3	Achieves test coverage goals set by project and organization standards. (P)
Practitioner	
1	Ensures project standards for unit test coverage are followed. (P)
Technical Leader	
1	Establishes project standards for unit test coverage. (L)
2	Selects appropriate debugging tools and techniques for a project. (L)
Senior Software Engineer	
1	Establishes organization standards for unit testing. (L)
2	Creates new unit testing tools and methods. (C)
Integrating and Collaborating	
Technician	
1	Assists in installation of integration tools. (A)
2	Assists in creation of code inspection packages. (A)
3	Assists in scheduling code inspections. (A)
4	Sets up build-and-install environments where the software packages can be integrated. (P)
Entry Level	
1	Follows project integration strategy and processes. (P)
2	Performs integration testing as part of the integration process. (P)
3	Collaborates with other team members in development activities (such as pair programming, informal reviews). (P)
4	Participates in project-defined reviews and inspections. (P)
Practitioner	
1	Leads code reviews and inspections. (L)
Technical Leader	
1	Assists in selection of project tools and processes for integration. (P)

Senior Software Engineer	
1	Establishes organization standards for integration tools and processes. (L)
2	Establishes organization standards for reviews and inspections. (L)
3	Creates new integration tools and processes. (C)
4	Creates new code review and inspection methods. (C)
SOFTWARE TESTING SKILLS	
Software Test Planning	
Technician	
Entry Level	
1	Identifies unit and integration testing success and failure criteria. (P)
2	Follows software test plan. (P)
3	Establishes the criteria for unit test execution completion, such as code coverage, defect intensity, and so forth. (P)
4	Develops unit test plan. (P)
5	Assists with the development of the test plan. (A)
Practitioner	
1	Identifies stakeholders participating in demonstration and testing activities. (P)
2	Designs and implements a software test plan. (P)
3	Identifies success and failure criteria for testing. (L/P)
4	Develops demonstration or other test plans. (P)
5	Establishes criteria for demonstration readiness. (A/P)
6	Selects the most appropriate demonstration, testing technique. (P)
7	Establishes the criteria for test completion, such as defect arrival rate, defect intensity, and so forth. (P)
8	Establishes criteria for regression testing, such as defect density and so forth. (P/L)
Technical Leader	
1	Establishes organizational procedures for testing. (P)
2	Identifies customer representatives and other stake holders participating in the acceptance testing and demonstrations. (L)
3	Identifies project test objectives. (L)
4	Identifies success and failure criteria for system and acceptance testing. (P)
5	Identifies test completion criteria. (P)
Senior Software Engineer	
1	Establishes organizational procedures for testing. (L)
Software Testing Infrastructure	
Technician	
1	Sets up the necessary test and demonstration environment. (P)
2	Installs the necessary tools. (P)
3	Develops the appropriate infrastructure for data warehousing. (P)
Entry Level	

1	Selects appropriate unit test techniques. (P)
2	Selects the most appropriate testing tools. (P)
3	Designs and implements the test environment. (P)
Practitioner	
1	Defines the necessary setup for testing and demonstration. (P/L)
2	Identifies the appropriate documentations necessary for the testing activities. (P)
3	Designs the test environment. (L/P)
Technical Leader	
1	Identifies testing tools and test data warehousing across projects. (P)
Senior Software Engineer	
1	Identifies organizational testing tools and data warehousing across projects. (L)
2	Creates new test documentation (in other words, test plan, defect recording, and so forth). (L/P)
Software Testing Techniques	
Technician	
1	Performs manual test activities (in other words, data entry, test case execution, and so forth). (P)
2	Executes regression testing. (P)
3	During demonstration, monitors customer use and records customer feedback for product improvement. (A)
Entry Level	
1	Designs and executes unit test cases. (P)
2	Assists with the development of the test cases. (P)
3	Executes integration and system test cases. (P)
4	Ensures the system is ready for demonstration by performing acceptance test. (P/L)
Practitioner	
1	Specifies appropriate test cases for the selected testing technique. (L/P)
2	Executes test cases. (L/P)
3	Develops automated test case scenarios. (L/P)
Technical Leader	
1	Designs system test plan and test cases. (L)
2	Identifies automated testing opportunities. (L/P)
Senior Software Engineer	
1	Creates new testing (in other words, unit, integration, stress, and so forth) techniques. (C)
Software Testing Measurement and Defect Tracking	
Technician	
1	Performs all appropriate data warehousing (gathering execution data, data entry, data archiving, and so forth). (P)
2	Generates appropriate reports associated with test / demonstration execution. (P)
3	Collects appropriate data associated with executing test cases. (P)

4	Provides test result report to appropriate stakeholders. (P)
Entry Level	
1	Collects appropriate data associated with test execution. (P)
2	Evaluates test execution results and identifies appropriate rework. (P)
3	Monitors test progress by evaluating defect arrival rate, failure intensity, and so forth. (A/P)
Practitioner	
1	Collects appropriate data associated with test execution. (L/P)
2	Conducts root cause analysis. (P)
3	Using the test results, assigns appropriate rework to team members. (P)
4	Uses test execution data and rework results to evaluate test effectiveness and decide for additional testing or regression testing. (P)
5	Monitors test progress by evaluating defect arrival rate, failure intensity, and so forth. (P)
Technical Leader	
1	Conducts root cause analysis. (L/P)
2	Analyzes test data to decide on further testing activities. (L)
3	Uses the data to evaluate test effectiveness. (L)
4	Evaluates test results to identify appropriate process improvement opportunities. (P)
5	Monitors overall test progress by evaluating defect arrival rate, failure intensity, and so forth. (L)
Senior Software Engineer	
1	Creates new root cause analysis techniques. (P)
SOFTWARE SUSTAINMENT SKILLS	
Software Transition	
Technician	
1	Follows instructions to help develop software transition and operational documentation. (F)
2	Follows instructions to help install software. (F)
3	Assists in training operational support personnel. (A)
Entry Level	
1	Assists in identifying applicable systems and software operational standards. (A)
2	Assists in the development of software transition and operational documentation. (A) Installs software. (P)
3	Assists in the development of training material for operational support personnel. (A)
4	Trains software operational support personnel. (P)
5	Performs software system activation and check-out procedures. (P)
6	Assists in determining the impacts of software changes on the operational environment. (A)
7	Assists in system acceptance. (A)

Practitioner	
1	Participates in developing transition plans. (P)
2	Participates in the identification of stakeholders for transition and operational requirements. (P)
3	Participates in the identification of system and software constraints. (P)
4	Identifies applicable systems and software operational standards. (P)
5	Leads in the development of software transition and operational documentation. (L)
6	Leads in the installation of software. (L)
7	Develops training material for operational support personnel. (P)
8	Leads software operational training. (L)
9	Develops software system activation and check-out procedures. (P)
10	Participates in determining the impacts of software changes on the operational environment. (P)
11	Participates in system acceptance. (P)
Technical Leader	
1	Leads development of transition plans. (L)
2	Leads in the identification of stakeholders for transition and operational requirements. (L)
3	Leads in the identification of system and software constraints. (L)
4	Modifies existing and develops new systems and software operational standards. (L)
5	Approves software transition and operational documentation. (L)
6	Approves new software installations. (L)
7	Approves training material for operational support personnel. (L)
8	Approves software system activation and check-out procedures. (L)
9	Leads in determining the impacts of software changes on the operational environment. (L)
10	Leads system acceptance. (L)
Senior Software Engineer	
1	Modifies existing and creates new guidelines for transition plans. (C)
2	Modifies existing and creates new guidelines for the identification of stakeholders. (C)
3	Modifies existing and creates new templates for software transition and operational documentation. (C)
4	Modifies existing and creates new guidelines for determining the impacts of software changes on the operational environment. (C)
5	Modifies existing and develops new system acceptance methods, tools, and techniques. (C)
Software Support	
Technician	
1	Operates operational software configuration management tools. (A)

2	Follows instructions to perform operational software assurance tasks. (F)
3	Installs COTS and other software updates. (P)
4	Diagnoses and responds to reported software defects, anomalies, and operational incidents and events. (P)
5	Operates tools to collect operational data under supervision. (F)
Entry Level	
1	Performs operational software configuration management. (P)
2	Performs operational software assurance. (P)
3	Updates COTS and other software technologies to maintain currency. (P)
4	Leads software help desk activities. (L)
5	Analyzes operational data. (P)
6	Assists in implementing software retirement procedures. (A)
Practitioner	
1	Develops operational software configuration management plans. (L)
2	Leads operational software assurance activities. (L)
3	Leads maintenance of COTS and other software technologies to maintain currency. (L)
4	Develops software help desk plans. (L)
5	Acquires tools and supervises analysis of operational data. (L)
6	Implements software retirement procedures. (P)
Technical Leader	
1	Approves operational software configuration management plans. (L)
2	Develops software assurance plans. (L)
3	Creates policies that cover help desk operations. (C)
4	Develops plans for collecting and processing operational data. (L)
5	Develops software retirement plans. (L)
Senior Software Engineer	
1	Modifies existing and creates new standards and frameworks for operational software configuration management. (C)
2	Modifies existing and creates new methods, tools, and techniques for collecting and processing operational data. (C)
Software Maintenance	
Technician	
1	Participates in obtaining baseline software artifacts. (P)
2	Assists in performing preventative maintenance and software re-engineering activities. (A)
Entry Level	
1	Assists in implementing software maintenance processes and plans. (A)
2	Obtains and maintains software baseline artifacts. (P)
3	Performs problem identification. (P)
4	Assists in making changes to software (corrective, adaptive, perfective). (A)
5	Performs preventative maintenance and software re-engineering activities. (P)

6	Assists in monitoring and analyzing software maintenance activities. (A)
Practitioner	
1	Implements software maintenance processes and plans. (P)
2	Identifies software baseline artifacts. (L)
3	Performs change impact analysis. (P)
4	Implements plans and makes changes to software (corrective, adaptive, perfective). (P)
5	Leads preventative maintenance and software re-engineering activities. (L)
6	Monitors and analyzes software maintenance activities. (P)
Technical Leader	
1	Leads development of software maintenance processes and plans. (L)
2	Leads problem identification and technical impact analysis. (P)
3	Leads development of plans and supervises making changes to software (corrective, adaptive, perfective). (L)
4	Plans for and supervises preventative maintenance and software re-engineering activities. (L)
5	Leads monitoring and analysis of software maintenance activities. (L)
Senior Software Engineer	
1	Modifies existing and creates new software maintenance policies, processes, and procedures. (C)
2	Modifies existing and creates new policies and procedures for making changes to software (corrective, adaptive, perfective). (C)
3	Modifies existing and creates new policies and procedures for preventative maintenance and software re-engineering. (C)
SOFTWARE PROCESS AND LIFE CYCLE SKILLS	
Software Development Life Cycle Implementation	
Technician	
1	Provides assistance in the installation and use of tools appropriate for a project’s designated life cycle model. (F/A)
Entry Level	
1	Carries out process activities specified in a life cycle process model script. (A/P)
Practitioner	
1	Leads a small team in execution of some portion of a life cycle process model (such as software design). (P/L)
Technical Leader	
1	Selects a life cycle model process for a software team. (L)
2	Assists in selection of a department or organizational-wide life cycle process model. (P/L)
Senior Software Engineer	
1	Determines the need for and selects or develops an organization-wide life cycle model. (C)
2	Selects department- or organization-wide process models. (C)

3	Advises on process infrastructure, training, and tools. (C)
Process Definition and Tailoring	
Technician	
1	Provides assistance in the installation and use of tools for defining and modifying software processes. (F/A)
Entry Level	
1	Interprets and adapts a software process to individual process responsibilities and tasks. (A/P)
Practitioner	
1	Provides review of defined and tailored processes. (A/P)
2	Tailors a software process to the work of a small team. (P/L)
Technical Leader	
1	Leads definition and tailoring of software processes for a project team or for a software engineering activity (such as requirements engineering, design, and so forth). (L/C)
2	Provides guidance to others involved in tailored processes (individual and team). (L)
Senior Software Engineer	
1	Conducts research into effective software process definition and tailoring. (C)
2	Leads definition and tailoring of organization-wide software processes. (L/C)
Process Implementation and Management	
Technician	
1	Provides assistance in the installation and use of tools for implementing, managing, and measuring software processes. (F/A)
Entry Level	
1	Collaborates in the execution of a team software process.
2	Implements and manages individual processes. (F/A)
Practitioner	
1	Leads small teams in the implementation and execution of a software process. (P/L)
2	Provides guidance and advice to individuals on the implementation and management of their personal processes. (P/L)
3	Serves as a member of a software engineering process group (SEPG). (P)
Technical Leader	
1	Leads large teams or multi-team projects in the implementation and execution of a software process. (L)
2	Provides guidance and advice to software teams on how to implement and manage software processes. (L/C)
3	Serves as leader of an SEPG. (L)
Senior Software Engineer	
1	Provides organization-wide guidance and advice on how to implement and manage software processes. (C)

2	Provides guidance and advice on the formation, structure, and responsibilities of SEPGs. (C)
Process Assessment and Improvement	
Technician	
1	Provides assistance in the installation and use of tools for assessing and improving software processes. (F/A)
Entry Level	
1	Assists in collecting data for assessment of a software process. (A)
2	Collects data relevant to individual process execution. (F/A)
3	Assesses and implements improvement of an individual software process. (A/P)
Practitioner	
1	Leads small teams in collecting data for assessment of software processes. (P/L)
2	Analyzes process assessment data and implements improvement of small team software processes. (P/L)
3	As a member of an SEPG, provides input on software process improvement. (A/P)
Technical Leader	
1	Leads software teams in collecting data for assessment of software processes. (P/L)
2	Analyzes process assessment data and implements improvement of team software processes. (P/L)
3	Leads the SEPG in providing guidance on department- or organization-wide software process improvement. (L/C)
Senior Software Engineer	
1	Conducts research into the effectiveness and improvement of software processes. (C)
2	Uses assessment data, team reports, and SEPG reports to establish organization procedures and standards for software process improvement. (C)
SOFTWARE SYSTEMS ENGINEERING SKILLS	
System Development Life Cycle Modeling	
Technician	
1	Uses tools and follows directions to prepared depictions and documentation of tailored development models. (F)
Entry Level	
1	Assists in integrating the selected software development model into the systems development model. (A)
2	Participates in systems engineering meetings and prepares meeting minutes to include decisions made, open issues, other relevant discussions, and action items. (A)
Practitioner	
1	Participates in integrating the selected software development model into the system development model. (P)
Technical Leader	

1	Participates in selection of the system development life cycle model. (P)
2	Leads selection of the software development life cycle model. (L)
3	Leads integration of the software development model into the system development model. (L)
4	Participates in/leads tailoring of policies, procedures, and templates, and selection of applicable standards for projects and programs. (P/L)
Senior Software Engineer	
1	Prepares frameworks for integrating an organization's system and software development models. (C)
2	Modifies existing models and creates new models and ways of integrated software engineering and system engineering development models. (C)
3	Determines applicable policies, procedures, standards, and guidelines and tailors them for an organization's system and software development models. (C)
Concept Definition	
Technician	
1	Assists by locating identified stakeholders. (A)
2	Arranges stakeholder meetings. (A)
3	Attends stakeholder meetings and takes meeting minutes. (A)
4	Follows directions to prepare elements of the Concept of Operations. (F)
Entry Level	
1	Identifies potential stakeholders by examining historical data and having discussions with appropriate personnel inside and outside the systems engineering team. (A/P)
2	Attends stakeholder meetings to document stakeholder needs, wants, and desires. (P)
3	Develops elements of the Concept of Operations. (A)
Practitioner	
1	Develops lists of stakeholders and categorizes their likely interests. (P/L)
2	Attends stakeholder meetings and solicits stakeholder needs, wants, and desires. (P)
3	Leads development of scenarios, storyboards, prototypes, and use cases. (L)
Technical Leader	
1	Prepares criteria for identifying stakeholders. (L)
2	Leads stakeholder meetings. (L)
3	Facilitates development of the Concept of Operations. (L)
4	Obtains stakeholder consensus on the Concept of Operations. (L)
5	Develops acceptance criteria. (P)
Senior Software Engineer	
1	Develops new techniques for identifying stakeholders. (C)
2	Develops new methods, tools, and techniques for concept definition. (C)
System Requirements Engineering	
Technician	

1	Attends meetings and documents the system development environment and technology constraints. (P)
2	Procures and operates traceability tools to establish and maintain traceability. (P)
3	Follows instructions to document the system requirement specification. (F)
4	Documents plans, procedures, and scenarios for system integration, verification, validation, and deployment. (A)
Entry Level	
1	Provides training on traceability procedures and tools. (A)
2	Assists in development of the system requirements specification. (A)
3	Assists in development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (A)
Practitioner	
1	Establishes the system development environment and identifies technology constraints. (P)
2	Identifies system-level traceability requirements and tools. (P)
3	Participates in development of the system requirements specification. (P)
4	Participates in development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (P)
Technical Leader	
1	Establishes the system development environment and identifies technology constraints. (L)
2	Identifies system-level traceability requirements and tools. (L)
3	Leads development of the system requirements specification. (L)
4	Leads the development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (L)
Senior Software Engineer	
1	Establishes organizational policies and procedures for system requirements engineering. (P)
2	Modifies existing and develops new methods, tools, and techniques for system requirements engineering. (C)
System Design	
Technician	
1	Identifies sources of software components to be procured. (A)
Entry Level	
1	Assists in developing alternative solution concepts and conducting trade studies to identify major system components. (A)
2	Participates in making buy/build decisions for software components. (P)
3	Assists in selecting components to be procured. (A)
Practitioner	
1	Participates in developing alternative solution concepts and conducting trade studies to identify major system components. (P)
2	Participates in identifying system components as well as the interfaces and relationships among components. (P)
3	Recommends buy/build decisions for software components. (P)

4	Procures selected software components. (L)
5	Participates in system design meetings to avoid isolated stovepipe units of software. (P)
Technical Leader	
1	Leads development of alternative solution concepts to identify major system components. (L)
2	Leads/participates in making buy/build decisions for major system components. (L/P)
3	Approves buy/build decisions for software. (L)
4	Leads/participates in system design meetings to avoid isolated stovepipe units of software. (P/L)
Senior Software Engineer	
1	Develops policies and procedures for system design in an organization. (C)
2	Develops new approaches to system design to avoid isolated stovepipe units of software. (C)
Requirements Allocation	
Technician	
1	Documents allocable and non-allocable requirements. (F)
2	Documents allocation of requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (F)
3	Operates traceability tools and generates traceability reports. (F)
4	Assists in clarifying and refining requirements allocated to software. (A)
Entry Level	
1	Assists in identifying allocable and non-allocable requirements. (A)
2	Attends meetings and records minutes to allocate requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (P/L)
3	Develops bi-directional traceability between system requirements and software requirements. (P)
4	Participates in clarifying and refining requirements allocated to software. (P)
Practitioner	
1	Identifies allocable and non-allocable requirements. (P)
2	Participates in meetings to allocate requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (P)
3	Participates in meetings to refine requirements allocated to software. (P)
4	Leads traceability from system requirements to software requirements. (L)
5	Leads in clarifying and refining requirements allocated to software. (L)
Technical Leader	
1	Leads/participates in meetings to identify and allocate requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (L/P)
2	Leads meetings to refine requirements allocated to software. (L)

Senior Software Engineer	
1	Develops new methods, tools, and techniques for requirements allocation and flowdown. (C)
Component Engineering	
Technician	
1	Documents buy/build decisions for software components. (A)
2	Maintains baselines of software components. (A)
Entry Level	
1	Assists in determining and documenting needed kinds of software components. (A)
2	Assists in determining buy/build decisions for software components. (L)
3	Develops and integrates software components. (A/P)
Practitioner	
1	Determines needed software components. (P)
2	Determines buy/build decisions for software components. (P)
3	Develops and integrates software components. (P/L)
Technical Leader	
1	Leads in determining needed kinds of software components for a project or program. (L)
2	Leads the buy/build decision-making process for software components. (L)
3	Establishes procedures to develop and integrate software components. (L)
4	Provides liaison from software engineering to systems engineering and other major component engineering. (L)
Senior Software Engineer	
1	Modifies existing and develops new methods, tools, and techniques for component engineering. (C)
System Integration and Verification	
Technician	
1	Assists in integration of software with other system components. (A)
Entry Level	
1	Assists in system verification activities. (A)
2	Assists in providing liaison to software component engineers. (A)
Practitioner	
1	Participates in integration of software with other system components. (P)
2	Participates in system verification activities. (P)
3	Provides liaison to software component engineers. (P)
Technical Leader	
1	Leads integration of software with other system components. (L)
2	Leads/participates in system verification activities. (L/P)
3	Leads/participates in providing liaison to software component engineers. (L)
Senior Software Engineer	

1	Modifies existing and provides new methods of integrating software with other system components. (C)
System Validation and Deployment	
Technician	
1	Operates tools for performing simulated and live system tests. (F)
2	Operates tools for performing system acceptance testing. (F)
Entry Level	
1	Assists in performing simulated and live system tests. (A)
2	Assists in system acceptance testing. (A)
Practitioner	
1	Participates in simulated and live system tests. (P)
2	Participates in system acceptance testing. (P)
3	Serves as liaison to software component engineers during system validation and deployment. (P)
Technical Leader	
1	Leads/participates in simulated and live system tests. (P/L)
2	Establishes system acceptance criteria. (L)
3	Leads/participates in system acceptance testing. (P/L)
4	Leads in providing liaison to software component engineers during system validation and deployment. (P/L)
Senior Software Engineer	
1	Modifies existing and develops new methods, tools, and techniques for system validation and deployment. (C)
System Sustainment Planning	
Technician	
Entry Level	
1	Assists in planning for system sustainment. (A)
Practitioner	
1	Participates in identifying stakeholders and developing a transition plan and requirements for operational support. (P)
2	Prepares for operational support. (P)
Technical Leader	
1	Establishes criteria and procedures for system sustainment
2	Leads/participates in planning for system sustainment. (L)
Senior Software Engineer	
SOFTWARE QUALITY SKILLS	
Software Quality Management (SQM)	
Technician	
1	Follows defined quality processes and standards. (P)
2	Assists with establishing the appropriate infrastructure (such as defect tracking tools) to support organizations' quality goals. (A)

Entry Level	
1	Follows quality standards for the product and supporting processes. (P)
2	Follows defined quality models. (P)
3	Uses appropriate tools and resources to develop quality products. (P)
4	Assists with identification of the different quality characteristics and attributes for the product. (A/P)
5	Ensures that product-quality goals are achieved. (P)
6	Collects quality metrics and prepares quality documentation to be shared with appropriate stakeholders. (P)
7	Develops and updates an appropriate traceability matrix for the product. (P)
Practitioner	
1	Establishes quality standards for the product. (P/L)
2	Selects appropriate SQM processes that support the identified quality goals for the project. (P)
3	Identifies quality characteristics and attributes for the product and establishes priorities. (P)
4	Identifies the quality models that need to be followed for the project. (P)
5	Develops the Quality Assurance (QA) plan for the project. (P)
6	Identifies appropriate stakeholders who have authority and accountability regarding the quality process and quality product. (P)
7	Identifies appropriate measures that support achieving product-quality goals. (P/L)
8	Identifies appropriate tools and resources that need to be used in order to achieve product-quality goals. (P/L)
9	Verifies that quality goals and requirements are met. (P/L)
10	Identifies continuous improvement opportunities across the project. (L)
Technical Leader	
1	Establishes a culture of producing quality products and of following quality processes across projects. (P/L)
2	Establishes quality standards, models, and processes for projects. (P/L)
3	Analyzes the advantages and disadvantages of alternative SQM processes and tools that can be used for achieving organizational goals for product quality. (P)
4	Develops the QA plan for the project. (L)
5	Identifies organizational measures that support achieving product-quality goals (across projects). (P/L)
6	Identifies continuous improvement opportunities across projects. (L)
Senior Software Engineer	
1	Creates new and improved quality practices for delivering high-quality products. (C)
2	Creates new processes. (C)
3	Examines and assesses the effectiveness of a specific SQM process across an organization. (C)
4	Makes recommendations related to organization-wide SQM processes. (C)

5	Creates/modifies SQM processes to achieve higher-quality products and processes. (C)
6	Proposes/Designs new tools to improve the achievement of product-quality goals. (L)
Reviews (review, walkthrough, inspection)	
Technician	
1	Assists with necessary logistics associated with reviews and inspections, including but not limited to: a. meeting logistics, (P) b. performing all appropriate data warehousing, (P) and c. generating appropriate reports associated with the meeting. (P)
Entry Level	
1	Participates as an active member of the review team in order to achieve the goals of the activity. (P)
2	Uses appropriate checklists called for by the review organizer. (P)
3	Collects appropriate and accurate data that is called for by the review organizer. (P)
4	Produces appropriate documentation called for by the quality management plan. (P)
5	Follows appropriate practices defined by the quality management plan. (P)
Practitioner	
1	Identifies appropriate review processes needed to achieve product-quality goals. (P/L)
2	Identifies appropriate personnel that need to participate in review activities. (P)
3	Identifies appropriate measures that need to be collected as part of the product review. (P)
4	Identifies appropriate artifacts under the review and corresponding checklist. (P)
5	Analyzes collected product data for the purpose of root cause analysis and assessment of review effectiveness. (P)
6	Identifies appropriate corrective actions in order to achieve product improvement. (P)
7	Leads the review team. (P)
Technical Leader	
1	Identifies appropriate organization-wide review processes. (P/L)
2	Conducts across-the-organization data analysis for the purpose of root cause analysis. (P)
3	Based on the review data, identifies appropriate corrective actions to be implemented across projects for the purpose of achieving product improvement. (P/L)
Senior Software Engineer	
1	Creates new or customizes review processes to meet organizational needs. (C)
2	Develops new root cause analysis techniques. (C)
Audits (concentrate on both product and process, but are done by an independent, internal or external, organization)	
Technician	

1	Establishes the environment necessary to conduct the audit. (A/P)
Entry Level	
1	Participates in audits. (P)
Practitioner	
1	Plans, organizes, and conducts audits. (P/L)
2	Classifies issues identified by audits. (P)
3	Establishes and implements appropriate resolution strategies for identified issues. (P)
Technical Leader	
1	Establishes audit infrastructure by identifying: a. appropriate organization to conduct the audit, (P) b. products and processes that need to be included in audits, (P) and c. stakeholders receiving the audit results. (P)
2	Analyzes audit results for continuous improvement. (P)
Senior Software Engineer	
1	Creates new audit processes. (C)
Statistical Control	
Technician	
1	Establishes the environment necessary for data collection and warehousing. (P)
Entry Level	
1	Collects a set of quality data under statistical control. (P)
Practitioner	
1	Analyzes the collected data. (L)
2	Deploys a set of control processes. (P)
3	Evaluates the effectiveness of the control processes. (A)
Technical Leader	
1	Identifies a set of quality data under statistical control. (P)
2	Identifies a set of subjective and objective variances for the data. (P)
3	Analyzes the collected data. (L)
4	Establishes and implements a set of control processes. (P)
5	Evaluates the effectiveness of the control processes. (P)
Senior Software Engineer	
1	Creates or modifies organizational statistical quality control gates. (C)
SOFTWARE SECURITY SKILLS	
Requirements	
Technician	
Entry Level	
Practitioner	
1	Identifies security risks (such as misuse cases). (P)
2	Creates requirements that capture security issues. (P)
3	Performs initial threat modeling. (P)
Technical Leader	

Senior Software Engineer	
1	Creates or proposes new methods for recognizing security vulnerabilities. (C)
Design	
Technician	
Entry Level	
1	Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data). (F)
2	Uses appropriate, secure design patterns. (F)
Practitioner	
1	Models threats and associated risks of new and modified systems. (P)
2	Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems. (P)
Technical Leader	
Senior Software Engineer	
Construction	
Technician	
Entry Level	
1	Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). (F)
2	Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems). (F)
Practitioner	
1	Selects or establishes project coding standards to avoid security vulnerabilities. (P)
2	Reviews and approves coding standards to avoid security vulnerabilities. (P)
Technical Leader	
1	Establishes organization coding standards to avoid security vulnerabilities. (L)
Senior Software Engineer	
1	Creates new coding standards to avoid security vulnerabilities. (C)
Process	
Technician	
1	Assists in the collection of metrics for security assessment processes. (A)
Entry Level	
1	Follows project standards in the collection of security assessment metrics. (F)
Practitioner	
Technical Leader	
1	Establishes organization standards for security assessment processes. (L)
Senior Software Engineer	
Quality	
Technician	

1	Assists in the installation of static analysis tools. (A)
Entry Level	
1	Performs code reviews to identify security vulnerabilities. (P)
2	Uses static analysis methods to identify security vulnerabilities. (P)
Practitioner	
1	Selects appropriate static analysis tools to identify security vulnerabilities. (P/L)
Technical Leader	
Senior Software Engineer	
1	Creates new static analysis methods or tools. (C)
SOFTWARE SAFETY SKILLS	
Requirements	
Technician	
1	Assists in collecting data for the creation of a hazard list. (F/A)
2	Assists in the identification of top-level mishaps and their causes. (F/A)
3	Assists with the installation of safety and reliability tools. (F/A)
Entry Level	
1	Creates and verifies preliminary hazard lists. (A/P)
2	Uses software tools to build safety models (FTA, ETA, FMEA). (A/P)
3	Assists in safety requirements identification. (F/A)
Practitioner	
1	Using tools, conducts formal system hazard analyses verifying safety models. (P)
2	Identifies safety requirements. (P)
3	Assures that safety requirements are included in the overall system requirements. (P)
Technical Leader	
1	Verifies completeness and correctness of safety requirements. (/L)
2	Interacts with system and software engineers to assure that safety requirements are complete and realizable. (L)
Senior Software Engineer	
1	Creates or proposes new methods for safety assessment, mitigation, and verification. (C)
Design	
Technician	
1	Assists in identifying mitigation techniques for defined safety requirements. (F/A)
Entry Level	
1	Implements design solutions to assure that the hazards are mitigated and the safety requirements are met. (A)
2	Follows the recommended design principles. (P)
3	Leads the project in deciding the proposed architectural solutions to mitigate hazards. (L)

Practitioner	
1	Proposes and selects design solutions to assure the hazards are mitigated. (P)
2	Supervises the design team. Analyzes risk and verifies design from a safety perspective. (P/L)
Technical Leader	
1	Verifies completeness and correctness of the design, including safety hazards and safety qualities. (P/L)
2	Leads the project in deciding the proposed architectural solutions to mitigate hazards. (L)
3	Evaluates risk related to design for safety. (P/L)
Senior Software Engineer	
1	Creates or proposes new design solutions, leading to the increased safety of new designs. (C)
Construction	
Technician	
Entry Level	
1	Implements large code components and their interfaces, considering safe coding practices to avoid safety violations. (A/P)
Practitioner	
1	Implements the architecture and design to ensure code safety. (P / L)
2	Manages the interfacing of large code components with special attention to potential safety issues. (P/L)
Technical Leader	
1	Establishes organization standards to ensure code safety. (L/C)
2	Oversees and verifies that the safety aspects of the design are implemented in the produced code. (L)
Senior Software Engineer	
1	Creates new standards to ensure code safety. (C)
Testing	
Technician	
1	Assists in the installation of tools and infrastructure for safety requirements testing. (F/A)
Entry Level	
1	Performs testing using tools with a focus on safety requirements. (A/P)
Practitioner	
1	Selects appropriate testing techniques to assure the safety of the application. (P)
2	Applies applicable industry standards to assure that the product meets industry safety criteria. (P)
Technical Leader	
1	Establishes organization standards for safety validation and verification. (L/C)
2	Manages the project, being responsible for overall safety and meeting industry guidelines. (L)
Senior Software Engineer	

1	Contributes expertise to establish new organization guidelines related to testing the safety of software-intensive applications. (C)
Process	
Technician	
1	Assists in the collection of data to establish the project safety case. (F/A)
Entry Level	
1	Identifies artifacts required to establish the safety case, following industry standards. (A/P)
Practitioner	
1	Contributes to and verifies the completeness of the safety case, following selected industry criteria. (P)
Technical Leader	
1	Leads the safety team responsible for the project safety case. (L)
2	Establishes organization standards for safety assessment processes and selection of safety criteria. (L/C)
Senior Software Engineer	
1	Contributes expertise to establish better means of assessing safety. (C)
Quality	
Technician	
1	Assists in the collection of safety QM data. (A)
Entry Level	
1	Collects safety QM data and reports the project status. (A/P)
Practitioner	
1	Supervises collection of QM data and their compatibility with the safety case. (P/L)
Technical Leader	
1	Manages the overall project quality with a focus on safety aspects. (L)
Senior Software Engineer	
1	Contributes expertise to improve means of measuring and establishing the safety quality of the product and process. (C)
SOFTWARE CONFIGURATION MANAGEMENT SKILLS	
Plan SCM	
Technician	
1	Operates SCM tools. (F)
2	Operates and maintains the SCM library under technical leader supervision. (F)
Entry Level	
1	Assists in determining impact of constraints on SCM imposed by policies, contract, and SDLC. (A)
2	Assists in developing, updating, and maintaining the SCMP. (A)
3	Provides measurement data for SCM measures. (P)
4	Assists in identifying software configuration items (SCIs). (A)

5	Assists in selecting and procuring SCM tools. (A)
6	Sets up an SCM library for a project under technical leader supervision. (L)
Practitioner	
1	Participates in determining impact of constraints on SCM imposed by policies, contract, and SDLC. (P)
2	Develops and maintains the SCMP. (L)
3	Assists in specifying the SCM measures to be used. (A)
4	Participates in identifying SCIs and the relationships among them. (P)
5	Procures SCM tools. (P)
Technical Leader	
1	Determines constraints and impacts of constraints on SCM imposed by policies, contracts, and SDLC. (L)
2	Adopts an existing way of organizing for SCM and tailors a template for the SCMP. (L)
3	Specifies the SCM measures to be used. (L)
4	Identifies SCIs and the relationships among them. (L)
5	Specifies SCM tools. (L)
6	Specifies the template for, and supervises setting up, the SCM library. (L)
Senior Software Engineer	
1	Develops new ways of organizing to perform SCM. (C)
2	Develops new templates and ways of planning for SCM. (C)
3	Develops new measures and measurements for SCM. (C)
4	Develops procedures for identifying SCIs and the relationships among them. (C)
5	Specifies new SCM tools and ways of combining existing SCM tools. (C)
6	Specifies new ways of organizing SCM libraries. (C)
Conduct SCM	
Technician	
1	Operates tools to generate SCM status and audit reports. (F)
2	Generates, classifies, and manages problem reports. (F/A)
Entry Level	
1	Implements and documents approved changes to SCIs.
2	Generates, classifies, and manages problem reports. (P)
3	Assists in using adopted mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (A)
4	Assists in establishing and maintaining the mechanisms for recording and reporting SCM information and generating SCM audit reports. (A) Provides SCM audit reports as scheduled and requested. (P)
Practitioner	
1	Evaluates and reports to CCB the impacts of proposed changes to SCIs. (P)
2	Generates, classifies, and manages problem reports. (L)
3	Uses established procedures for populating and maintaining the SCM library. (P)
4	Uses established mechanisms to record and report SCM information. (P)

5	Develops and tailors tools for generating SCM audit reports. (P)
Technical Leader	
1	Tailors and adopts mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (P)
2	Appoints members and convenes the CCB. (L)
3	Leads CCB in making yes/no decisions on change requests. (L)
4	Ensures that approved changes are made and documented. (L)
5	Maintains mechanisms for recording and reporting SCM information. (L)
6	Establishes and maintains mechanisms for generating SCM audit reports. (L)
Senior Software Engineer	
1	Revises existing and develops new mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (C)
2	Develops new mechanisms for SCM status accounting. (C)
3	Develops new processes and procedures for generating SCM audit reports. (C)
Manage Software Releases	
Technician	
1	Operates tools to build software releases. (A/P)
2	Uses software release tools to produce software releases. (A)
3	Modifies existing and creates new tools for building software releases. (C)
Entry Level	
1	Participates in developing software release plans. (A/P)
2	Participates in the building and verifying of software releases. (P)
3	Participates in the building of software releases. (P)
Practitioner	
1	Participates in developing software release plans. (P)
2	Leads the building and verifying of software releases. (L)
3	Implements release plans. (P)
Technical Leader	
1	Develops software release plans. (L)
Senior Software Engineer	
1	Modifies existing and develops new formats and procedures for implementing software release plans. (C)
2	Modifies existing and creates new tools for building software releases. (C)
SOFTWARE MEASUREMENT SKILLS	
Plan Measurement Process	
Technician	
1	Procures and installs measurement tools. (L)
Entry Level	
1	Assists in planning for data storage. (A)
2	Assists in selecting data collection and analysis methods. (A)
3	Assists in identifying measurement tools and manual procedures. (A)

Practitioner	
1	Participates in identifying measurement needs for a project or program. (P)
2	Assists in selecting measures and measurement scales. (A)
3	Establishes data collection and analysis methods. (P)
4	Participates in setting target values and thresholds. (P)
5	Participates in establishing report formats and reporting procedures. (P)
6	Identifies measurement tools and manual procedures. (P)
7	Plans for data storage. (P)
8	Selects data collection and analysis methods. (P)
9	Selects measurement tools and manual procedures. (A)
Technical Leader	
1	Participates in identifying measures for an organization. (P)
2	Leads identification of measurement needs for a project or program. (L)
3	Selects measures and measurement scales. (L)
4	Reviews and approves data collection and analysis methods. (L)
5	Sets target values and thresholds. (L)
6	Establishes report formats and reporting procedures. (L)
7	Reviews and approves measurement tools and manual procedures. (L)
8	Reviews and approves plan for data storage. (L)
9	Approves data collection and analysis methods. (L)
10	Approves measurement tools and manual procedures. (A)
Senior Software Engineer	
1	Defines attributes of the process and product measures to be used throughout an organization. (C)
2	Develops new ways to identify measurement needs. (C)
3	Defines new measures and measurement scales. (C)
4	Develops new data collection and analysis methods. (C)
5	Develops new report formats and reporting procedures. (C)
6	Develops new measurement tools and manual procedures. (C)
Perform Measurement Process	
Technician	
1	Uses measurement tools to collect data. (P)
2	Maintains valid data in a repository. (P)
3	Generates and distributes reports (P)
Entry Level	
1	Uses manual procedures to collect data. (P)
2	Assists in validating collected data. (A)
3	Identifies and recommends improvements to the measurement process. (A)
Practitioner	
1	Validates collected data. (P)
Technical Leader	

1	Leads and coordinates the measurement process. (L)
2	Approves tactical improvements to the measurement process. (L)
Senior Software Engineer	
1	Periodically reviews methods, tools, and techniques used to perform the measurement process. (C)
2	Modifies existing and develops new review methods, tools, and techniques used to perform the measurement process. (C)
HUMAN-COMPUTER INTERACTION SKILLS	
Requirements	
Technician	
1	Assists in identifying target users. (A)
2	Follows directions to create simple prototypes for use in eliciting user interface requirements. (F)
Entry Level	
1	Identifies stakeholders to provide HCI requirements. (P)
2	Identifies target users and their attributes. (P)
3	Assists in identifying user interface requirements. (A)
4	Designs and creates prototypes for use in eliciting user interface requirements. (P)
Practitioner	
1	Reviews identification of stakeholders to provide HCI requirements. (P)
2	Assists in selecting a process model for HCI interface development. (P)
3	Recom-mends HCI tools for project use. (A)
4	Reviews and refines target user identification and describes their relevant attributes. (P)
5	Leads identification of user interface requirements. (L)
6	Reviews and refines prototypes, tests for elicitation, and refines user interface requirements. (P)
7	Uses prototypes to elicit and refine user interface requirements. (P)
8	Identifies technical interface requirements (between the user interface and system components). (P)
9	Designs the details of the technical interface requirements (between the user interface and system components). (P)
Technical Leader	
1	Coordinates work activities for stakeholder identification. (L)
2	Determines which process model approach will be used by the HCI team to develop the interface. (L)
3	Selects HCI tools for project use. (P)
4	Leads review and refinement of user interface requirements. (L)
5	Identifies constraints on user interface implementation. (L)
6	Defines the technical interface requirements (between the user interface and system components). (L)
7	Selects applicable standards. (L)

Senior Software Engineer	
1	Modifies existing and creates new methods and tools for stakeholder identification. (C)
2	Modifies existing and creates new methods and tools for target user identification. (C)
3	Modifies existing and creates new methods and tools for prototyping. (C)
4	Modifies existing and creates new methods and tools for specifying technical interface requirements. (C)
Interaction Style Design	
Technician	
1	Follows instruction to assist in documenting use cases, scenarios, interaction dialogs, and storyboards. (F)
2	Assists in identifying user input errors. (A)
3	Documents two-way traceability to requirements and to test cases and test scenarios. (P)
4	Follows directions to develop or refine interface prototypes. (F)
Entry Level	
1	Documents use cases, scenarios, interaction dialogs, and storyboards. (P)
2	Assists in identifying user input errors and error handling approaches. (A)
3	Assists in identifying interaction modes. (A)
4	Establishes two-way traceability between use cases, scenarios, interaction dialogs, and storyboards and specific user interface requirements and acceptance criteria. (P)
5	Develops interface prototypes. (P)
Practitioner	
1	Reviews and refines use cases, scenarios, interaction dialogs, and storyboards. (L)
2	Identifies user input errors and error handling approaches. (P)
3	Applies metaphors and conceptual models to define interaction style. (P)
4	Identifies and refines interaction modes. (P)
5	Reviews and refines interface prototypes. (P)
Technical Leader	
1	Leads and coordinates interaction-style design activities. (L)
2	Selects and refines user error handling approaches. (P)
3	Selects metaphors and conceptual models. (L)
4	Works with the system design team to establish component interfaces between the user interface and system components. (L)
Senior Software Engineer	
1	Modifies existing and creates new methods and tools for interaction-style design. (C)
2	Modifies existing and creates new methods and tools for interaction-style design. (C)
Visual Design	
Technician	

1	Follows instructions to assist in the creation of mock-ups and sketches. (F)
Entry Level	
1	Assists in designing page/screen layout. (A)
2	Assists in selecting from existing icons and designing new icons. (A)
3	Assists in selecting color theme, font styles, and font sizes. (A)
4	Assists in menu design. (A)
5	Creates mock-ups and sketches. (P)
Practitioner	
1	Designs page/screen layout. (P)
2	Selects from existing icons and designs new icons. (P)
3	Selects color theme, font styles, and font sizes. (P)
4	Designs menus. (P)
5	Reviews selections for color theme, font styles, and font sizes, and checks selection against applicable standards. (P)
6	Reviews and revises mock-ups and sketches with stakeholders. (P)
Technical Leader	
1	Revises/approves final page/screen layouts. (L)
2	Revises/approves icons and identifies new icons as needed. (L)
3	Revises/approves color theme, font styles, and font sizes. (L)
4	Reviews and refines menu designs. (L)
5	Approves visual design components and reviews design with stakeholders and/or target users. (L)
Senior Software Engineer	
1	Modifies existing and creates new methods and tools for visual design. (C)
Usability Testing and Evaluation	
Technician	
1	Writes user tests that evaluate user behavior. (A)
2	Assists in conducting usability tests and collecting data. (A)
3	Follows instructions to assist in analyzing results of usability testing. (F)
Entry Level	
1	Analyzes design with a usability checklist. (P)
2	Assists in identifying representative test subjects from the target user group. (A)
3	Assists in obtaining test subjects. (A)
4	Assists in designing usability tests. (A)
5	Conducts usability tests and collects data. (P)
6	Analyzes results of usability testing. (P)
Practitioner	
1	Selects and tailors one or more usability checklists. (P)
2	Identifies representative test subjects from the target user group. (F)
3	Reviews results of checklist analysis and recommends design changes. (P/F)
4	Obtains test subjects. (P)
5	Designs usability tests. (P)

6	Supervises usability testing. (P/L)
7	Makes recommendations based on analysis of usability testing results. (P)
Technical Leader	
1	Leads and coordinates usability testing and evaluation activities. (L)
2	Approves selection of one or more usability checklists. (L)
3	Approves selection of test subjects. (L)
4	Reviews, refines, and finalizes usability tests. (L)
5	Reviews and approves recommendations and results of usability testing. (L)
Senior Software Engineer	
1	Modifies existing and creates new methods and tools for usability testing. (C)
Accessibility	
Technician	
Entry Level	
1	Assists in identifying accessibility needs for user interfaces. (A)
2	Assists in identifying the needs for international accessibility (languages, cultural considerations, and so forth). (A)
3	Uses the selected tools and techniques for implementing required accessibility. (A)
Practitioner	
1	Identifies accessibility needs for user interfaces. (P)
2	Develops acceptance criteria and tests for accessibility aspects of the user interface. (P)
3	Identifies the needs for international accessibility (languages, cultural considerations, and so forth). (P)
4	Selects tools and techniques for providing required accessibility. (P)
Technical Leader	
1	Leads and coordinates accessibility activities. (L)
2	Determines which accessibility needs must be addressed in the user interface. (L)
3	Determines the extent to which the user interface must accommodate needs for international accessibility. (L)
Senior Software Engineer	
1	Develops new tools and techniques for providing accessible interface elements. (C)

The SWECOM Skill Areas, Sets and Activities by competency level [IEEE-CS14]

APPENDIX E

Security integrated SE2004 curriculum recommendation.

Integrated Curriculum Recommendation with Software Assurance	
Software Security Requirements Skills (SWECOM SKILL SET)	
Software Modeling & Analysis (SEEK AREA)	
Requirements fundamentals (SEEK UNIT)	
	Creates requirements that capture security issues. (SWECOM ACTIVITY)
Eliciting requirements (SEEK UNIT)	
	Identifies security risks (such as misuse cases, which are Use Cases that show possible misuse of the system). (SWECOM ACTIVITY)
Requirements specification & documentation (SEEK UNIT)	
	Creates requirements that capture security issues. (SWECOM ACTIVITY)
Requirements validation (SEEK UNIT)	
	Performs initial threat modeling. (SWECOM ACTIVITY)
Software Security Design Skills (SWECOM SKILL SET)	
Software Design (SEEK AREA)	
Design concepts (SEEK UNIT)	
	Uses appropriate, secure design patterns. (SWECOM ACTIVITY)

Design strategies (SEEK UNIT)	
Architectural design (SEEK UNIT)	
	Follows recommended design principles to create secure systems (such as providing multiple layers of protection, using access control mechanisms, and encrypting sensitive data). (SWECOM ACTIVITY)
Human computer interface design (SEEK UNIT)	
	Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems. (SWECOM ACTIVITY)
Detailed design (SEEK UNIT)	
	Models threats and associated risks of new and modified systems. (SWECOM ACTIVITY)
Design support tools and evaluation (SEEK UNIT)	
Software Security Construction Skills (SWECOM SKILL SET)	
Computing Essentials (SEEK AREA)	
Computer science foundations (SEEK UNIT)	
	Selects or establishes project coding standards to avoid security vulnerabilities. (SWECOM ACTIVITY)
Construction technologies (SEEK UNIT)	
	Follows recommended coding standards to avoid security vulnerabilities (such as validating input and preventing exception handling mechanisms from revealing too much information about applications and systems). (SWECOM ACTIVITY)

Construction tools (SEEK UNIT)	
Formal construction methods (SEEK UNIT)	
	Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). (SWECOM ACTIVITY)
	Reviews and approves coding standards to avoid security vulnerabilities. (SWECOM ACTIVITY)
Software Security Process Skills (SWECOM SKILL SET)	
Software Verification and Validation (SEEK AREA)	
V & V Terminology and Foundations (SEEK UNIT)	
	Follows project standards in the collection of security assessment metrics. (SWECOM ACTIVITY)
Reviews (SEEK UNIT)	
	Assists in the collection of metrics for security assessment processes. (SWECOM ACTIVITY)
Testing (SEEK UNIT)	
Human Computer UI Testing and Evaluation (SEEK UNIT)	
Problem Analysis and Reporting (SEEK UNIT)	
Software Security Quality Skills (SWECOM SKILL SET)	
Software Quality (SEEK AREA)	
Software Quality Concepts and Culture (SEEK UNIT)	
	Selects appropriate static analysis tools to identify security vulnerabilities. (SWECOM ACTIVITY)

Software Quality Standards (SEEK UNIT	
	Performs code reviews to identify security vulnerabilities. (SWECOM ACTIVITY
Software Quality Processes (SEEK UNIT)	
	Uses static analysis methods to identify security vulnerabilities. (SWECOM ACTIVITY)
	Assists in the installation of static analysis tools. (SWECOM ACTIVITY)
Process Assurance (SEEK UNIT)	
Product Assurance (SEEK UNIT)	

Security integrated SE2004 curriculum recommendation [IEEE-CS04A, IEEE-CS14]

VITA

Robert Evans has a Bachelor of Science degree from the University of North Florida in Computer Information Systems and expects to receive a Master of Science in Computer Information Systems from the University of North Florida, December 2015. Dr. Robert Roggio of the University of North Florida is serving as Robert's thesis advisor.

Robert has had a lifelong interest in information technology. He began working in automated data processing (ADP) while in high school. He continued his interests while serving in the US Navy with a specialty in aviation data analysis. More recently he has expanded his interest after graduation from UNF by working in software development for local companies.

Currently he is working as a senior software developer working for the Department of Defense in navy medicine. He is married to the former Tracy Baerlin and has two children, Carole and Matthew, and one grandchild Alex.