Software Engineering Competency Model

IEEE

IEEE computer society

Software Engineering Competency Model

Version 1.0

SWECOM

A Project of the IEEE Computer Society

Paperback ISBN-10: 0-7695-5373-7
Paperback ISBN-13: 978-0-7695-5373-3

IEEE Computer Society Staff for This Publication

Angela Burgess, Executive Director
Anne Marie Kelly, Associate Executive Director, Director of Governance
Evan M. Butterfield, Director of Products and Services
John Keppler, Senior Manager, Professional Education
Dorian McClenahan, Education Program Product Developer
Kate Guillemette, Product Development Editor
Michelle Phon, Professional Education and Certification Program Coordinator

IEEE Computer Society Products and Services. The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering journals, magazines, conference proceedings, and professional education products. Visit the Computer Society at www.computer.org for more information.

TABLE OF CONTENTS

ABSTRACT

This software engineering competency model (SWECOM) describes competencies for software engineers who participate in developing and modifying software-intensive systems. Skill areas, skills within skill areas, and work activities for each skill are specified. Activities are specified at five levels of increasing competency. Case studies of how the SWECOM model can be used by a manager, an employee, a new hire, or a curriculum designer are provided. The SWECOM-Staffing Gap Analysis and Individual Gap Analysis worksheets are included in an appendix.

1. INTRODUCTION

A competent person has the skills needed to perform, at a given level of competency, the work activities assigned to him or her. *Knowledge*, in this competency model, is different from *skill*: knowledge is what one knows, while skill is what one can do. This document presents a competency model for use by those who develop software, their managers, human-resource personnel, curriculum designers, and others listed in Appendix B of this document. An individual who develops or maintains software might use this competency model to assess his or her current competency levels for various software engineering activities or to develop a plan for improving his or her competencies (such as requirements elicitation, design synthesis, software construction, test planning). A manager (project, functional, or line) might use this competency model to inventory staff skills and identify areas for needed additions and improvements. Additionally, a manager might use this model to counsel individual employees, or HR personnel might use the model to identify needed training and recruitment activities. Each activity in this competency model is described at five levels of competency.

Skill areas in this competency model include skills that are decomposed into activities, rather than job roles, because job roles are typically dependent on the organizational environment in which the work activities occur. The activities in this model can be grouped into job roles by organizations, organizational units, or projects to satisfy their needs.

This competency model is termed the Software Engineering Competency Model (SWECOM). It has been validated by invited subject

matter and interested public reviewers. Subsequent revisions have been made in response to those reviews. SWECOM contributors are listed in Appendix A.

Appendix B lists the intended audience for SWECOM. Appendix C includes use cases to indicate how managers, employees, and new hires might find SWECOM useful. Appendix D includes gap analysis worksheets for use by individual practitioners and those who do staffing for projects and organizational units. A Glossary of Terms provides definitions of terms whose meanings, as used in SWECOM, may differ from conventional meanings.

SWECOM includes skill areas, skills, and activities for individuals who develop and maintain software (that is, software engineers and others). SWECOM is based on the following primary references:

- *SWEBOK Guide* Version 3 (*Guide to the Software Engineering Body of Knowledge*),
- ISO/IEEE Standard 12207 (software engineering processes),
- Elements of ISO/IEEE Standard 15288 (systems engineering processes) applicable to the development of software-intensive systems,
- Relevant material in SEBoK (Systems Engineering Body of Knowledge) and GRCSE (Graduate Reference Curriculum for Systems Engineering),
- The Software Assurance Competency Model,
- GswE2009 (graduate software engineering curriculum guidelines), and
- SE2004 (undergraduate software engineering curriculum guidelines).

The references section of this document provides citations for these foundational documents.

This competency model adds to the growing body of knowledge that characterizes the software engineering profession and software engineering professionals. It is based on, and supplements, the information found in the primary references and in the extensive list of references found in that section. SWECOM is presented as a framework that can be tailored to fit the needs of organizations,

programs, and projects. It is not a prescriptive model of the software engineering profession or a characterization of a software engineering professional. Various organizations, agencies, and other institutions may choose to adopt and enforce particular elements of the model to fit their needs and extend SWECOM in various ways.

SWECOM covers technical skills but does not include project management or general management skills other than to identify the behavioral attributes and skills of effective software developers and the leadership skills needed for software project technical leaders of various skill areas. The *PMBOK® Guide—Fifth Edition* [PMBOK 2013], the *Software Extension to the PMBOK® Guide—Fifth Edition* [SWX 2013], and many other references address project management and general management. Also, SWECOM does not recommend specific software tools or development methods (such as waterfall, Scrum, XP).

2. SWECOM AND THE US IT COMPETENCY MODEL

SWECOM includes elements similar to those in the US Department of Labor Information Technology (US IT) Competency Model, which was developed to identify the knowledge, skills, and abilities needed for workers to perform successfully in the field of information technology [INFOCOMP 2012].[1]

Table 1 indicates the correspondences between the US IT Competency Model and the analogous elements of SWECOM.

1 "Information technology" is broadly interpreted in the US IT competency model.

Table 1. Correspondences between the US IT Competency Model and SWECOM

US IT Competency Model	SWECOM
Personal Effectiveness	Behavioral Attributes and Skills
Academic Competencies	Requisite Knowledge for SWECOM Technical Skills
Workplace Competencies	Cognitive Skills
Industry-Wide Technical Competencies	SWECOM Technical Skills
Industry-Sector Technical Competencies	Possible extensions to SWECOM for software applications, embedded software, and domain-specific competencies (for example, health sciences, communication, automotive domains)
Management Competencies	Skills related to scheduling, budgeting, and resource management are excluded from SWECOM
Occupation-Specific Requirements	Excluded from SWECOM

As indicated in Table 1, knowledge, behavioral attributes and skills, and cognitive skills in SWECOM are the counterparts of personal effectiveness, academic competencies, and workplace competencies in the US IT Competency Model. Technical skills are the primary focus of SWECOM and are the counterpart of industry-wide technical competencies. The other elements of SWECOM are included to support the technical competencies. Industry-sector competencies for various sectors of the software engineering industry represent extensions that could be added to SWECOM. Management competencies other than leadership skills related to leading technical contributors are not included because management includes a distinct, though related, set of competencies that are covered in the PMI *Guide to the Project Management Body of Knowledge* (*PMBOK® Guide*) [PMBOK 2013], the *Software Extension to the PMBOK® Guide* [SWX 2013], and other similar documents. Occupation-specific requirements include factors such as certifications and licensing requirements needed to pursue specific occupations; they are not included in SWECOM.

3. THE ELEMENTS OF SWECOM

The elements of SWECOM are illustrated in Figure 1. Cognitive skills and behavioral attributes and skills are described below. These foundations are not unique to SWECOM but were developed for SWECOM as necessary for the effective performance of software engineering technical activities. Requisite knowledge is the intellectual basis for the software engineering profession. The references listed above, those cited in the references section, and those in the consolidated reference list in the *SWEBOK Guide* (www.swebok.org, Appendix C) characterize requisite knowledge.

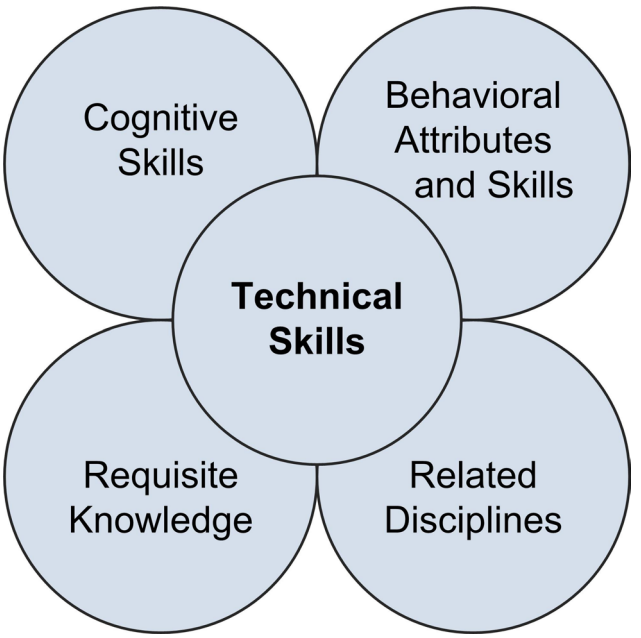

Figure 1. The Elements of SWECOM

Related disciplines include but are not limited to:

- Computer Engineering,
- Computer Science,
- General Management,
- Mathematics,
- Project Management,

- Quality Management, and
- Systems Engineering.

There are many related disciplines; these listed are *closely* related disciplines, as cited in [SWEBOK 2014].

Cognitive skills apply across all the skill areas, skills, and activities of SWECOM. They are exhibited in the ability to apply knowledge and reasoning while performing SWECOM activities within technical skill areas. Competency levels for cognitive skills are not included in SWECOM, but cognitive skills become increasingly important at higher levels of technical competency because the scope and complexity of work activities increases and expands as the levels of competency and related job assignments increase. Some examples of cognitive skills are listed and briefly described in Table 2.

As shown in Table 2, the SWECOM cognitive skills include four classifications. It should be emphasized that these classifications are not independent: the skills listed across classifications overlap and combine to support effective cognitive competencies. Furthermore, the list in Table 2 is intended to be illustrative—not exhaustive—of cognitive skills for software engineers. Citations that provide the basis for and details of these cognitive skills are listed in the references section of this document.

Table 2. SWECOM Cognitive Skills

Cognitive Skills	Examples
Reasoning provides the basis for making decisions in a logical and effective manner.	Inductive Reasoning Deductive Reasoning Heuristic Reasoning Use of Abstraction Hierarchical and Associative Reasoning
Analytical skills are related to techniques that involve data collection, organization and aggregation of data, and analysis and evaluation in order to draw conclusions or make decisions.	Application of Measurement Principles Statistical/Data Analysis Root Cause Analysis Risk Identification and Analysis Impact Analysis

Table 2. SWECOM Cognitive Skills	
Cognitive Skills	**Examples**
Problem solving is concerned with various methods that employ reasoning, analytic techniques, and prioritizing information to solve problems.	Divide and Conquer Stepwise Refinement Top-down Approach Bottom-up Approach Analogy and Reuse Patterns and Pattern Recognition Iterative and Incremental Approaches
Innovation involves skills used to create models and abstractions that support analysis and problem solving.	Brainstorming Prototype Development Modeling and Simulation

Behavioral attributes and skills are exhibited in the ability to productively apply knowledge, cognitive skills, and technical skills; they are not unique to software engineering but allow software engineers to effectively contribute to desired outcomes. Some important behavioral attributes and skills for software engineers are listed in Table 3; other behavioral attributes and skills could be added.

Table 3. SWECOM Behavioral Attributes and Skills	
Aptitude	Exhibited by the ability to effectively perform a software engineering task. Aptitude is not the same as knowledge or skill but rather indicates the ability (either intuitive or learned) to apply knowledge in a skillful way.
Initiative	Exhibited by enthusiastically starting and following through on a software engineering work task.
Enthusiasm	Exhibited by expressing and communicating interest in performing a work task.
Work ethic	Exhibited by being reliable, acquiring new skills, and being willing to perform work tasks.
Willingness	Exhibited by undertaking a task when asked and capably performing it, even if it is a task the individual is not enthusiastic about performing.
Trustworthiness	Demonstrated over time by exhibiting ethical behavior, honesty, integrity, and dependability in an individual's decisions and actions.

Table 3. SWECOM Behavioral Attributes and Skills	
Cultural sensitivity	Exhibited by an awareness of and accommodation for differences in communication styles, social interactions, dress codes, and overall behavior based on ethnic, religious, gender orientation, and other behavioral characteristics.
Communication skills	Exhibited by expressing concepts, techniques, thoughts, and ideas in both oral and written forms in a clear and concise manner while interacting with team members, managers, project stakeholders, and others; includes effective listening.
Team participation skills	Exhibited by working enthusiastically and willingly with other team members while collaborating on shared tasks.
Technical leadership skills	Exhibited by effectively communicating a vision, strategy, method, or technique that is then accepted and shared by team members, managers, project stakeholders, and others.

Behavioral attributes and skills apply to all elements and at all levels of technical skill areas, skills, and activities. Behavioral attributes and skills and cognitive skills are not specified by competency level; however, increasing competency in cognitive skills and behavioral attributes and skills becomes more important as the levels of technical competencies, the scope of responsibilities, and the breadth of interactions increase.

4. SWECOM TECHNICAL SKILLS

Technical skills and associated activities are the primary focus of SWECOM; they are grouped as life cycle skill areas and crosscutting skill areas. A life cycle skill area is one that includes skills needed to accomplish various work activities within a phase of software development or sustainment—for example, software requirements engineering. Life cycle skill areas are categorized by typical phases of software development and modification. In practice, software phases are often intermixed, interleaved, and iterated in various ways; however, no implication of development processes (for example, predictive versus adaptive) is intended.

A crosscutting skill area is one that applies across all life cycle skill areas (for example, quality assurance) and, in some cases, a crosscutting skill may apply to other crosscutting skill areas (for example, a software process model). Crosscutting skill areas are sometimes called "specialty disciplines" that are practiced by specialists in those skill areas (such as safety, security, systems engineering). Software engineers who are competent in one or more life cycle skill areas typically have some working knowledge of crosscutting skill areas.

The five life cycle skill areas and eight crosscutting skill areas of SWECOM are listed in Tables 4 and 5, respectively. The references cited in the tables are in the references section of this document; they provide the knowledge foundations for each skill area.

Table 4. Software Engineering Life Cycle Skill Areas and Skills

Life Cycle Skill Areas	Skills
Software Requirements Skills *References:* [ACM 2004] [Laplante 2009] [Robertson 2012] [SWEBOK 2014] [Wiegers 2013]	Software Requirements Elicitation Software Requirements Analysis Software Requirements Specification Software Requirements Verification and Validation Software Requirements Process and Product Management
Software Design Skills *References:* [IEEE 1016-2009] [IEEE 12207-2008] [IEEE 15528-2008] [SWEBOK 2014]	Software Design Fundamentals Software Design Strategies and Methods Software Architectural Design Software Design Quality Analysis and Evaluation
Software Construction Skills *References:* [ACM 2004] [Fowler 1999] [Hunt 1999] [McConnell 2004] [SWEBOK 2014]	Software Construction Planning Managing Software Construction Detailed Design and Coding Debugging and Testing Integrating and Collaborating

Table 4. Software Engineering Life Cycle Skill Areas and Skills	
Life Cycle Skill Areas	**Skills**
Software Testing Skills *References:* [IEEE 730-2002] [IEEE 829-2008] [IEEE 1012-2012] [Myers 2011] [SWEBOK 2014]	Software Test Planning Software Testing Infrastructure Software Testing Techniques Software Testing Measurement and Defect Tracking
Software Sustainment Skills *References:* [IEEE 12207-2008] [ISO/IEC/IEEE 24765:2010] [IEEE 828-2012] [Lapham 2006] [SWEBOK 2014]	Software Transition Software Support Software Maintenance

Table 5. Software Engineering Crosscutting Skill Areas	
Crosscutting Skill Areas	**Skills**
Software Process and Life Cycle Skills *References:* [IEEE 12207-2008] [IEEE 15528-2008] [SWEBOK 2014]	Software Development Life Cycle Implementation Process Definition and Tailoring Process Implementation and Management Process Assessment and Improvement
Software Systems Engineering Skills *References:* [IEEE 12207-2008] [IEEE 15528-2008] [SEBoK 2013] [SWEBOK 2014]	System Development Life Cycle Modeling Concept Definition System Requirements Engineering System Design Requirements Allocation Component Engineering System Integration and Verification System Validation and Deployment System Sustainment Planning
Software Quality Skills *References:* [IEEE 730-2002] [IEEE 829-2008] [IEEE 1012-2012] [IEEE 12207-2008] [IEEE 15528-2008] [SWEBOK 2014]	Software Quality Management (SQM) Reviews (review, walkthrough, inspection) Audits (concentrate on both product and process, but are done by an independent internal or external organization) Statistical Control

Table 5. Software Engineering Crosscutting Skill Areas	
Crosscutting Skill Areas	**Skills**
Software Security Skills *References:* [Allen 2008] [BITS 2012] [Hilburn 2013] [Merkow 2010] [Seacord 2005]	Requirements Design Construction Testing Process Quality
Software Safety Skills *References:* [Hilburn 2013] [IEEE 12207-2008] [Leveson 1995] [Stephans 2004] [Vincoli 2006]	Requirements Design Construction Testing Process Quality
Software Configuration Management Skills *References:* [Aiello 2010] [Babich 1986] [IEEE 828-2012] [SWEBOK 2014]	Plan SCM Conduct SCM Manage Software Releases
Software Measurement Skills *References:* [IEEE 12207-2008] [IEEE 15528-2008] [IEEE 15939-2008] [SWEBOK 2014]	Plan Measurement Process Perform Measurement Process
Human-Computer Interaction Skills *References:* [ISO 9241-210:2010] [Rogers 2011] [SWEBOK 2014]	Requirements Interaction Style Design Visual Design Usability Testing and Evaluation Accessibility

The activities for each skill in Tables 4 and 5 are listed in each skill area's Tables A and B; see below. Table A lists the activities for each skill and Table B lists activities by competency level. SWECOM does not address competency in using tools or adherence to prescribed standards to accomplish activities because these will be specific to organizations and projects.

5. SWECOM COMPETENCY LEVELS

SWECOM is organized by skill area (for example, software requirements), skills within skill areas (for example, software requirements elicitation), and activities within skills (for example, prototyping to elicit requirements). Activities are specified at five levels of competency:

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

In general, a Technician follows instructions, an Entry Level Practitioner assists in performance of an activity or performs an activity with supervision; a Practitioner performs activities with little or no supervision; a Technical Leader leads individuals and teams in the performance of activities; and a Senior Software Engineer modifies existing methods and tools and creates new ones. Some organizations may choose to merge the Technician and Entry Level Practitioner levels. A Senior Software Engineer may serve as a "chief engineer" for a software organization and some Senior Software Engineers may be recognized as industry experts who contribute to shaping and advancing the profession of software engineering.

In addition to the activities specified at the various competency levels, an additional competency of all software engineers is to instruct and mentor others, as appropriate, in the methods, tools, and techniques used to accomplish those activities. For example, a Technician or Entry Level Practitioner might instruct or mentor others on the use of configuration management tools as needed to perform their activities, or a Team Leader might instruct or mentor a Practitioner on how to lead inspections and reviews.

The following notations are also used in SWECOM:

- Follows (F),
- Assists (A),

- Participates (P),
- Leads (L), and
- Creates (C).

For the requirements prototyping activity cited above, a Technician would be competent to use software tools while following instructions (F) to create prototypes. An Entry Level Practitioner would be competent to assist in creating prototypes and to develop prototypes under supervision (A); a Practitioner would create prototypes and interact with customers and users in evaluating the prototypes (P); a Technical Leader would supervise and lead prototyping activities (L); and a Senior Software Engineer would create new approaches to prototyping (C).

There may be situations where an Entry Level Practitioner might be competent, for example, to lead a prototyping activity, or a Technical Leader might be competent to create a new approach to prototyping, so notations are used in SWECOM to distinguish specific competencies from the named competency levels when it is appropriate.

A Practitioner, for example, might be competent to either participate in an activity (P) or lead the activity (L), depending on the scope and complexity of the work to be accomplished. In this case, the activity is labeled (P/L) at the Practitioner level. Similarly, an Entry Level Practitioner might be competent to assist or fully participate in an activity, which would be labeled (A/P).

SWECOM does not prescribe the knowledge level or years of experience associated with these competency levels; however, the following general guidelines are typical:

An individual who is competent at the Technician level to perform the activities in one or more skills or skill areas might have some advanced education (for example, a two-year US associate's degree or equivalent), one or more industrial certifications, and any number of years of experience.

An individual who is competent as an Entry Level Practitioner to perform the activities in one or more skills or skill areas would probably have requisite knowledge equivalent[2] to that provided by an

2 Knowledge equivalence might be gained by a combination of education, mentoring, training, and on-the-job experience.

ABET-accredited software engineering degree program or equivalent and zero to four or five years of relevant experience.[3]

An individual who is competent at the Practitioner level to perform the activities in one or more skills or skill areas would probably have knowledge equivalent to or greater than that of an Entry Level Practitioner, might have a master's degree in software engineering or a related discipline, and would probably have more than five years of experience in the relevant skill areas.

An individual who is competent as a Technical Leader for one or more SWECOM activities, skills, or skill areas would likely have relevant knowledge and experience equal to or greater than that of a Practitioner plus the behavioral attributes and skills needed to be an effective technical leader.

A Senior Software Engineer is an individual who is competent to develop policies, procedures, and guidelines for the technical processes and work products within an organizational unit that is engaged in software engineering.

These characterizations of education and experience are examples and not to be interpreted as prescriptive requirements.

Some activities may not have corresponding lower level activities. For example, conducting an impact analysis to determine the effect of modifying or adding requirements for product security or performance might be a Practitioner skill and not an activity that a person at a lower level of competency would be competent to perform.

An individual may be at different levels of competency for different skill areas, skills within skill areas, and activities within skills, depending on his or her educational background, work experiences, and aptitude. The SWECOM activities, skills, and skill areas are presented as a framework that can be tailored to fit the needs of individual organizations, programs, and projects. Some organizations may choose to use SWECOM in a prescriptive manner by requiring software engineers who are competent in a skill area at a given competency level to be competent in all of the skills and activities in that skill area at that level and at all lower levels. Other organizations, programs, and projects may use SWECOM to pick and

3 Relevant experience is the experience needed to acquire ability, at a given level of competency, for a SWECOM skill area, skill, or activity.

choose skill areas, skills, and activities needed for particular missions, programs, or projects without regard to other competencies and competency levels.

An example of activities competency levels from the requirements management skill within the software requirements skill area illustrates the approach taken in subsequent sections of this competency model; see Table 6. In general, these notations correspond to the five levels of competency. In some cases, an individual at a lower level of skill competency may be competent to perform some activities—but not all—at a higher level. For example, an Entry Level Practitioner may be competent to perform traceability analysis (P), or a Practitioner may be competent to lead certain activities (L).

Table 6. Competency Levels for Software Requirements Management Work Activities

Skill Area: Software Requirements					
Skill: Requirements Management					
Competency Levels	Technician	Entry Level Practitioner	Practitioner	Team Leader	Senior Software Engineer
Activities	1. Follows defined procedures to support requirements management (F)	1. Assists requirements management through the use of appropriate tools (A)	1. Implements requirements management plans for projects (P/L)	1. Prepares requirements management plans for projects (L)	1. Modifies existing and creates new guidelines, templates, tools, and techniques for requirements management (C)

Note that in some cases (such as for the Practitioner level in Table 6) an individual may be competent to either participate in or lead a work activity such as implementing a requirements management plan. Whether that individual is competent to participate or lead may depend on the size, scope, and complexity of the project and product; in such cases, the notation is (P/L).

6. EMPLOYER AND INDIVIDUAL GAP ANALYSIS

Appendix D includes two worksheets similar to those in the US IT Competency Model [INFOCOMP 2012]. The first spreadsheet (SWECOM Staffing Gap Analysis Worksheet) is for use by managers, human resources personnel, and others who analyze available and needed skills within an organizational unit.

The second spreadsheet (SWECOM Individual Gap Analysis) is for use by an individual who desires to assess his or her levels of competency for different skills and activities at different competency levels. An individual can use the spreadsheet for self-assessment or an individual and manager can use it as a basis for developing a plan of improvement for the individual; the improvement plan might include future work assignments, mentoring, and/or additional education and training.

7. SWECOM VALIDATION

SWECOM has been validated by 22 subject matter reviewers and 40 public reviewers.

Many narrative review comments were received from both subject matter reviewers (SMEs) and public reviewers. The SWECOM author team adjudicated all comments and informed reviewers of their decisions.

Members of the SWECOM team interviewed six software engineering professionals. The purpose of the interviews was to determine the value of a software engineering competency model and the relevancy and usefulness of various SWECOM elements (for example, cognitive attributes, behavioral attributes and skills, skill areas, competency levels, and so forth). These interviews also allowed the SWECOM developers to conduct a "sanity check" on SWECOM before releasing a draft for external review.

The interview results can be summarized as follows:

- All those interviewed had degrees in computing-related disciplines and 12 to 29 years of experience in the software industry, and were serving in mid- to high-level positions in their organizations (for example, system architect, quality assurance director, software development manager, technical support manager, software development director).
- There was unanimous agreement that SWECOM will provide valuable support for recruiting, evaluating, developing, and advancing software engineering professionals.
- Most interviewees voiced the opinion that nontechnical competencies were essential to the success of a software engineering professional: good people skills, flexibility, and the ability to communicate, work in teams, work with customers, learn new things, and work with people from different cultures. As a result of the last observation, the SWECOM developers added a behavioral attribute of "Cultural Sensitivity."
- Most of those interviewed thought five or six levels of competency were appropriate. However, none of those interviewed who have used other competency models have ever before used the Technician Level or equivalent.
- Some interviewees expressed the view that no degree or minimum years of experience should be specified for the competency levels. SWECOM only describes "typical" backgrounds, and no requirements or precise expectations are stated.
- There were no recommendations for major changes to SWECOM.

These six interviews did not provide a statistically significant sample of opinions but they did indicate that the SWECOM effort is well conceived. The SME and public review comments provided many valuable recommendations, but none invalidated the SWECOM concept.

The following sections of this document specify life cycle and cross-cutting skill areas plus skills and activities at various competency levels within each skill area. Each skill area includes two tables: Table A lists skills and corresponding activities for that skill area, and Table B lists the activities across all five competency levels.

8. ACKNOWLEDGEMENTS

Appendix A lists individuals who developed this competency model, the subject matter expert reviewers, the public reviewers, and those who were interviewed.

9. REFERENCES

[Abran 2010] Alain Abran, *Software Metrics and Software Metrology*, Wiley-IEEE Computer Society Press, 2010.

[ACM 2004] ACM/IEEE-CS Joint Task Force on Computing Curricula, *Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, Aug. 2004; www.acm.org/education/curricula.html.

[Aiello 2010] Bob Aiello and Leslie Sach, *Configuration Management Best Practices: Practical Methods that Work in the Real World*, Addison-Wesley Professional, 2010.

[Allen 2008] Julia Allen et al., *Software Security Engineering: A Guide for Project Managers*, Addison-Wesley Professional, 2008.

[Babich 1986] Wayne A. Babich, *Software Configuration Management: Coordination for Team Productivity,* Addison-Wesley, 1986.

[BITS 2012] *BITS Software Assurance Framework*, Financial Services Roundtable, 2012; www.bits.org/publications/security/BITSSoftwareAssurance0112.pdf.

[Bozzano 2010] Marco Bozzano and Adolfo Villafiorita, *Design and Safety Assessment of Critical Systems,* CRC Press, 2010.

[Buxton 2007] Bill Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*, Morgan Kaufmann Publishers, 2007.

[CMMI 2014] *Capability Maturity Model Integrated*, CMMI Institute, 2014; http://cmmiinstitute.com.

[Fowler 1999] M. Fowler et al., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

[Hilburn 2013] Thomas Hilburn et al., *Software Assurance Competency Model*, Technical Note CMU/SEI-2013-TN-004, Software Engineering Institute, Mar. 2013; http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=47953.

[Hunt 1999] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley, 1999.

[IEEE 730-2002] *IEEE Std. 730-2002, IEEE Standard for Software Quality Assurance Plans*, IEEE, 2002.

[IEEE 828-2012] *IEEE Std. 828-2012, IEEE Standard for Configuration Management in Systems and Software Engineering*, IEEE, 2012.

[IEEE 829-2008] *IEEE Std. 829-2008, IEEE Standard for Software and System Test Documentation,* IEEE, 2008.

[IEEE 1012-2012] *IEEE Std. 1012-2012, IEEE Standard for System and Software Verification and Validation*, IEEE, 2012.

[IEEE 1016-2009] *IEEE Std. 1016-2009, IEEE Standard for Information Technology-Systems Design—Software Design Descriptions,* IEEE, 2009.

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes,* IEEE, 2008.

[IEEE 15939-2008] *IEEE Std. 15939-2008, Standard Adoption of ISO/IEC 15939:2007 System and Software Engineering Measurement Process*, IEEE, 2008.

[IEEE/ISO/IEC 24765-2010] *IEEE/ISO/IEC 24765:2010, Systems and Software Engineering—Vocabulary*, IEEE, 2010.

[INFOCOMP 2012] *Information Technology Competency Model*, Employment and Training Administration, United States Department of Labor, Sep. 2012; www.careeronestop.org/competencymodel/pyramid_download.aspx?IT=Y.

[ISO 9241-210:2010] *ISO 9241-210:2010, Ergonomics of Human-System Interaction*, ISO, 2010.

[ISO/IEC 25060:2010] ISO/IEC 25060:2010, *Systems and Software Engineering—Systems and Software Product Quality Requirements and Evaluation (SQuaRE)—Common Industry Format (CIF) for Usability: General Framework for Usability-Related Information*, ISO, 2010.

[ISO/IEC/IEEE 15289:2011] *ISO/IEC/IEEE 15289:2011, Systems and Software Engineering—Content of Life-Cycle Information Products (Documentation),* ISO, 2011.

[Kan 2002] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd ed., Addison-Wesley, 2002.

[Lams 2009] Alex van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, John Wiley and Sons, Inc., 2009.

[Lapham 2006] M. Lapham and C. Woody, *Sustaining Software-Intensive Systems*, CMU/SEI-2006-TN-007, Software

Engineering Institute, 2006; http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=7865.

[Laplante 2009] Phillip A. Laplante, *Requirements Engineering for Software and Systems*, 2nd ed., CRC Press, 2009.

[Laplante 2013] Phillip A. Laplante, Beth Kalinowski, and Mitchell Thornton, "A Principles and Practices Exam Specification to Support Software Engineering Licensure in the United States of America," *Software Quality Professional*, vol. 15, no. 1, Jan. 2013, pp. 4–15.

[Leveson 1995] N. Leveson, *Safeware: System Safety and Computers*, Addison-Wesley, 1995.

[Leveson 2011] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, The MIT Press, 2011.

[McConnell 2004] S. McConnell, *Code Complete*, 2nd ed., Microsoft Press, 2004.

[Merkow 2010] M. Merkow and L. Raghavan, *Secure and Resilient Software Development*, CRC Press, 2010.

[Myers 2011] Glenford J. Myers, *The Art of Software Testing*, 3rd ed., Wiley, 2011.

[PMBOK 2013] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Fifth Edition*, Project Management Institute, 2013.

[Pohl 2010] Klaus Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer-Verlag, 2010.

[Rierson 2013] Leanna Rierson, *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*, CRC Press, 2013.

[Robertson 2012] Suzanne Robertson and James C. Robertson, *Mastering the Requirements Process: Getting Requirements Right*, 3rd ed., Addison-Wesley Professional, 2012.

[Rogers 2011] Y. Rogers, H. Sharp, and J. Preece, *Interaction Design: Beyond Human Computer Interaction*, 3rd ed., Wiley, 2011.

[RTCA DO-178C] RTCA, Inc., *Software Considerations in Airborne Systems and Equipment Certification*, DO-178C/ED-12C, 13 Dec. 2011.

[Seacord 2005] R. Seacord, *Secure Coding in C and C++*, Addison-Wesley, 2005.

[SEBoK 2013] A. Pyster and D.H. Olwell, eds., *The Guide to the Systems Engineering Body of Knowledge (SEBoK),* The Trustees of the Stevens Institute of Technology, vol. 1.2, 2013; http://sebokwiki.org.

[Stephans 2004] R.A. Stephans, *System Safety for the 21st Century: The Updated and Revised Edition of System Safety 2000*, Wiley, 2004.

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

[SWX 2013] Project Management Institute and IEEE Computer Society, *Software Extension to the PMBOK® Guide—Fifth Edition*, Project Management Institute, 2013.

[Vincoli 2006] J.W. Vincoli, *Basic Guide to System Safety*, Wiley, 2006.

[Westfall 2009] Linda Westfall, *The Certified Software Quality Engineering Handbook*, Quality Press, 2009.

[Wiegers 2013] Karl E. Wiegers and Joy Beatty, *Software Requirements*, 3rd ed., Microsoft Press, 2013.

10. GLOSSARY OF TERMS

This glossary provides definitions of terms whose meanings, as used in SWECOM, may differ from conventional meanings. Other terms used in SWECOM are intended to convey the meanings in IEEE/ISO/IEC Standard 24765:2010, *Systems and Software Engineering—Vocabulary*, IEEE, 2010 (SEVOCAB). The *Guide to the Software Engineering Body of Knowledge* [SWEBOK 2014] also provides detailed discussions of many of the terms used in SWECOM.

Terms italicized in the definitions of other terms are also defined in this glossary.

Activity: a self-contained unit of work to be performed. Activities are the smallest units of technical skills in SWECOM.

Behavioral Attribute: a characteristic of personality and character that enables an individual to apply knowledge, experience, and *cognitive attributes* to perform *activities* in a productive manner within the work environment.

Cognitive Skill: a characteristic of intellect that allows an individual to apply knowledge and reasoning ability while performing *activities* within technical *skill areas*.

Competency: the demonstrated ability to perform work *activities* at a stated *competency level*.

Competency Level: one of five increasing levels of ability to perform an *activity*; denoted as *Technician, Entry Level Practitioner, Practitioner, Technical Leader,* or *Senior Software Engineer*.

Entry Level Practitioner: an individual who is competent to assist in performing an *activity* or to perform activities with some supervision.

Gap Analysis: the process of specifying the competencies an individual or organization has, the competencies needed, and gaps between what is had and what is needed.

Senior Software Engineer: an individual who is competent to create new—and modify existing—processes, procedures, methods, and tools for performing *activities*, groups of *activities* within *skills*, and *skills* within *skill areas*.

Practitioner: an individual who is competent to perform an *activity* with little or no supervision.

Skill: a grouping of logically related *activities*.

Skill Area: a grouping of logically related *skills*.

Technical Leader: an individual who is competent to lead and direct participants in the performance of *activities* in a *skill* or *skill area*.

Technician: an individual who is competent to follow instructions while performing an *activity*.

Usability Test: a test case that states what the user needs to do but does not tell the user how to do it. It measures the user interface's ability to support user behavior.

11. SOFTWARE REQUIREMENTS SKILL AREA

Software requirements engineering consists of activities performed to discover what functional and nonfunctional attributes and interfaces a software system should have to satisfy the needs of the customer. It also includes analysis and management activities performed in order to discover flaws in requirements artifacts and to manage the requirements engineering process.

REFERENCES

[ACM 2004] ACM/IEEE-CS Joint Task Force on Computing Curricula, *Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, Aug. 2004; www.acm.org/education/curricula.html.

[Laplante 2009] Phillip A. Laplante, *Requirements Engineering for Software and Systems*, 2nd ed., CRC Press, 2009.

[Robertson 2012] Suzanne Robertson and James C. Robertson, *Mastering the Requirements Process: Getting Requirements Right*, 3rd ed., Addison-Wesley Professional, 2012.

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

[Wiegers 2013] Karl E. Wiegers and Joy Beatty, *Software Requirements*, 3rd ed., Microsoft Press, 2013.

Table A11

Software Requirements Skill Sets	Software Requirements Activities
Software Requirements Elicitation	• Identifies stakeholders for elicitation of requirements. • Engages stakeholders in elicitation of requirements. • Uses appropriate methods to capture requirements. • Negotiates conflicts among stakeholders during elicitation.
Software Requirements Analysis	• Uses appropriate domain analysis techniques. • Performs analysis of requirements for feasibility and emergent properties.
Software Requirements Specification	• Uses appropriate notations for describing requirements.
Software Requirements Verification and Validation	• Checks requirements for accuracy, lack of ambiguity, completeness, consistency, traceability, and other desired attributes. • Constructs and analyzes prototypes. • Negotiates conflicts among stakeholders during verification.
Software Requirements Process and Product Management	• Uses appropriate methods for management of requirements, including configuration management.

The following notations are used in Table B11: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B11

Software Requirements Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Requirements Elicitation			1. Identifies important stakeholders. (P/L)		
		1. Assists in engaging different stakeholders to determine needs and requirements. (A)	2. Engages different stakeholders to determine needs and requirements. (P)		
		2. Assists in applying different methods to the project as appropriate to elicit requirements. (A)	3. Applies different methods to the project as appropriate to elicit requirements. (P)		

Table B11					
Software Requirements Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Requirements Elicitation	1. Assists requirements engineers with preparation of surveys and other elicitation instruments. (F/A)			1. Selects appropriate methods to engage and communicate with stakeholders in requirements activities. (L)	1. Creates new ways to engage and communicate with stakeholders, the management team, and developers in requirements activities (C)
			4. Assists in negotiating conflicts between stakeholders in requirements elicitation. (A)	2. Negotiates conflicts between stakeholders in requirements elicitation. (P/L)	
Software Requirements Analysis		1. Assists in domain analysis. (A)	1. Selects the most appropriate domain analysis methods. (P/L)	1. Leads identification of emergent properties and requirements throughout the software development life cycle. (P)	1. Creates new domain analysis methods. (C)

Table B11

Software Requirements Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Requirements Analysis			2. Identifies emergent properties and requirements throughout the software development life cycle. (P)		
Software Requirements Specification	1. Assists with preparation of requirements for consistency with internal and published standards. (F/A)	1. Prepares requirements documentation including descriptions of interfaces and functional and non-functional requirements. (P)	1. Selects the most appropriate formal and informal notations for describing interfaces and functional and non-functional requirements. (P/L)	1. Leads development of the SRS. (L)	1. Creates new requirements specification methods. (C)
				2. Selects the most appropriate formal and informal notations for describing interfaces and functional and non-functional requirements. (L)	

Table B11

Software Requirements Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Requirements Verification and Validation		1. Reviews specifications of requirements for errors and omissions. (P)	1. Reviews specifications of requirements for errors and omissions. (L)	1. Selects the most appropriate formal and informal requirements validation and verification techniques. (L)	1. Creates new requirements validation and verification techniques. (C)
	2. Assists in prototype construction and testing. (F/A)		2. Creates prototypes of different types as needed. (P)		
			3. Assists in negotiating conflicts between stakeholders in requirements verification. (A)	2. Negotiates conflicts between stakeholders in requirements verification. (P/L)	
Software Requirements Process Management	1. Follows and applies defined processes for requirements engineering with guidance. (F/A)	1. Assists in applying defined processes for requirements engineering. (A)	1. Performs tradeoff analysis of requirements activities. (P/L)		1. Sets strategy and direction for the requirements process across projects and functional units of an organization. (L)

12. SOFTWARE DESIGN SKILL AREA

Software design skills are used to develop and describe the software architecture of a system based on its software requirements: this consists of a description of how software is decomposed into components and the interfaces between those components. The components are specified at a level of detail that enables their construction. This skill area also includes skills related to processes and techniques for software design quality, analysis, and evaluation.

REFERENCES

[IEEE 1016-2009] *IEEE Std. 1016-2009, IEEE Standard for Information Technology-Systems Design—Software Design Descriptions*, IEEE, 2009.

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes*, IEEE, 2008.

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

Table A12	
Software Design Skill Sets	**Software Design Activities**
Software Design Fundamentals	• Employ enabling techniques (such as abstraction, coupling/cohesion, information hiding, and so forth) in software design. • Apply exception handling and fault tolerance techniques in software design. • Use restructuring and refactoring methods in software design. • Apply, as appropriate, design techniques in the areas of concurrency, event handling, data persistence, or distributed software.
Software Design Strategies and Methods	• Determine the process and strategy to be used in software design (such as top-down or bottom-up, stepwise refinement, use of patterns and pattern languages, iterative and incremental processes, and so forth). • Select and apply the appropriate design methodology (such as a structural or object-oriented approach). • Consider design alternatives and perform trade-off analysis. • Manage software design activities.
Software Architectural Design	• Use architectural styles, views, models, and patterns to specify the high-level organization of a software system. • Specify the component interfaces. • Design software components and modules using models, design patterns, notations, and diagramming techniques.
Software Design Quality Analysis and Evaluation	• Utilize software design reviews. • Perform static analysis tasks to evaluate design quality. • Develop and use simulation and prototypes to evaluate software design quality. • Manage requirements change.

The following notations are used in Table B12: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B12

Software Design Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Design Fundamentals	1. Assists software designers with tools and techniques for gathering information about application and use of software design fundamentals. (F/A)	1. Assists in the application of enabling techniques in the design of software components and modules. (A)	1. Applies enabling techniques (such as abstraction, coupling/cohesion, information hiding, and so forth) to the design of software components and modules. (P)	1. Evaluates the effectiveness of the application of software design enabling techniques. (P/L)	1. Analyzes and makes recommendations related to organization-wide application of software design fundamentals. (C)
		2. Assists in the application of design techniques in the areas of concurrency, event handling, data persistence, or distributed software. (A)	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)	2. Provides direction and advice on methods and techniques to be used in the areas of concurrency, event handling, or distributed software. (L)	

Table B12					
Software Design Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Design Fundamentals		3. Assists in the application of exception handling and fault tolerance techniques in the design of software components and modules. (A)	3. Applies exception handling and fault tolerance techniques in the design of software components and modules. (P)		
		4. Assists in the use of restructuring and refactoring methods in the design of software components and modules. (A)	4. Uses restructuring and refactoring methods in the design of software components and modules. (P)		

Table B12

Software Design Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Design Strategies and Methods	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)	1. Assists in the application of the designated software design strategy and methodology to create a software design (such as an incremental object-oriented approach). (A/P)	1. Applies the designated software design strategy and methodology to create a software design (such as an incremental object-oriented approach). (P)	1. Determines the process and strategy to be used in software design at the project level (such as top-down or bottom-up, stepwise refinement, use of patterns and pattern languages, iterative and incremental processes, and so forth). (L)	1. Examines and assesses the effectiveness, across an organization, of the application of software design strategies and methods. (M)
				2. Selects the appropriate design methodology (such as object-oriented, function-oriented, component-based) and strategies to be used at the project level. (L)	2. Analyzes and makes recommendations related to organization-wide software design strategies and methodologies. (M)

Table B12					
Software Design Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Design Strategies and Methods				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)	3. Creates new techniques evaluating software design quality. (M)
			2. Determines design alternatives and performs trade-off analysis. (P/L)	5. Determines design alternatives and performs trade-off analysis. (L)	

Table B12

Software Design Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Architectural Design	1. Provides assistance in the installation and use of software architecture tools. (F/A)	1. Assists in architectural design tasks associated with use of standard notations, diagramming techniques, models, and patterns. (A)	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)	1. Provides direction and advice on standard notations, diagramming techniques, models, and patterns to be applied. (L)	1. Analyzes and makes recommendations related to organization-wide software architectural design. (M)
		2. Applies a selected software design pattern to the design of a software component or module. (A/P)	2. Creates multiple views of the software system. (P/L)	2. Evaluates the effectiveness of the creation of software architecture. (P/L)	2. Determines new methods and techniques to be used in architectural design. (M)

Table B12					
Software Design Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Architectural Design			3. Uses design patterns and frameworks to design mid-level software components or modules. (P)		
Software Design Quality Analysis and Evaluation	1. Assists software designers with tools and techniques for collecting design metrics and evaluating software design quality. (F/A)	1. Participates in software design reviews. (P)	1. Facilitates software design reviews. (P/L)	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)	1. Analyzes and makes recommendations related to organization-wide design quality evaluation and analysis techniques. (M)
		2. Carries out static analysis tasks to evaluate design quality. (P)	2. Leads static analysis tasks to evaluate design quality. (P/L)		

Table B12

Software Design Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Design Quality Analysis and Evaluation		3. Assists in development and use of simulation and prototypes to evaluate software design quality. (A)	3. Develops and uses simulation and prototypes to evaluate software design quality. (P)		
				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)	2. Creates new techniques for evaluating software design quality. (M)
				3. Provides guidance and direction related to the need for requirements change resulting from design review. (P/L)	

13. SOFTWARE CONSTRUCTION SKILL AREA

Software construction is the collection of activities and processes for converting design specifications into functional software solutions that meet customer needs. It includes planning, designing, programming, debugging, testing, and integrating software components. Most software construction is performed by teams of professionals using tools (editors, compilers, version control software, debuggers, and so forth) and processes to accomplish and coordinate their work.

REFERENCES

[ACM 2004] ACM/IEEE-CS Joint Task Force on Computing Curricula, *Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, August 2004; www.acm.org/education/curricula.html.

[Fowler 1999] M. Fowler et al., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

[Hunt 1999] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley, 1999.

[McConnell 2004] S. McConnell, *Code Complete*, 2nd ed., Microsoft Press, 2004.

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

Table A13

Software Construction Skill Sets	Software Construction Activities
Software Construction Planning	• Select appropriate processes and models for constructing software, including appropriate reuse processes. • Select appropriate languages and tools for software construction. • Select appropriate frameworks, platforms, and environments.
Managing Software Construction	• Establish and follow project standards for version control and configuration management. • Collect and monitor standard measures of code quality and size.
Detailed Design and Coding	• Create detailed designs that minimize complexity and enhance quality. • Create code to implement detailed designs. • Refactor code when needed. • Establish and follow standards for designs and code. • Use appropriate design patterns. • Use defensive coding techniques to minimize propagation of errors and threats. • Document code through comments to support software maintenance. • Generate code from design models.
Debugging and Testing	• Use appropriate tools and techniques for debugging. • Create and execute unit tests for all delivered code. • Achieve test coverage goals.
Integrating and Collaborating	• Establish and follow integration strategy and processes. • Perform integration testing as part of the integration process. • Collaborate with other team members in development activities (such as pair programming, informal reviews). • Participate or lead reviews and inspections.

The following notations are used in Table B13: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B13

Software Construction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Construction Planning			1. Assists in the selection of appropriate processes and models for software construction. (F/P)	1. Selects appropriate processes and models for constructing software on individual projects (such as compilation, generation from domain-specific languages). (L)	1. Creates or proposes new processes and models for software construction. (C)
			2. Assists in the selection of appropriate languages and tools for software construction. (F/P)		2. Creates new languages and tools for software construction. (C)

Table B13					
Software Construction Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Construction Planning			3. Assists in the selection of appropriate frameworks, platforms, and environments for software construction. (F/P)	2. Selects appropriate languages and tools for software construction on individual projects. (L)	3. Creates or proposes new frameworks, platforms, and environments. (C)
				3. Selects appropriate frameworks, platforms, and environments for individual projects. (L)	
Managing Software Construction	1. Assists in the installation of tools and repositories for version control and configuration management. (A)	1. Uses standard tools and processes for version control and configuration management. (P)	1. Monitors standard measures of code quality and size. (P)	1. Establishes project standards for version control and configuration management. (L)	1. Establishes organization standards for version control and configuration management. (L)
		2. Collects standard measures of code quality and size. (P)			2. Sets organization standards for code quality and size. (L)

Table B13

Software Construction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Managing Software Construction					3. Creates new tools and processes for version control and configuration management. (C)
Detailed Design and Coding	1. Assists in the installation of tools and repositories for design and coding. (A)	1. Creates code to implement detailed designs. (P)	1. Creates and reviews detailed designs and code that meet quality requirements. (P)	1. Measures and monitors an organization's use of design patterns. (L)	1. Establishes organization standards for detailed designs and code. (L)
		2. Refactors code when needed. (P)	2. Suggests and performs appropriate code refactoring when needed. (L)		2. Creates new design patterns. (C)
		3. Follows project and organization standards for designs and code. (F)	3. Selects or establishes project standards for designs and code. (P)		
		4. Uses appropriate design patterns. (P)	4. Suggests and uses appropriate design patterns. (L)		

Table B13					
Software Construction Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Detailed Design and Coding		5. Uses defensive coding techniques to minimize propagation of errors and threats. (P)	5. Suggests and uses defensive coding techniques to minimize propagation of errors and threats. (L)		
		6. Documents code through comments to support software maintenance. (P)			
		7. Generates code and systems from models (such as UML) as appropriate. (P)	6. Writes executable models suitable for code generation as appropriate. (P/L)	2. Plans and initiates model-driven development processes as appropriate. (L)	

Table B13

Software Construction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Debugging and Testing	1. Assists in the installation of tools for debugging and testing. (A)	1. Uses appropriate tools and techniques for debugging. (P)	1. Ensures project standards for unit test coverage are followed. (P)	1. Establishes project standards for unit test coverage. (L)	1. Establishes organization standards for unit testing. (L)
		2. Creates and executes unit tests for all delivered code. (P)		2. Selects appropriate debugging tools and techniques for a project. (L)	2. Creates new unit testing tools and methods. (C)
		3. Achieves test coverage goals set by project and organization standards. (P)			
Integrating and Collaborating	1. Assists in installation of integration tools. (A)	1. Follows project integration strategy and processes. (P)	1. Leads code reviews and inspections. (L)	1. Assists in selection of project tools and processes for integration. (P)	1. Establishes organization standards for integration tools and processes. (L)
	2. Assists in creation of code inspection packages. (A)	2. Performs integration testing as part of the integration process. (P)			2. Establishes organization standards for reviews and inspections. (L)

Table B13					
Software Construction Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Integrating and Collaborating	3. Assists in scheduling code inspections. (A)	3. Collaborates with other team members in development activities (such as pair programming, informal reviews). (P)			3. Creates new integration tools and processes. (C)
		4. Participates in project-defined reviews and inspections. (P)			4. Creates new code review and inspection methods. (C)
	4. Sets up build-and-install environments where the software packages can be integrated. (P)				

14. SOFTWARE TESTING SKILL AREA

Software testing is a component of overall software quality; however, the software quality skill area in SWEBOK is a crosscutting skill area, whereas the software testing skill area is a life cycle skill area. Software testing includes all activities that are performed to evaluate overall product quality, which requires code execution. This software testing skill area covers "dynamic verification" and is concerned with selecting an appropriate set of test cases that demonstrate the expected behavior of the product by executing the software using prepared test cases and their results to analyze and improve the quality of the product. It is well known that software cannot be tested exhaustively for all possible situations; therefore, "selecting an appropriate set of test cases" has a major effect on the success or failure of testing activities.

REFERENCES

[IEEE 730-2002] *IEEE Std. 730-2002, IEEE Standard for Software Quality Assurance Plans*, IEEE, 2002.

[IEEE 829-1998] *IEEE Std. 829-1998, IEEE Standard for Software Test Documentation*, IEEE, 1998.

[IEEE 1012-2012] *IEEE Std. 1012-2012, IEEE Standard for System and Software Verification and Validation*, IEEE, 2012.

[Myers 2011] Glenford J. Myers, *The Art of Software Testing*, 3rd ed., Wiley, 2011.

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

Table A14

Software Testing Skill Sets	Software Testing Activities
Software Test Planning	• Identify all stakeholders involved in software testing. • Identify success and failure criteria. • Identify test completion criteria. • Design and implement the software test plan. • Identify and coordinate customer representatives and other stakeholders participating in the software acceptance and/or demonstration.
Software Testing Infrastructure	• Identify tools to be used throughout testing activities. • Identify appropriate documentation to be generated and archived. • Design/select and implement the test environment.
Software Testing Techniques	• Identify test objectives. • Select appropriate testing/demonstration techniques. • Design, implement, and execute test cases.
Software Testing Measurement and Defect Tracking	• Identify, collect, and store appropriate data resulting from testing/demonstration. • Report test results to appropriate stakeholders. • Identify, assign, and perform necessary corrective actions. • Analyze test data for test coverage, test effectiveness, and process improvement.

NOTE: Throughout the industry, a large number of testing techniques have been and are used. Whenever testing techniques are discussed in this skill area, it could potentially refer to any of those techniques; in such cases, "testing" is in *italics*. In a few instances, "testing" refers to a specific technique and is not in italics.

The following notations are used in Table B14: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B14

Software Testing Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Test Planning		1. Identifies unit and integration testing success and failure criteria. (P)	1. Identifies stakeholders participating in demonstration and testing activities. (P)	1. Establishes organizational procedures for testing. (P)	1. Establishes organizational procedures for *testing*. (L)
		2. Follows software test plan. (P)	2. Designs and implements a software test plan. (P)	2. Identifies customer representatives and other stake holders participating in the acceptance testing and demonstrations. (L)	

Table B14					
Software Testing Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Test Planning		3. Establishes the criteria for unit test execution completion, such as code coverage, defect intensity, and so forth. (P)	3. Identifies success and failure criteria for *testing*. (L/P)	3. Identifies project test objectives. (L)	
		4. Develops unit test plan. (P)	4. Develops demonstration or other *test* plans. (P)	4. Identifies success and failure criteria for system and acceptance testing. (P)	
		5. Assists with the development of the *test* plan.	5. Establishes criteria for demonstration readiness. (A/P)	5. Identifies *test* completion criteria. (P)	
			6. Selects the most appropriate demonstration, testing technique. (P)		

Table B14

Software Testing Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Test Planning			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)		
Software Testing Infrastructure	1. Sets up the necessary *test* and demonstration environment. (P)	1. Selects appropriate unit test techniques.	1. Defines the necessary setup for *testing* and demonstration. (P/L)	1. Identifies testing tools and test data warehousing across projects. (P)	1. Identifies organizational *testing* tools and data warehousing across projects. (L)
	2. Installs the necessary tools. (P)	2. Selects the most appropriate *testing* tools. (P)	2. Identifies the appropriate documentations necessary for the *testing* activities. (P)		2. Creates new test documentation (in other words, test plan, defect recording, and so forth). (L/P)

Table B14

Software Testing Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Testing Infrastructure	3. Develops the appropriate infrastructure for data warehousing. (P)	3. Designs and implements the *test* environment. (P)	3. Designs the *test* environment. (L/P)		
Software Testing Techniques	1. Performs manual *test* activities (in other words, data entry, test case execution, and so forth). (P)	1. Designs and executes unit test cases. (P)	1. Specifies appropriate test cases for the selected testing technique. (L/P)	1. Designs system test plan and test cases. (L)	1. Creates new *testing* (in other words, unit, integration, stress, and so forth) techniques. (C)
	2. Executes regression testing. (P)	2. Assists with the development of the *test* cases. (P)		2. Identifies automated *testing* opportunities. (L/P)	
	3. During demonstration, monitors customer use and records customer feedback for product improvement. (A)	3. Executes integration and system test cases. (P)			

Table B14

Software Testing Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Testing Techniques		4. Ensures the system is ready for demonstration by performing acceptance test. (P/L)	2. Executes *test* cases. (L/P)		
			3. Develops automated *test* case scenarios. (L/P)		
Software Testing Measurement and Defect Tracking	1. Performs all appropriate data warehousing (gathering execution data, data entry, data archiving, and so forth). (P)	1. Collects appropriate data associated with test execution. (P)	1. Collects appropriate data associated with *test* execution. (L/P)	1. Conducts root cause analysis. (L/P)	1. Creates new root cause analysis techniques. (P)
	2. Generates appropriate reports associated with test / demonstration execution. (P)	2. Evaluates test execution results and identifies appropriate rework. (P)	2. Conducts root cause analysis. (P)	2. Analyzes test data to decide on further testing activities. (L)	

Table B14					
Software Testing Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Testing Measurement and Defect Tracking	3. Collects appropriate data associated with executing test cases. (P)	3. Monitors *test* progress by evaluating defect arrival rate, failure intensity, and so forth. (A/P)	3. Using the test results, assigns appropriate rework to team members. (P)	3. Uses the data to evaluate test effectiveness. (L)	
	4. Provides test result report to appropriate stakeholders. (P)		4. Uses test execution data and rework results to evaluate test effectiveness and decide for additional *testing* or regression testing. (P)	4. Evaluates test results to identify appropriate process improvement opportunities. (P)	
			5. Monitors *test* progress by evaluating defect arrival rate, failure intensity, and so forth. (P)	5. Monitors overall *test* progress by evaluating defect arrival rate, failure intensity, and so forth. (L)	

15. SOFTWARE SUSTAINMENT SKILL AREA

As systems become larger, more complex, and increasingly reliant on software, sustainment issues become increasingly complex and time consuming. A study of software systems effort distribution indicated that 55% of software system life cycle effort involves sustainment activities [CMU/SEI-2006-TN-007]. The risks of inadequate sustainment can potentially undermine the stability, enhancement, and longevity of operational systems.

No authoritative definition of "software sustainment" exists. The Software Engineering Institute's working definition is, "The processes, procedures, people, material, and information required to support, maintain, and operate the software aspects of a system" [Lapham 2006].

The IEEE Standard Glossary of Software Engineering Terminology defines software maintenance as "The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment" [IEEE/ISO/IEC 24765-2010].

According to these definitions, software sustainment addresses issues of maintenance plus documentation, deployment, operation, security, configuration management, training (users and sustainment personnel), help desks, COTS product management, technology updates, and software retirement.

Three categories of software maintenance activities—corrective, adaptive, and perfective—involve modifying a software product/system after delivery. A fourth category of software maintenance activities focuses on preventive maintenance.

Successful software sustainment depends on the culture of the sustainment organization, the skills of the sustainment team (which is the focus of this skill area), the flexibility of the customer, and the operational domain of the product, in addition to skills needed to modify source code for corrective, adaptive, and perfective enhancements.

REFERENCES

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

[IEEE/ISO/IEC 24765-2010] *IEEE/ISO/IEC 24765:2010 Systems and Software Engineering—Vocabulary*, IEEE, 2010.

[Lapham 2006] M. Lapham and C. Woody, *Sustaining Software-Intensive Systems*, CMU/SEI-2006-TN-007, Software Engineering Institute, 2006; http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=7865.

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

Table A15	
Software Sustainment Skill Sets	**Software Sustainment Activities**
Software Transition	• Develops transition plan. • Identifies stakeholders for transition and operational requirements. • Identifies system and software constraints. • Identifies applicable systems and software operational standards (such as information assurance). • Develops software transition and operational documentation. • Installs software. • Performs software operational training (for both users and sustainment personnel). • Determines the impacts on the operational environment. • Develops software system activation and check-out procedures. • Participates in system acceptance.
Software Support	• Maintains current software configurations. • Performs operational software assurance. • Updates COTS and other software technologies to maintain currency. • Diagnoses and responds to reported software defects, anomalies, and operational incidents and events. • Monitors system operation and collects operational data. • Develops and implements software retirement procedures.
Software Maintenance	• Establishes software maintenance processes and plans. • Obtains and maintains baseline software artifacts. • Performs problem identification and technical impact analysis. • Makes and assures changes to software (corrective, adaptive, perfective). • Performs preventative maintenance and software re-engineering. • Monitors and analyzes software maintenance activities.

The following notations are used in Table B15: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B15

Software Sustainment Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Transition			1. Participates in developing transition plans. (P)	1. Leads development of transition plans. (L)	1. Modifies existing and creates new guidelines for transition plans. (C)
			2. Participates in the identification of stakeholders for transition and operational requirements. (P)	2. Leads in the identification of stakeholders for transition and operational requirements. (L)	2. Modifies existing and creates new guidelines for the identification of stakeholders. (C)
			3. Participates in the identification of system and software constraints. (P)	3. Leads in the identification of system and software constraints. (L)	
			4. Identifies applicable systems and software operational standards. (P)	4. Modifies existing and develops new systems and software operational standards. (L)	

Table B15

Software Sustainment Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Transition		1. Assists in identifying applicable systems and software operational standards. (A)	5. Leads in the development of software transition and operational documentation. (L)	5. Approves software transition and operational documentation. (L)	
	1. Follows instructions to help develop software transition and operational documentation. (F)	2. Assists in the development of software transition and operational documentation. (A) Installs software. (P)	6. Leads in the installation of software. (L)	6. Approves new software installations. (L)	3. Modifies existing and creates new templates for software transition and operational documentation. (C)
	2. Follows instructions to help install software. (F)	3. Assists in the development of training material for operational support personnel. (A)	7. Develops training material for operational support personnel. (P)	7. Approves training material for operational support personnel. (L)	
	3. Assists in training operational support personnel. (A)	4. Trains software operational support personnel. (P)	8. Leads software operational training. (L)		

Table B15					
Software Sustainment Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Transition		5. Performs software system activation and check-out procedures. (P)	9. Develops software system activation and check-out procedures. (P)	8. Approves software system activation and check-out procedures. (L)	
		6. Assists in determining the impacts of software changes on the operational environment. (A)	10. Participates in determining the impacts of software changes on the operational environment. (P)	9. Leads in determining the impacts of software changes on the operational environment. (L)	4. Modifies existing and creates new guidelines for determining the impacts of software changes on the operational environment. (C)
		7. Assists in system acceptance. (A)	11. Participates in system acceptance. (P)	10. Leads system acceptance. (L)	5. Modifies existing and develops new system acceptance methods, tools, and techniques. (C)

Table B15

Software Sustainment Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Support	1. Operates operational software configuration management tools. (A)	1. Performs operational software configuration management. (P)	1. Develops operational software configuration management plans. (L)	1. Approves operational software configuration management plans. (L)	1. Modifies existing and creates new standards and frameworks for operational software configuration management. (C)
	2. Follows instructions to perform operational software assurance tasks. (F)	2. Performs operational software assurance. (P)	2. Leads operational software assurance activities. (L)	2. Develops software assurance plans. (L)	
	3. Installs COTS and other software updates. (P)	3. Updates COTS and other software technologies to maintain currency. (P)	3. Leads maintenance of COTS and other software technologies to maintain currency. (L)		
	4. Diagnoses and responds to reported software defects, anomalies, and operational incidents and events. (P)	4. Leads software help desk activities. (L)	4. Develops software help desk plans. (L)	3. Creates policies that cover help desk operations. (C)	

Table B15

Software Sustainment Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Support	5. Operates tools to collect operational data under supervision. (F)	5. Analyzes operational data. (P)	5. Acquires tools and supervises analysis of operational data. (L)	4. Develops plans for collecting and processing operational data. (L)	2. Modifies existing and creates new methods, tools, and techniques for collecting and processing operational data. (C)
		6. Assists in implementing software retirement procedures. (A)	6. Implements software retirement procedures. (P)	5. Develops software retirement plans. (L)	
Software Maintenance		1. Assists in implementing software maintenance processes and plans. (A)	1. Implements software maintenance processes and plans. (P)	1. Leads development of software maintenance processes and plans. (L)	1. Modifies existing and creates new software maintenance policies, processes, and procedures. (C)
	1. Participates in obtaining baseline software artifacts. (P)	2. Obtains and maintains software baseline artifacts. (P)	2. Identifies software baseline artifacts. (L)		

Table B15

Software Sustainment Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Maintenance		3. Performs problem identification. (P)	3. Performs change impact analysis. (P)	2. Leads problem identification and technical impact analysis. (P)	
		4. Assists in making changes to software (corrective, adaptive, perfective). (A)	4. Implements plans and makes changes to software (corrective, adaptive, perfective). (P)	3. Leads development of plans and supervises making changes to software (corrective, adaptive, perfective). (L)	2. Modifies existing and creates new policies and procedures for making changes to software (corrective, adaptive, perfective). (C)
	2. Assists in performing preventative maintenance and software re-engineering activities. (A)	5. Performs preventative maintenance and software re-engineering activities. (P)	5. Leads preventative maintenance and software re-engineering activities. (L)	4. Plans for and supervises preventative maintenance and software re-engineering activities. (L)	3. Modifies existing and creates new policies and procedures for preventative maintenance and software re-engineering. (C)
		6. Assists in monitoring and analyzing software maintenance activities. (A)	6. Monitors and analyzes software maintenance activities. (P)	5. Leads monitoring and analysis of software maintenance activities. (L)	

16. SOFTWARE PROCESS AND LIFE CYCLE SKILL AREA

Software process models and life cycle models skills are concerned with process definition and tailoring, implementation, workflow, assessment, measurement, management, and improvement of the software life cycle processes, including processes both for guiding a specific set of activities and for postmortem. The skills apply to a range of software process paradigms that range from plan-driven processes (sometimes called predictive processes) to agile processes (sometimes called adaptive processes) and the spectrum of processes between the two.

A key element of software processes is measurement and assessment of the effectiveness of individual activities and their combinations for software projects. The key purpose of process assessment is to identify process activities and combinations that need to be modified to better achieve project goals.

Software process model and life cycle model skills apply across most of the other skill areas in this document, including both life cycle and crosscutting skill areas.

REFERENCES

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes*, IEEE, 2008.

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

Table A16

Software Process and Life Cycle Skill Sets	Software Process and Life Cycle Activities
Software Development Life Cycle Implementation	• Determine one or more organization-wide life cycle models (such as waterfall, spiral, V-model, incremental, maturity models). • Select a team software process (plan-driven, adaptive). • Lead a small team in execution of some portion of a life cycle process model (such as software design). • Carry out process activities specified in a life cycle process model script.
Process Definition and Tailoring	• Define software processes for a project team or for a software engineering activity (such as requirements engineering). • Tailor a defined software process to the needs of a project team or software engineering activity. • Interpret and adapt a software process to individual process responsibilities and tasks. • Lead definition and tailoring of organization-wide software processes.
Process Implementation and Management	• Implement and execute software processes. • Provide guidance and advice to software teams on how to implement and manage software processes. • Serve as a member of a software engineering process group.
Process Assessment and Improvement	• Collect data for assessment of a software process. • Analyze process assessment data and implement improvement of team software processes. • Use assessment information and reports for software process improvement.

The following notations are used in Table B16: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B16

Software Process and Life Cycle Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Software Development Life Cycle Implementation	1. Provides assistance in the installation and use of tools appropriate for a project's designated life cycle model. (F/A)	1. Carries out process activities specified in a life cycle process model script. (A/P)	1. Leads a small team in execution of some portion of a life cycle process model (such as software design). (P/L)	1. Selects a life cycle model process for a software team. (L)	1. Determines the need for and selects or develops an organization-wide life cycle model. (C)
				2. Assists in selection of a department or organizational-wide life cycle process model. (P/L)	2. Selects department- or organization-wide process models. (C)
					3. Advises on process infrastructure, training, and tools. (C)

Table B16

Software Process and Life Cycle Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Process Definition and Tailoring	1. Provides assistance in the installation and use of tools for defining and modifying software processes. (F/A)	1. Interprets and adapts a software process to individual process responsibilities and tasks. (A/P)	1. Provides review of defined and tailored processes. (A/P)	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)	1. Conducts research into effective software process definition and tailoring. (C)
			2. Tailors a software process to the work of a small team. (P/L)	2. Provides guidance to others involved in tailored processes (individual and team). (L)	2. Leads definition and tailoring of organization-wide software processes. (L/C)

Table B16

Software Process and Life Cycle Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Process Implementation and Management	1. Provides assistance in the installation and use of tools for implementing, managing, and measuring software processes. (F/A)	1. Collaborates in the execution of a team software process.	1. Leads small teams in the implementation and execution of a software process. (P/L)	1. Leads large teams or multi-team projects in the implementation and execution of a software process. (L)	1. Provides organization-wide guidance and advice on how to implement and manage software processes. (C)
		2. Implements and manages individual processes. (F/A)	2. Provides guidance and advice to individuals on the implementation and management of their personal processes. (P/L)	2. Provides guidance and advice to software teams on how to implement and manage software processes. (L/C)	
			3. Serves as a member of a software engineering process group (SEPG). (P)	3. Serves as leader of an SEPG. (L)	2. Provides guidance and advice on the formation, structure, and responsibilities of SEPGs. (C)

Table B16

Software Process and Life Cycle Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Process Assessment and Improvement	1. Provides assistance in the installation and use of tools for assessing and improving software processes. (F/A)	1. Assists in collecting data for assessment of a software process. (A)	1. Leads small teams in collecting data for assessment of software processes. (P/L)	1. Leads software teams in collecting data for assessment of software processes. (P/L)	1. Conducts research into the effectiveness and improvement of software processes. (C)
		2. Collects data relevant to individual process execution. (F/A)	2. Analyzes process assessment data and implements improvement of small team software processes. (P/L)	2. Analyzes process assessment data and implements improvement of team software processes. (P/L)	
		3. Assesses and implements improvement of an individual software process. (A/P)	3. As a member of an SEPG, provides input on software process improvement. (A/P)	3. Leads the SEPG in providing guidance on department- or organization-wide software process improvement. (L/C)	2. Uses assessment data, team reports, and SEPG reports to establish organization procedures and standards for software process improvement. (C)

17. SOFTWARE SYSTEMS ENGINEERING SKILL AREA

Systems are collections of interconnected components that interact with other systems and the environment in which they are embedded. They include natural systems, such as our solar system, and human-made systems, such as digital computers. Modern engineered systems range from household appliances to medical devices, automobiles, spacecraft, and nuclear reactors. Engineered systems are increasingly reliant on software to provide functionality, behavior, interconnections among components, and external interfaces to environments that may be complex and ill defined.

Software engineers are often members, and may be leaders, of teams that develop and modify large and/or complex systems composed of diverse kinds of components, including software. Not all software engineers will be, or need to be, competent software systems engineers; however, those software engineers who participate as members of systems engineering teams for software-intensive systems should have the skills required to participate in systems engineering projects. The skills and activities in this skill area apply to development of systems for which software is a critical component but for which software may not be the primary cost item or the key technology to be exploited.

Not all software-intensive systems are engineered systems in the traditional sense. For example, an enterprise information system (EIS) may be dependent on software but the time, effort, and cost of procuring and installing hardware and infrastructure software may far exceed the time, effort, and cost of developing business-specific

software within the EIS environment. Systems engineering skills will be needed in these cases.

Table A provides a list of skills and work activities for a software engineer to competently participate in as a member of a systems engineering team that develops or modifies a large and/or complex software-intensive system.

Table A does not include all of the skills and activities that are important for a software engineer who participates in systems engineering of software-intensive systems but rather highlights those that are especially important at the systems level. For example, requirements elicitation is not included because the necessary skills are similar at both the systems and software levels. The difference in requirement elicitation, and in many other skill areas, is in the scope of considerations that must be addressed and the kinds and numbers of stakeholders that are typically involved at the systems level. System-level stakeholders usually include different kinds of engineers who have different technical specialties and different ways of thinking about systems that include electrical, mechanical, civil, and other kinds of components.

Another important consideration is that software engineers who participate in systems engineering activities at a given competency level should have the same or higher level of competency in the corresponding software engineering activity. For example, a software engineer who participates as a Practitioner in systems requirement elicitation should be a competent software engineering Practitioner for the activity of software requirements elicitation. Similarly, a Team Leader for requirements engineering at the systems level should be a competent Team Leader for requirements engineering at the software level.

Although the activities in Table A are listed in a sequential order, they are often accomplished in iterative and incremental steps. Table B includes the activities, at five different competency levels, for each skill in this skill area.

REFERENCES

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes*, IEEE, 2008.

[SEBoK 2013] A. Pyster and D.H. Olwell, eds., *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, The Trustees of the Stevens Institute of Technology, v. 1.2, 2013; http://sebokwiki.org.

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

Table A17

Software Systems Engineering Skill Sets	Software Systems Engineering Activities
System Development Life Cycle Modeling	• Select and integrate a software engineering development model into a systems engineering development model. • Tailor policies, procedures, and templates and select applicable standards.
Concept Definition	• Identify system stakeholders. • Develop the operational concepts (system context, operational environment(s), prioritized features, quality attributes, operational scenarios, assumptions, dependencies, limitations, and exclusions).

Table A17	
Software Systems Engineering Skill Sets	**Software Systems Engineering Activities**
System Requirements Engineering	• Establish the system development environment and identify technology constraints. • Identify system-level traceability requirements and tools. • Identify system requirements. • Develop the system requirements specification. • Develop plans, procedures, and scenarios for system integration, verification, validation, and deployment.
System Design	• Develop alternative solution concepts. • Identify system components and specify relationships and interfaces among components. • Conduct trade studies to identify major system components for hardware/software/ manual operations. • Participate in making acquisition decisions for system components. • Influence system design to avoid isolated stovepipe units of software.
Requirements Allocation	• Allocate requirements and interfaces to system components (functional, behavioral, structural, quality) and interfaces between software components and other major system components. • Develop bi-directional traceability between system requirements and software requirements. • Analyze, clarify, and refine requirements allocated to software.
Component Engineering	• Determine needed kinds of software components (database, algorithms, internet protocols). • Make acquisition decisions for software components (buy, build, open source, and so forth). • Work with software engineers who develop and integrate software components. • Provide liaison from software engineering to systems engineering and other major component engineering.

Table A17	
Software Systems Engineering Skill Sets	**Software Systems Engineering Activities**
System Integration and Verification	• Integrate software with other system components. • Participate in system verification activities. • Provide liaison to software component engineers.
System Validation and Deployment	• Participate in simulated and live system tests. • Participate in system acceptance testing. • Provide liaison to software component engineers.
System Sustainment Planning	• Participate in planning for system sustainment. • Prepare for operational support. • Provide liaison and participate in planning for software sustainment.

The following notations are used in Table B17: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B17					
Software Systems Engineering Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
System Development Life Cycle Modeling	1. Uses tools and follows directions to prepared depictions and documentation of tailored development models. (F)	1. Assists in integrating the selected software development model into the systems development model. (A)	1. Participates in integrating the selected software development model into the system development model. (P)	1. Participates in selection of the system development life cycle model. (P)	1. Prepares frameworks for integrating an organization's system and software development models. (C)
				2. Leads selection of the software development life cycle model. (L)	

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Development Life Cycle Modeling				3. Leads integration of the software development model into the system development model. (L)	
		2. Participates in systems engineering meetings and prepares meeting minutes to include decisions made, open issues, other relevant discussions, and action items. (A)		4. Participates in/leads tailoring of policies, procedures, and templates, and selection of applicable standards for projects and programs. (P/L)	2. Modifies existing models and creates new models and ways of integrated software engineering and system engineering development models. (C)

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Development Life Cycle Modeling					3. Determines applicable policies, procedures, standards, and guidelines and tailors them for an organization's system and software development models. (C)
Concept Definition	1. Assists by locating identified stakeholders. (A)	1. Identifies potential stakeholders by examining historical data and having discussions with appropriate personnel inside and outside the systems engineering team. (A/P)	1. Develops lists of stakeholders and categorizes their likely interests. (P/L)	1. Prepares criteria for identifying stakeholders. (L)	1. Develops new techniques for identifying stakeholders. (C)

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Concept Definition	2. Arranges stakeholder meetings. (A)		2. Attends stakeholder meetings and solicits stakeholder needs, wants, and desires. (P)	2. Leads stakeholder meetings. (L)	
	3. Attends stakeholder meetings and takes meeting minutes. (A)	2. Attends stakeholder meetings to document stakeholder needs, wants, and desires. (P)			
	4. Follows directions to prepare elements of the Concept of Operations. (F)	3. Develops elements of the Concept of Operations. (A)	3. Leads development of scenarios, storyboards, prototypes, and use cases. (L)	3. Facilitates development of the Concept of Operations. (L)	2. Develops new methods, tools, and techniques for concept definition. (C)
				4. Obtains stakeholder consensus on the Concept of Operations. (L)	
				5. Develops acceptance criteria. (P)	

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Requirements Engineering	1. Attends meetings and documents the system development environment and technology constraints. (P)		1. Establishes the system development environment and identifies technology constraints. (P)	1. Establishes the system development environment and identifies technology constraints. (L)	1. Establishes organizational policies and procedures for system requirements engineering. (P)
	2. Procures and operates traceability tools to establish and maintain traceability. (P)	1. Provides training on traceability procedures and tools. (A)	2. Identifies system-level traceability requirements and tools. (P)	2. Identifies system-level traceability requirements and tools. (L)	2. Modifies existing and develops new methods, tools, and techniques for system requirements engineering. (C)
	3. Follows instructions to document the system requirement specification. (F)	2. Assists in development of the system requirements specification. (A)	3. Participates in development of the system requirements specification. (P)	3. Leads development of the system requirements specification. (L)	

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Requirements Engineering	4. Documents plans, procedures, and scenarios for system integration, verification, validation, and deployment. (A)	3. Assists in development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (A)	4. Participates in development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (P)	4. Leads the development of plans, procedures, and scenarios for system integration, verification, validation, and deployment. (L)	
System Design		1. Assists in developing alternative solution concepts and conducting trade studies to identify major system components. (A)	1. Participates in developing alternative solution concepts and conducting trade studies to identify major system components. (P)	1. Leads development of alternative solution concepts to identify major system components. (L)	1. Develops policies and procedures for system design in an organization. (C)
		2. Participates in making buy/build decisions for software components. (P)	2. Participates in identifying system components as well as the interfaces and relationships among components. (P)	2. Leads/ participates in making buy/build decisions for major system components. (L/P)	

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Design		3. Assists in selecting components to be procured. (A)	3. Recommends buy/build decisions for software components. (P)	3. Approves buy/build decisions for software. (L)	
	1. Identifies sources of software components to be procured. (A)		4. Procures selected software components. (L)		
			5. Participates in system design meetings to avoid isolated stovepipe units of software. (P)	4. Leads/participates in system design meetings to avoid isolated stovepipe units of software. (P/L)	2. Develops new approaches to system design to avoid isolated stovepipe units of software. (C)

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements Allocation	1. Documents allocable and non-allocable requirements. (F)	1. Assists in identifying allocable and non-allocable requirements. (A)	1. Identifies allocable and non-allocable requirements. (P)	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)	1. Develops new methods, tools, and techniques for requirements allocation and flowdown. (C)
	2. Documents allocation of requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (F)	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)	2. Participates in meetings to allocate requirements (functional, behavioral, structural, quality) and interfaces to software components and other major system components. (P)		

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements Allocation			3. Participates in meetings to refine requirements allocated to software. (P)	3. Leads meetings to refine requirements allocated to software. (L)	
	3. Operates traceability tools and generates traceability reports. (F)	3. Develops bi-directional traceability between system requirements and software requirements. (P)	4. Leads traceability from system requirements to software requirements. (L)		
	4. Assists in clarifying and refining requirements allocated to software. (A)	4. Participates in clarifying and refining requirements allocated to software. (P)	5. Leads in clarifying and refining requirements allocated to software. (L)		
Component Engineering		1. Assists in determining and documenting needed kinds of software components. (A)	1. Determines needed software components. (P)	1. Leads in determining needed kinds of software components for a project or program. (L)	1. Modifies existing and develops new methods, tools, and techniques for component engineering. (C)

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Component Engineering	1. Documents buy/build decisions for software components. (A)	2. Assists in determining buy/build decisions for software components. (L)	2. Determines buy/build decisions for software components. (P)	2. Leads the buy/build decision-making process for software components. (L)	
	2. Maintains baselines of software components. (A)	3. Develops and integrates software components. (A/P)	3. Develops and integrates software components. (P/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)	
System Integration and Verification	1. Assists in integration of software with other system components. (A)	1. Assists in system verification activities. (A)	1. Participates in integration of software with other system components. (P)	1. Leads integration of software with other system components. (L)	1. Modifies existing and provides new methods of integrating software with other system components. (C)

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Integration and Verification			2. Participates in system verification activities. (P)	2. Leads/ participates in system verification activities. (L/P)	
		2. Assists in providing liaison to software component engineers. (A)	3. Provides liaison to software component engineers. (P)	3. Leads/ participates in providing liaison to software component engineers. (L)	
System Validation and Deployment	1. Operates tools for performing simulated and live system tests. (F)	1. Assists in performing simulated and live system tests. (A)	1. Participates in simulated and live system tests. (P)	1. Leads/ participates in simulated and live system tests. (P/L)	1. Modifies existing and develops new methods, tools, and techniques for system validation and deployment. (C)
	2. Operates tools for performing system acceptance testing. (F)	2. Assists in system acceptance testing. (A)	2. Participates in system acceptance testing. (P)	2. Establishes system acceptance criteria. (L)	

Table B17

Software Systems Engineering Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
System Validation and Deployment			3. Serves as liaison to software component engineers during system validation and deployment. (P)	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)	
System Sustainment Planning		1. Assists in planning for system sustainment. (A)	1. Participates in identifying stakeholders and developing a transition plan and requirements for operational support. (P)	1. Establishes criteria and procedures for system sustainment.	
			2. Prepares for operational support. (P)	2. Leads/ participates in planning for system sustainment. (L)	

18. SOFTWARE QUALITY SKILL AREA

The software quality skill area consists of fundamental skills that a software engineer needs to possess in order to produce a high-quality product, which is defined as one that conforms to its requirements and satisfies user needs. In SWECOM, the software quality skill area includes both software quality assurance and software quality control, which includes verification and validation. Measurement plays a major role in the areas of software quality assurance and control. In this skill area, we only discuss the data collection and data analysis portion of measurement as an activity in a specific skill topic. A complete treatment of measurement is covered in the software measurement skill area.

Software Quality Assurance (SQA) includes all competencies associated with ensuring a quality process. In other words, SQA within an organization establishes a series of processes, methods, standards, and techniques that are used throughout the organization to develop high-quality products. SQA is also responsible for implementing monitoring techniques to assure that established processes, methods, standards, and techniques are followed. Finally, SQA is responsible for implementing appropriate feedback loops that prevent defects from being introduced into the product. Collecting information and conducting root cause analyses continually improves the organization's ability to produce high-quality software products in the future.

NOTE: For the purposes of this document, its treatment of SQA is not comprehensive because some SQA processes are covered in other skill areas (for example, code source control in the

construction skill area) or as other skill areas (for example, configuration management).

Software Quality Control (SQC) includes all competencies associated with ensuring a quality product. In other words, an organization ensures that the software product will meet its quality goals through SQC. Software quality control may also be referred to as Software Validation and Verification; there are, however, some minor differences between the two terms.

Software Verification and Validation consists of fundamental competencies that a software engineer should possess in order to produce a high-quality product. Some of these competencies are needed throughout the software development life cycle (for example, reviews) and some are more specific to a specific phase of the project (for example, testing, which is covered in the software testing skill area).

Within software industries, some of the above terminologies have been used interchangeably. The purpose of this competency model is not to debate the proper use of these terminologies but to identify some of the major skills and corresponding activities that are required of software engineers in order to deliver high-quality products. Table A presents the skill areas and corresponding activities performed within those skill areas. Table B presents competency levels for the activities. It is important to note that, in this document, some of the activities associated with a specific skill may belong to either SQA or SQC.

REFERENCES

[IEEE 730-2002] *IEEE Std. 730-2002, IEEE Standard for Software Quality Assurance Plans*, IEEE, 2002.

[IEEE 829-2008] *IEEE Std. 829-2008, IEEE Standard for Software and System Test Documentation*, IEEE, 2008.

[IEEE 1012-2012] *IEEE Std. 1012-2012, IEEE Standard for System and Software Verification and Validation*, IEEE, 2012.

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes*, IEEE, 2008.

[Kan 2002] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd ed., Addison-Wesley, 2002.

[RTCA DO-178C] RTCA, Inc., *Software Considerations in Airborne Systems and Equipment Certification*, DO-178C/ED-12C, 13 Dec. 2011.

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

[Westfall 2009] Linda Westfall, *The Certified Software Quality Engineering Handbook*, Quality Press, 2009.

Table A18	
Software Quality Skill Sets	**Software Quality Activities**
Software Quality Management (SQM)	• Instill a culture of producing high-quality products. • Establish and follow quality goals and quality attributes. • Establish and follow a quality plan. • Identify, establish, follow, and verify appropriate processes, standards, and quality models that facilitate achieving quality goals and attributes. • Identify stakeholders that have authority and/or accountability for the process and product quality. • Identify and use appropriate tools and measurements needed to reach quality goals and attributes. • Establish and execute corrective actions if quality goals are not achieved. • Establish and execute appropriate continuous improvement processes. • Establish and update requirement traceability metrics and verification metrics.
Reviews (review, walkthrough, inspection)	• Plan, organize, and conduct appropriate review meetings. • Participate as a member of the review team. • Collect and analyze appropriate data resulting from the review. • Identify, assign, and perform necessary corrective actions.
Audits (concentrate on both product and process, but are done by an independent, internal or external, organization)	• Plan, organize, and conduct independent audits. • Collect appropriate data resulting from the audit. • Collect and analyze data collected from the audits. • Establish and implement appropriate resolutions for identified problems.
Statistical Control	• Identify and collect a set of quality data under statistical control. • Identify a set of subjective and objective variances for the data. • Analyze collected data. • Establish and implement a set of control processes. • Evaluate the effectiveness of the control processes.

The following notations are used in Table B18: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B18

Software Quality Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Quality Management (SQM)	1. Follows defined quality processes and standards. (P)	1. Follows quality standards for the product and supporting processes. (P)	1. Establishes quality standards for the product. (P/L)	1. Establishes a culture of producing quality products and of following quality processes across projects. (P/L)	1. Creates new and improved quality practices for delivering high-quality products. (C)
	2. Assists with establishing the appropriate infrastructure (such as defect tracking tools) to support organizations' quality goals. (A)	2. Follows defined quality models. (P)	2. Selects appropriate SQM processes that support the identified quality goals for the project. (P)		2. Creates new processes. (C)
		3. Uses appropriate tools and resources to develop quality products. (P)	3. Identifies quality characteristics and attributes for the product and establishes priorities. (P)	2. Establishes quality standards, models, and processes for projects. (P/L)	3. Examines and assesses the effectiveness of a specific SQM process across an organization. (C)

Table B18

Software Quality Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Quality Management (SQM)		4. Assists with identification of the different quality characteristics and attributes for the product. (A/P)	4. Identifies the quality models that need to be followed for the project. (P)	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. Makes recommendations related to organization-wide SQM processes. (C)
		5. Ensures that product-quality goals are achieved. (P)	5. Develops the Quality Assurance (QA) plan for the project. (P)	4. Develops the QA plan for the project. (L)	5. Creates/ modifies SQM processes to achieve higher-quality products and processes. (C)
		6. Collects quality metrics and prepares quality documentation to be shared with appropriate stakeholders. (P)	6. Identifies appropriate stakeholders who have authority and accountability regarding the quality process and quality product. (P)	5. Identifies organizational measures that support achieving product-quality goals (across projects). (P/L)	6. Proposes/ Designs new tools to improve the achievement of product-quality goals. (L)

Table B18

Software Quality Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Software Quality Management (SQM)		7. Develops and updates an appropriate traceability matrix for the product. (P)	7. Identifies appropriate measures that support achieving product-quality goals. (P/L)	6. Identifies continuous improvement opportunities across projects. (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)		

Table B18					
Software Quality Skill Sets and Activities by Competency Level					
	Levels				
Skill Sets	**Technician**	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Reviews	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)	1. Participates as an active member of the review team in order to achieve the goals of the activity. (P)	1. Identifies appropriate review processes needed to achieve product-quality goals. (P/L)	1. Identifies appropriate organization-wide review processes. (P/L)	1. Creates new or customizes review processes to meet organizational needs. (C)
		2. Uses appropriate checklists called for by the review organizer. (P)	2. Identifies appropriate personnel that need to participate in review activities. (P)	2. Conducts across-the-organization data analysis for the purpose of root cause analysis. (P)	2. Develops new root cause analysis techniques. (C)

Table B18

Software Quality Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Reviews		3. Collects appropriate and accurate data that is called for by the review organizer. (P)	3. Identifies appropriate measures that need to be collected as part of the product review. (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)	
		4. Produces appropriate documentation called for by the quality management plan. (P)	4. Identifies appropriate artifacts under the review and corresponding checklist. (P)		
		5. Follows appropriate practices defined by the quality management plan. (P)	5. Analyzes collected product data for the purpose of root cause analysis and assessment of review effectiveness. (P)		

Table B18					
Software Quality Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Reviews			6. Identifies appropriate corrective actions in order to achieve product improvement. (P)		
			7. Leads the review team. (P)		
Audits	1. Establishes the environment necessary to conduct the audit. (A/P)	1. Participates in audits. (P)	1. Plans, organizes, and conducts audits. (P/L)	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)	1. Creates new audit processes. (C)

Table B18

Software Quality Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Audits			2. Classifies issues identified by audits. (P)	2. Analyzes audit results for continuous improvement. (P)	
			3. Establishes and implements appropriate resolution strategies for identified issues. (P)		
Statistical Control	1. Establishes the environment necessary for data collection and warehousing. (P)	1. Collects a set of quality data under statistical control. (P)	1. Analyzes the collected data. (L)	1. Identifies a set of quality data under statistical control. (P)	1. Creates or modifies organizational statistical quality control gates. (C)
			2. Deploys a set of control processes. (P)	2. Identifies a set of subjective and objective variances for the data. (P)	
			3. Evaluates the effectiveness of the control processes. (A)	3. Analyzes the collected data. (L)	

Table B18					
Software Quality Skill Sets and Activities by Competency Level					
	Levels				
Skill Sets	**Technician**	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Statistical Control				4. Establishes and implements a set of control processes. (P)	
				5. Evaluates the effectiveness of the control processes. (P)	

19. SOFTWARE SECURITY SKILL AREA

Software security is a crosscutting skill area that affects the entire software development and operation life cycle. It includes techniques and processes to identify potential security vulnerabilities, avoid such vulnerabilities in design and implementation, and discover them in software artifacts. It includes mimicking an attacker and reviewing attack patterns. It also includes collecting and monitoring metrics to ensure that disciplined software development processes are followed. Testing is included in a separate skill area.

REFERENCES

[Allen 2008] Julia Allen et al., *Software Security Engineering: A Guide for Project Managers*, Addison-Wesley Professional, 2008.

[BITS 2012] *BITS Software Assurance Framework*, Financial Services Roundtable, 2012; www.bits.org/publications/security/BITSSoftwareAssurance0112.pdf.

[Hilburn 2013] Thomas Hilburn et al., *Software Assurance Competency Model*, Technical Note CMU/SEI-2013-TN-004, Software Engineering Institute, Mar. 2013; http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=47953.

[Merkow 2010] M. Merkow and L. Raghavan, *Secure and Resilient Software Development*, CRC Press, 2010.

[Seacord 2005] R. Seacord, *Secure Coding in C and C++*, Addison-Wesley, 2005.

Table A19

Software Security Skill Sets	Software Security Activities
Requirements	• Identify security risks (such as misuse cases). • Create requirements that capture security issues. • Perform initial threat modeling.
Design	• Model threats and associated risks of new and modified systems. • Identify the attack surface of new and modified systems. • Follow recommended design principles to create secure systems. • Use appropriate, secure design patterns.
Construction	• Follow recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). • Follow recommended coding standards to avoid security vulnerabilities.
Process	• Collect and monitor metrics for security assessment processes.
Quality	• Perform code reviews to identify security vulnerabilities. • Use static analysis methods to identify security vulnerabilities.

The following notations are used in Table B19: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B19

Software Security Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements			1. Identifies security risks (such as misuse cases). (P)		1. Creates or proposes new methods for recognizing security vulnerabilities. (C)
			2. Creates requirements that capture security issues. (P)		
			3. Performs initial threat modeling. (P)		

Table B19					
Software Security Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Design		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)	1. Models threats and associated risks of new and modified systems. (P)		
		2. Uses appropriate, secure design patterns. (F)	2. Identifies the attack surface (in other words, the areas of potential weakness exploited by attackers) of new and modified systems. (P)		

Table B19

Software Security Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Construction		1. Follows recommended secure coding principles to avoid security vulnerabilities (such as buffer overflow, input validation). (F)	1. Selects or establishes project coding standards to avoid security vulnerabilities. (P)	1. Establishes organization coding standards to avoid security vulnerabilities. (L)	1. Creates new coding standards to avoid security vulnerabilities. (C)
		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)	2. Reviews and approves coding standards to avoid security vulnerabilities. (P)		

Table B19					
Software Security Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Process	1. Assists in the collection of metrics for security assessment processes. (A)	1. Follows project standards in the collection of security assessment metrics. (F)		1. Establishes organization standards for security assessment processes. (L)	
Quality	1. Assists in the installation of static analysis tools. (A)	1. Performs code reviews to identify security vulnerabilities. (P)	1. Selects appropriate static analysis tools to identify security vulnerabilities. (P/L)		1. Creates new static analysis methods or tools. (C)
		2. Uses static analysis methods to identify security vulnerabilities. (P)			

20. SOFTWARE SAFETY SKILL AREA

The quality of safety-critical systems (systems in which human life, health, property, or the environment is at risk) is essential. Modern system safety is based on analysis of system functionality and identification of hazards, risks, and acceptance criteria. Such analysis and identification result in specific safety requirements that need to be considered in order to guide subsequent design and implementation.

The safety process must be comprehensive, with structured objectives that require rigorous engineering evidence to verify safety functionality, which must be deterministic and result in a system with acceptable risk for its intended operating environment. Development of safety-critical systems must address functional hazard analyses and include detailed specification, design, and implementation artifacts at all levels.

REFERENCES

[Bozzano 2010] Marco Bozzano and Adolfo Villafiorita, *Design and Safety Assessment of Critical Systems*, CRC Press, 2010.

[Hilburn 2013] Thomas Hilburn et al., *Software Assurance Competency Model*, Technical Note CMU/SEI-2013-TN-004, Software Engineering Institute, Mar. 2013; http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=47953.

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

[Leveson 2011] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, The MIT Press, 2011.

[Rierson 2013] Leanna Rierson, *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*, CRC Press, 2013

[Stephans 2004] R.A. Stephans, *System Safety for the 21st Century: The Updated and Revised Edition of System Safety 2000*, Wiley, 2004.

[Vincoli 2006] J.W. Vincoli, *Basic Guide to System Safety*, Wiley, 2006.

Table A20	
Software Safety Skill Sets	**Software Safety Activities**
Requirements	• Conduct formal system hazard analyses. • Identify safety requirements and verify their completeness. • Assure that safety requirements are correct and realizable.
Design	• Propose and select design solutions to assure the hazards are mitigated. • Analyze design risk from a safety perspective. • Verify completeness and correctness of the design from a safety perspective. • Ensure safety requirements are met.
Construction	• Select project coding standards to assure code safety. • Implement code components and their interfaces, considering safe coding practices to avoid safety violations. • Verify that the safety aspects of a design are implemented in the produced code.
Testing	• Perform testing to assure safety requirements are met. • Use industry guidelines and established organization standards for safety validation and verification.
Process	• Use established organization standards for safety assessment and selection of safety criteria. • Identify artifacts required to establish a safety case. • Use industry criteria to verify the completeness of the safety requirements.
Quality	• Collect data and report about the safety aspects of the product and process. • Analyze quality management (QM) data to assess and manage the overall project quality, with a focus on safety aspects.

The following notations are used in Table B20: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B20

Software Safety Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements	1. Assists in collecting data for the creation of a hazard list. (F/A)	1. Creates and verifies preliminary hazard lists. (A/P)	1. Using tools, conducts formal system hazard analyses verifying safety models. (P)	1. Verifies completeness and correctness of safety requirements. (/L)	
	2. Assists in the identification of top-level mishaps and their causes. (F/A)	2. Uses software tools to build safety models (FTA, ETA, FMEA). (A/P)			
	3. Assists with the installation of safety and reliability tools. (F/A)	3. Assists in safety requirements identification. (F/A)	2. Identifies safety requirements. (P)	2. Interacts with system and software engineers to assure that safety requirements are complete and realizable. (L)	

Table B20

Software Safety Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements			3. Assures that safety requirements are included in the overall system requirements. (P)		1. Creates or proposes new methods for safety assessment, mitigation, and verification. (C)
Design	1. Assists in identifying mitigation techniques for defined safety requirements. (F/A)	1. Implements design solutions to assure that the hazards are mitigated and the safety requirements are met. (A)	1. Proposes and selects design solutions to assure the hazards are mitigated. (P)	1. Verifies completeness and correctness of the design, including safety hazards and safety qualities. (P/L)	1. Creates or proposes new design solutions, leading to the increased safety of new designs. (C)
		2. Follows the recommended design principles. (P)	2. Supervises the design team. Analyzes risk and verifies design from a safety perspective. (P/L)	2. Leads the project in deciding the proposed architectural solutions to mitigate hazards. (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)	

Table B20					
Software Safety Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Construction		1. Implements large code components and their interfaces, considering safe coding practices to avoid safety violations. (A/P)	1. Implements the architecture and design to ensure code safety. (P / L)	1. Establishes organization standards to ensure code safety. (L/C)	1. Creates new standards to ensure code safety. (C)
			2. Manages the interfacing of large code components with special attention to potential safety issues. (P/L)	2. Oversees and verifies that the safety aspects of the design are implemented in the produced code. (L)	
Testing	1. Assists in the installation of tools and infrastructure for safety requirements testing. (F/A)	1. Performs testing using tools with a focus on safety requirements. (A/P)	1. Selects appropriate testing techniques to assure the safety of the application. (P)	1. Establishes organization standards for safety validation and verification. (L/C)	1. Contributes expertise to establish new organization guidelines related to testing the safety of software-intensive applications. (C)

Table B20					
Software Safety Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Testing			2. Applies applicable industry standards to assure that the product meets industry safety criteria. (P)	2. Manages the project, being responsible for overall safety and meeting industry guidelines. (L)	
Process	1. Assists in the collection of data to establish the project safety case. (F/A)	1. Identifies artifacts required to establish the safety case, following industry standards. (A/P)	1. Contributes to and verifies the completeness of the safety case, following selected industry criteria. (P)	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)	1. Contributes expertise to establish better means of assessing safety. (C)

Table B20					
Software Safety Skill Sets and Activities by Competency Level					
Skill Sets	**Levels**				
	Technician	**Entry Level**	**Practitioner**	**Technical Leader**	**Senior Software Engineer**
Quality	1. Assists in the collection of safety QM data. (A)	1. Collects safety QM data and reports the project status. (A/P)	1. Supervises collection of QM data and their compatibility with the safety case. (P/L)	1. Manages the overall project quality with a focus on safety aspects. (L)	1. Contributes expertise to improve means of measuring and establishing the safety quality of the product and process. (C)

21. SOFTWARE CONFIGURATION MANAGEMENT SKILL AREA

According to IEEE Standard 828-2012, configuration management is the discipline of applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, to control changes to those characteristics, to record and report change processing and implementation status, and to verify compliance with specified requirements [IEEE 828-2012]. According to the Software Configuration Management KA in the *SWEBOK Guide* [SWEBOK 2014], the elements of software configuration management include:

- Software configuration identification
- Software configuration control
- Software configuration status accounting
- Software configuration auditing
- Software release management and delivery

Skills for each of these elements and the associated activities are indicated in Table A21. The following acronyms are used in Table A21 and throughout this skill area:

- SCM: Software Configuration Management
- SCMP: Software Configuration Management Plan
- SCI: Software Configuration Item
- SDLC: Software Development Life Cycle

REFERENCES

[Aiello 2010] *Bob Aiello and Leslie Sach, Configuration Management Best Practices: Practical Methods that Work in the Real World*, Addison-Wesley Professional, 2010.

[Babich 1986] Wayne A. Babich, *Software Configuration Management: Coordination for Team Productivity*, Addison-Wesley, 1986.

[IEEE 828-2012] *IEEE Std. 828-2012, IEEE Standard for Configuration Management in Systems and Software Engineering*, IEEE, 2012.

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

Table A21	
Software Configuration Management Skill Sets	**Software Configuration Management Activities**
Plan SCM	• Determine organizational context for and constraints on SCM. • Identify software components to be controlled by SCM. • Design data and code repositories. • Plan versioning procedures for path branching and path integration. • Develop/adopt a change control process. • Identify and procure SCM tools. • Establish SCM library. • Develop SCMP.
Conduct SCM	• Follow SCMP. • Use SCM tools. • Control path branching and path integration during development. • Generate, classify, and manage problem reports. • Maintain and update SCM baselines. • Prepare SCM reports. • Conduct SCM audits.
Manage Software Releases	• Develop software release plan. • Identify and procure software release tools. • Use software release tools. • Produce software releases. • Design and implement tools and procedures for generating patches to be delivered.

The following notations are used in Table B21: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B21

Software Configuration Management Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Plan SCM	1. Operates SCM tools. (F)	1. Assists in determining impact of constraints on SCM imposed by policies, contract, and SDLC. (A)	1. Participates in determining impact of constraints on SCM imposed by policies, contract, and SDLC. (P)	1. Determines constraints and impacts of constraints on SCM imposed by policies, contracts, and SDLC. (L)	1. Develops new ways of organizing to perform SCM. (C)
	2. Operates and maintains the SCM library under technical leader supervision. (F)	2. Assists in developing, updating, and maintaining the SCMP. (A)	2. Develops and maintains the SCMP. (L)	2. Adopts an existing way of organizing for SCM and tailors a template for the SCMP. (L)	2. Develops new templates and ways of planning for SCM. (C)
		3. Provides measurement data for SCM measures. (P)	3. Assists in specifying the SCM measures to be used. (A)	3. Specifies the SCM measures to be used. (L)	3. Develops new measures and measurements for SCM. (C)

Table B21

Software Configuration Management Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Plan SCM		4. Assists in identifying software configuration items (SCIs). (A)	4. Participates in identifying SCIs and the relationships among them. (P)	4. Identifies SCIs and the relationships among them. (L)	4. Develops procedures for identifying SCIs and the relationships among them. (C)
		5. Assists in selecting and procuring SCM tools. (A)	5. Procures SCM tools. (P)	5. Specifies SCM tools. (L)	5. Specifies new SCM tools and ways of combining existing SCM tools. (C)
		6. Sets up an SCM library for a project under technical leader supervision. (L)		6. Specifies the template for, and supervises setting up, the SCM library. (L)	6. Specifies new ways of organizing SCM libraries. (C)

Table B21

Software Configuration Management Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Conduct SCM	1. Operates tools to generate SCM status and audit reports. (F)	1. Implements and documents approved changes to SCIs.	1. Evaluates and reports to CCB the impacts of proposed changes to SCIs. (P)	1. Tailors and adopts mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (P)	1. Revises existing and develops new mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (C)
	2. Generates, classifies, and manages problem reports. (F/A)	2. Generates, classifies, and manages problem reports. (P)	2. Generates, classifies, and manages problem reports. (L)	2. Appoints members and convenes the CCB. (L)	2. Develops new mechanisms for SCM status accounting. (C)
		3. Assists in using adopted mechanisms for requesting, evaluating, and approving software changes, including deviations and waivers. (A)	3. Uses established procedures for populating and maintaining the SCM library. (P)	3. Leads CCB in making yes/no decisions on change requests. (L)	3. Develops new processes and procedures for generating SCM audit reports. (C)

Table B21

Software Configuration Management Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Conduct SCM		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)	4. Uses established mechanisms to record and report SCM information. (P)	4. Ensures that approved changes are made and documented. (L)	
			5. Develops and tailors tools for generating SCM audit reports. (P)	5. Maintains mechanisms for recording and reporting SCM information. (L)	
				6. Establishes and maintains mechanisms for generating SCM audit reports. (L)	

Table B21

Software Configuration Management Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Manage Software Releases	1. Operates tools to build software releases. (A/P)	1. Participates in developing software release plans. (A/P)	1. Participates in developing software release plans. (P)	1. Develops software release plans. (L)	1. Modifies existing and develops new formats and procedures for implementing software release plans. (C)
	2. Uses software release tools to produce software releases. (A)	2. Participates in the building and verifying of software releases. (P)	2. Leads the building and verifying of software releases. (L)		2. Modifies existing and creates new tools for building software releases. (C)
	2. Modifies existing and creates new tools for building software releases. (C)	3. Participates in the building of software releases. (P)	3. Implements release plans. (P)		

22. SOFTWARE MEASUREMENT SKILL AREA

Measurement is foundational to the engineering disciplines, including software engineering. Measurements are used to quantify attributes of processes and products for the purposes of assessing progress and providing indications of real or impending problems. Measurement is a crosscutting skill area that applies to each of the other skill areas in this competency model. To be effective, measurement activities are planned prior to implementation and adjusted as necessary during implementation to improve effectiveness.

Planning at the project level includes identifying measurement needs, selecting measures and measurement scales, establishing data collection and analysis methods, setting target values and thresholds, and other planning activities, which are listed in Table A22. Measurement planning skills at the organizational level are also included. Plans are then implemented to perform measurement activities. Planning and performing measurement includes the activities listed in Table B22.

The skills and activities in this skill area apply equally to measurement of management attributes, such as schedule and budget. The emphasis of this skill area, however, is on measurement of process and product attributes. Process attributes to be measured may include items such as the percent of effort for various work activities, levels of rework for various work products, and so forth. Product measures may include items such as work products completed, rate of defect discovery and defect correction, and so forth. See the cited references for more information on process and product measures and measurement.

REFERENCES

[Abran 2010] Alain Abran, *Software Metrics and Software Metrology*, Wiley-IEEE Computer Society, 2010.

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

[IEEE 15528-2008] *IEEE Std. 15528-2008, IEEE Standard for Systems and Software Engineering—System Life Cycle Processes*, IEEE, 2008.

[IEEE 15939-2008] *IEEE Std. 15939-2008, Standard Adoption of ISO/IEC 15939:2007 System and Software Engineering Measurement Process*, IEEE, 2008.

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

Table A22

Software Measurement Skill Sets	Software Measurement Activities
Plan Measurement Process	• Identify measurement needs. • Define measures. • Select measures and measurement scales. • Establish data collection and analysis methods. • Set target values and thresholds. • Establish report formats and reporting procedures. • Identify measurement tools. • Procure and install measurement tools. • Plan for data storage.
Perform Measurement Process	• Use measurement tools and manual procedures to collect data. • Validate collected data. • Retain valid data in a repository. • Generate and distribute reports. • Identify and recommend improvements to the measurement process.

The following notations are used in Table B22: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B22

Software Measurement Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Plan Measurement Process				1. Participates in identifying measures for an organization. (P)	1. Defines attributes of the process and product measures to be used throughout an organization. (C)
			1. Participates in identifying measurement needs for a project or program. (P)	2. Leads identification of measurement needs for a project or program. (L)	2. Develops new ways to identify measurement needs. (C)
			2. Assists in selecting measures and measurement scales. (A)	3. Selects measures and measurement scales. (L)	3. Defines new measures and measurement scales. (C)
			3. Establishes data collection and analysis methods. (P)	4. Reviews and approves data collection and analysis methods. (L)	4. Develops new data collection and analysis methods. (C)

Table B22

Software Measurement Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Plan Measurement Process			4. Participates in setting target values and thresholds. (P)	5. Sets target values and thresholds. (L)	
			5. Participates in establishing report formats and reporting procedures. (P)	6. Establishes report formats and reporting procedures. (L)	5. Develops new report formats and reporting procedures. (C)
			6. Identifies measurement tools and manual procedures. (P)	7. Reviews and approves measurement tools and manual procedures. (L)	6. Develops new measurement tools and manual procedures. (C)
		1. Assists in planning for data storage. (A)	7. Plans for data storage. (P)	8. Reviews and approves plan for data storage. (L)	
		2. Assists in selecting data collection and analysis methods. (A)	8. Selects data collection and analysis methods. (P)	9. Approves data collection and analysis methods. (L)	

Table B22

Software Measurement Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Plan Measurement Process	1. Procures and installs measurement tools. (L)	3. Assists in identifying measurement tools and manual procedures. (A)	9. Selects measurement tools and manual procedures. (A)	10. Approves measurement tools and manual procedures. (A)	
Perform Measurement Process	1. Uses measurement tools to collect data. (P)	1. Uses manual procedures to collect data. (P)		1. Leads and coordinates the measurement process. (L)	1. Periodically reviews methods, tools, and techniques used to perform the measurement process. (C)
	2. Maintains valid data in a repository. (P)	2. Assists in validating collected data. (A)	1. Validates collected data. (P)	2. Approves tactical improvements to the measurement process. (L)	2. Modifies existing and develops new review methods, tools, and techniques used to perform the measurement process. (C)
	3. Generates and distributes reports (P)	3. Identifies and recommends improvements to the measurement process. (A)			

23. HUMAN-COMPUTER INTERACTION SKILL AREA

Design of human-computer interaction (HCI) and of user interfaces has been traditionally regarded as part computer science and part human factors. Software engineers have, by necessity, become increasingly involved in the development cycle of HCI analysis, design, implementation, evaluation, and deployment because the user interface is often the difference between a successful product and a product that is either difficult to use or not used at all. To the user, the interface is the system.

This skill area covers skills and activities specific to the development of HCIs. There is, however, a great deal of similarity to other skill areas. For example, requirements elicitation for HCIs has much in common with conventional elicitation but there are some requirements-gathering skills that are particular to the development of human-computer interfaces. Some activities are unique to HCI, for example, the activities of interaction style design.

REFERENCES

[Buxton 2007] Bill Buxton, *Sketching User Experiences: Getting the Design Right and the Right Design*, Morgan Kaufmann Publishers, 2007.

[ISO 9241-210:2010] *ISO 9241-210:2010, Ergonomics of Human-System Interaction*, ISO, 2010.

[ISO/IEC 25060:2010] *ISO/IEC 25060:2010, Systems and Software Engineering—Systems and Software Product Quality Requirements and Evaluation (SQuaRE)—Common Industry Format (CIF) for Usability: General Framework for Usability-Related Information*, ISO, 2010.

[Rogers 2011] Y. Rogers, H. Sharp, and J. Preece, *Interaction Design: Beyond Human Computer Interaction*, 3rd ed.,Wiley, 2011.

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

Table A23	
Human-Computer Interaction Skill Sets	**Human-Computer Interaction Activities**
Requirements	• Identify stakeholders who provide HCI requirements. • Determine a process model for HCI development. • Select HCI-related tools. • Identify target users and their attributes. • Develop user interface requirements. • Identify constraints on user interface implementation. • Prototype to elicit requirements. • Specify applicable standards. • Identify interface requirements between the user interface and system components.
Interaction Style Design	• Identify metaphors and conceptual models. • Identify interaction mode(s). • Document primary and exception use case scenarios. • Develop interaction dialogs. • Develop models and prototypes for interaction flow. • Design input error handling. • Establish two-way traceability to user interface requirements, test scenarios, and test cases. • Refine existing and develop new prototypes. • Design technical interfaces between the user interface and other system components.
Visual Design	• Design page/screen layout. • Select icons. • Design menus. • Select color theme, font styles, and sizes. • Develop mock-ups and sketches of screens.
Usability Testing and Evaluation	• Test user interface with a usability checklist. • Conduct heuristic or expert evaluations. • Identify and obtain representative test subjects. • Design usability tests. • Conduct usability tests with users. • Analyze and report the results of usability testing.
Accessibility	• Determine accessibility needs of special needs users (such as color-blindness, physical disabilities, hearing or vision loss). • Test for special-needs user accessibility. • Utilize tools and techniques to provide accessible interface elements.

The following notations are used in Table B23: Follow (F), Assist (A), Perform (P), Lead (L), Create (C).

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements		1. Identifies stakeholders to provide HCI requirements. (P)	1. Reviews identification of stakeholders to provide HCI requirements. (P)	1. Coordinates work activities for stakeholder identification. (L)	1. Modifies existing and creates new methods and tools for stakeholder identification. (C)
			2. Assists in selecting a process model for HCI interface development. (P)	2. Determines which process model approach will be used by the HCI team to develop the interface. (L)	
			3. Recommends HCI tools for project use. (A)	3. Selects HCI tools for project use. (P)	
	1. Assists in identifying target users. (A)	2. Identifies target users and their attributes. (P)	4. Reviews and refines target user identification and describes their relevant attributes. (P)		2. Modifies existing and creates new methods and tools for target user identification. (C)

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements		3. Assists in identifying user interface requirements. (A)	5. Leads identification of user interface requirements. (L)	4. Leads review and refinement of user interface requirements. (L)	
	2. Follows directions to create simple prototypes for use in eliciting user interface requirements. (F)	4. Designs and creates prototypes for use in eliciting user interface requirements. (P)	6. Reviews and refines prototypes, tests for elicitation, and refines user interface requirements. (P)		3. Modifies existing and creates new methods and tools for prototyping. (C)
			7. Uses prototypes to elicit and refine user interface requirements. (P)		
			8. Identifies technical interface requirements (between the user interface and system components). (P)	5. Identifies constraints on user interface implementation. (L)	

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Requirements			9. Designs the details of the technical interface requirements (between the user interface and system components). (P)	6. Defines the technical interface requirements (between the user interface and system components). (L)	4. Modifies existing and creates new methods and tools for specifying technical interface requirements. (C)
				7. Selects applicable standards. (L)	
Interaction Style Design	1. Follows instruction to assist in documenting use cases, scenarios, interaction dialogs, and storyboards. (F)	1. Documents use cases, scenarios, interaction dialogs, and storyboards. (P)	1. Reviews and refines use cases, scenarios, interaction dialogs, and storyboards. (L)	1. Leads and coordinates interaction-style design activities. (L)	1. Modifies existing and creates new methods and tools for interaction-style design. (C)
	2. Assists in identifying user input errors. (A)	2. Assists in identifying user input errors and error handling approaches. (A)	2. Identifies user input errors and error handling approaches. (P)	2. Selects and refines user error handling approaches. (P)	

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Interaction Style Design		3. Assists in identifying interaction modes. (A)	3. Applies metaphors and conceptual models to define interaction style. (P)	3. Selects metaphors and conceptual models. (L)	
			4. Identifies and refines interaction modes. (P)	4. Works with the system design team to establish component interfaces between the user interface and system components. (L)	2. Modifies existing and creates new methods and tools for interaction-style design. (C)
	3. Documents two-way traceability to requirements and to test cases and test scenarios. (P)	4. Establishes two-way traceability between use cases, scenarios, interaction dialogs, and storyboards and specific user interface requirements and acceptance criteria. (P)			

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Interaction Style Design	4. Follows directions to develop or refine interface prototypes. (F)	5. Develops interface prototypes. (P)	5. Reviews and refines interface prototypes. (P)		
Visual Design		1. Assists in designing page/screen layout. (A)	1. Designs page/screen layout. (P)	1. Revises/ approves final page/ screen lay-outs. (L)	1. Modifies existing and creates new methods and tools for visual design. (C)
		2. Assists in selecting from exist-ing icons and design-ing new icons. (A)	2. Selects from exist-ing icons and designs new icons. (P)	2. Revises/ approves icons and identifies new icons as needed. (L)	
		3. Assists in select-ing color theme, font styles, and font sizes. (A)	3. Selects color theme, font styles, and font sizes. (P)	3. Revises/ approves color theme, font styles, and font sizes. (L)	
		4. Assists in menu design. (A)	4. Designs menus. (P)	4. Reviews and refines menu designs. (L)	

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Visual Design			5. Reviews selections for color theme, font styles, and font sizes, and checks selection against applicable standards. (P)	5. Approves visual design components and reviews design with stakeholders and/or target users. (L)	
	1. Follows instructions to assist in the creation of mock-ups and sketches. (F)	5. Creates mock-ups and sketches. (P)	6. Reviews and revises mock-ups and sketches with stakeholders. (P)		
Usability Testing and Evaluation		1. Analyzes design with a usability checklist. (P)	1. Selects and tailors one or more usability checklists. (P)	1. Leads and coordinates usability testing and evaluation activities. (L)	1. Modifies existing and creates new methods and tools for usability testing. (C)
		2. Assists in identifying representative test subjects from the target user group. (A)	2. Identifies representative test subjects from the target user group. (F)	2. Approves selection of one or more usability checklists. (L)	

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Usability Testing and Evaluation		3. Assists in obtaining test subjects. (A)	3. Reviews results of checklist analysis and recommends design changes. (P/F)		
			4. Obtains test subjects. (P)	3. Approves selection of test subjects. (L)	
	1. Writes user tests that evaluate user behavior. (A)	4. Assists in designing usability tests. (A)	5. Designs usability tests. (P)	4. Reviews, refines, and finalizes usability tests. (L)	
	2. Assists in conducting usability tests and collecting data. (A)	5. Conducts usability tests and collects data. (P)	6. Supervises usability testing. (P/L)		
	3. Follows instructions to assist in analyzing results of usability testing. (F)	6. Analyzes results of usability testing. (P)	7. Makes recommendations based on analysis of usability testing results. (P)	5. Reviews and approves recommendations and results of usability testing. (L)	

Table B23

Human-Computer Interaction Skill Sets and Activities by Competency Level

Skill Sets	Levels				
	Technician	Entry Level	Practitioner	Technical Leader	Senior Software Engineer
Accessibility		1. Assists in identifying accessibility needs for user interfaces. (A)	1. Identifies accessibility needs for user interfaces. (P)	1. Leads and coordinates accessibility activities. (L)	1. Develops new tools and techniques for providing accessible interface elements. (C)
			2. Develops acceptance criteria and tests for accessibility aspects of the user interface. (P)	2. Determines which accessibility needs must be addressed in the user interface. (L)	
		2. Assists in identifying the needs for international accessibility (languages, cultural considerations, and so forth). (A)	3. Identifies the needs for international accessibility (languages, cultural considerations, and so forth). (P)	3. Determines the extent to which the user interface must accommodate needs for international accessibility. (L)	
		3. Uses the selected tools and techniques for implementing required accessibility. (A)	4. Selects tools and techniques for providing required accessibility. (P)		

24. APPENDIX A: CONTRIBUTORS

SWECOM Team Members

Mark Ardis, *Stevens Institute of Technology*
Dick Fairley, *Software and Systems Engineering Associates (S2EA)*, Team Leader
Thomas Hilburn, *Embry Riddle University*
Ken Nidiffer, *Software Engineering Institute*
Massood Towhidnejad, *Embry Riddle University*
Mary Jane Willshire, *Software and Systems Engineering Associates* (S2EA)
Kate Guillemette, *IEEE Computer Society*

Interviewees

Elias Abughazaleh, *Quality Assurance Director, Symantec Corporation*
Laura Jordan, *Technical Support Manager, Symantec Corporation*
Matthew Larson, *Software Development Manager, Symantec Corporation*
Stephen Link, *Chief Systems Engineer, Harris Corporation*
Kim Madler, *Software Development Director, Symantec Corporation*
Thomas Schabowsky, *System Engineer and Architect, Harris Corporation*

Subject Matter Reviewers

Alain Abran, *École de technologie supérieure (ÉTS)*
Bob Aiello, *CM Best Practices Consulting*
Radu Babiceanu, *Embry-Riddle Aeronautical University*
Wayne Babich, *Charles River Development*
Pieter Botman, *True North Systems Consulting*
Nick Brixius, *Embry-Riddle Aeronautical University*
Drew Calhoun, *Lockheed Martin*
Cynthia Calongne, *Colorado Technical University*
Don Gelosh, *Worcester Polytechnic Institute*
Shafagh Jafer, *Embry-Riddle Aeronautical University*
Andrew Kornecki, *Embry-Riddle Aeronautical University*
Susan K. Land, *Missile Defense Agency*
Phil Laplante, *Pennsylvania State University*
Nancy Mead, *Software Engineering Institute*
Mark Merkow, *Charles Schwab*
Donald Reifer, *Reifer Consultants LLC*
Annette Reilly, *Lockheed Martin*
Leanna Rierson, *Digital Safety Consulting*
Linda Shafer, *IEEE Press*
Richard Hall Thayer, *California State University*
Steve Tockey, *Construx*
Janusz Zalewski, *Florida Gulf Coast University*

Public Reviewers

Sultan M. Al Khatib, *UK*
Jim Albers, *USA*
Asghar Ali, *Canada*
Simich Arturo, *Peru*
Augusto E. Bernuy, *Peru*
Ulloa Rubio Bertha, *Peru*
Miklos Biro, *Austria*
Rafael Capilla, *Spain*
Lynn Robert Carter, *USA*
JiMna Cho, *South Korea*
Kyeong-Ho Choi, *South Korea*
Julio Cordoba Retana, *Costa Rica*
Mauricio N. Coria, *Argentina*
Varuna Eswer, *India*
Ernesto Exposito, *France*
Hemer Figueroa, *Colombia*
Garth Glynn, *UK*
Michael A. Jablonski, *USA*
SeungWon Jung, *South Korea*
Hwang Jung Sik, *South Korea*
Umut Kahramankaptan, *Belgium*
Pankaj Kamthan, *Canada*
Perry Kapadia, *USA*
Rameshchandra Bhaskar Ketharaju, *India*
Martin Kropp, *Switzerland*
Timothy C. Lethbridge, *Canada*
John Macasio, *Philippines*
Ana M. Moreno, *Spain*
Sam Mori, *South Korea*
Kiron Rao, *India*
Stephen C. Schwarm, *USA*
Jo SeongChan, *South Korea*
Clifford Shaffer, *USA*
Peraphon Sophatsathit, *Thailand*
Chedu Suh, *South Korea*
Chris Taylor, *USA*

Thomas R. Turner, *USA*
Sivaraj Veera, *India*
Mario Winter, *Germany*
HyungJin Yoon, *South Korea*

25. APPENDIX B: SWECOM INTENDED AUDIENCES

The intended audiences for this software engineering competency model (SWECOM) include individual software and systems engineering practitioners, their managers, and workforce planners. Others who may also find these models useful are indicated.

- *Software engineering and systems engineering practitioners* will use SWECOM for self-evaluation, self-improvement, and career planning. In addition, practitioners can use a competency model to provide guidance in selecting academic programs and training classes. SWECOM can also provide a framework for discussions with leaders and supervisors.
- *Managers of practitioners* will use SWECOM to select skills at various skill levels and group them into job roles and job descriptions, to establish performance criteria, to establish an objective basis for performance evaluations and development of career paths for individual practitioners, and to perform gap analysis.
- *Work force planners w*ill use SWECOM to develop skills inventories and perform gap analysis, to prepare workforce development plans, to define career ladders, and to select and hire employees, contract personnel, and contractor organizations.
- *Curriculum designers and other software engineering education researchers* will use SWECOM to design competency-based education and training curricula.

Others who may find SWECOM useful include

- *IEEE Computer Society*: for standards preparation, services to industry (training and consulting), curriculum development, assistance to schools and industry for development and assessment of professional development programs, and as a source of authoritative credentials and credibility in the computing professions.
- *Other professional societies*: to determine common interests, overlaps, and boundaries.
- *Legislative and legal bodies: to provide guidance for licensing criteria.*
- *Regulatory agencies*: to provide guidance in establishing regulations that impact the health, safety, and welfare of the general population.
- *Others*: to find novel uses for SEWCOM not envisioned by the developers.
- *Society at large*: to use as a model for increasing the number of competent software and systems engineering professionals.

26. APPENDIX C: SWECOM USE CASES

USE CASE #1:
Organization Using SWECOM to Create a New Hire Job Description and Screen Job Candidates

Goal: An organization will use SWECOM to create a job description and hire personnel.

Actors: A hiring manager and human resource personnel

Preconditions and Assumptions:

1. There is a need to hire additional personnel with specific competencies.
2. The organization has access to SWECOM and the Staffing Gap Analysis worksheet.

Trigger: There is a need to hire additional personnel with specific competencies.

Normal Flow:

1. A manager has identified a need for additional personnel.
2. The manager uses SWECOM and the Staffing Gap Analysis worksheet to identify the needed skills.
3. The manager, in collaboration with human resource personnel, develops a job description to be posted.

4. Human resource personnel conduct an initial screening of applicants, using the job description and competency model as a guide.
5. The manager and/or human resource personnel use SWECOM to choose the best candidate who meets the needs of the organization.

Post Conditions:
1. The organization has identified the needed skills.
2. The organization has hired an employee who best matches the needed skills.

USE CASE #2:
Employee Using SWECOM for Self-Improvement

Goal: An employee will use SWECOM to evaluate his or her software systems engineering competencies for the purposes of self-evaluation and improvement.

Actor: A software engineer working in a software industry.

Preconditions and Assumptions:
1. The employee has identified an interest in software systems engineering.
2. The employee has access to SWECOM and the Individual Gap Analysis worksheet.

Trigger: The employee wants to conduct a self-evaluation of his or her own capabilities in software systems engineering and/or is interested in improving his or her capabilities in software systems engineering in order to reach a higher level of competency.

Normal Flow:
1. The employee has identified the software systems engineering skills he or she is interested in to conduct self-evaluation.
2. The employee conducts a self-evaluation against each activity in the skills of interest.

3. The employee identifies his or her current competencies.
4. The employee uses the Individual Gap Analysis Worksheet to document his or her competencies for the selected skills.

Post Conditions:

1. The employee has a good understanding of his or her current competency levels.
2. The employee has a good understanding of activities to be improved to reach the desired competency levels for those activities.

USE CASE #3: Manager Using SWECOM for Evaluation and Improvement Planning for Team Member

Goal: A technical manager will use SWECOM and the Staffing Gap Analysis worksheet to evaluate a member of his or her team and/or develop an improvement plan for the team member.

Actor: A technical manager

Preconditions and Assumptions:

1. The team member to be evaluated or guided has been identified.
2. The team member's areas of expertise or his or her roles and responsibilities in the team have been identified.
3. The manager has access to and familiarity with SWECOM and the Staffing Gap Analysis worksheet.

Trigger: The manager has identified the need to assess the team member's competencies and/or develop an improvement plan to guide him or her in advancement.

Normal Flow:

1. The manager has identified skill areas that apply to the team member's present and future roles and responsibilities.
2. The result of the gap analysis will be used as part of the team member's evaluation and preparation of an improvement plan.

Post Conditions:

1. The manager has completed the evaluation of the team member.
2. The manager and team member have met to discuss the manager's evaluation of the team member's present competencies and competencies needed in the future.
3. The team member has a good understanding of what he or she needs to improve in order to eliminate the existing gap between his or her present competencies and desired competencies.

USE CASE #4:
Curriculum Designer Using SWECOM to Prepare a Competency-Based Curriculum

Goal: A curriculum designer will use SWECOM 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 skill or skill area.

Primary Actor: A curriculum designer

Secondary Actor: A sponsoring academic or industrial organization

Preconditions and Assumptions:

1. The skill area (or areas) and the level (or levels) of competency to be achieved have been identified.
2. The curriculum designer has access to and familiarity with SWECOM and the Staffing or Individual Gap Analysis worksheet.

Trigger: An academic or industrial organization has identified the need to improve the identified competencies of students or employees and has commissioned a curriculum designer to prepare a competency-based curriculum for one or more skills or skill areas at a stated level (or levels) of competency to be achieved.

Normal Flow:

1. The curriculum designer conducts a gap analysis to determine the skills that candidate students or employees currently have, the learning outcomes to be achieved and demonstrated, and the gap to be closed by education and/or training.
2. The curriculum designer prepares the curriculum, including topics to be covered, reference materials, facilities needed, and the learning outcomes to be demonstrated.
3. The curriculum designer presents the curriculum to the academic or industrial organization and makes requested changes.

Post Conditions:

1. The curriculum designer has completed preparation of the desired curriculum.
2. The academic or industrial organization has reviewed and approved the curriculum, perhaps after requested revisions are made by the curriculum designer.
3. The resultant curriculum is suitable as a basis for designing courses, preparing materials, and acquiring necessary facilities and resources.

27. APPENDIX D: GAP ANALYSIS WORKSHEETS

Staffing Gap Analysis Worksheet			
Date Completed: [xxx]			
Organizational Unit: [xxx]			
Completed by: [names and titles of those completing the worksheet]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Software Requirements Skills			
Software Requirements Elicitation			
Software Requirements Analysis			
Software Requirements Specification			
Software Requirements Verification and Validation			
Software Requirements Process and Product Management			
Software Design Skills			
Software Design Fundamentals			
Software Design Strategies and Methods			
Software Architectural Design			
Software Design Quality Analysis and Evaluation			
Software Construction Skills			
Software Construction Planning			
Note: Analysis may be at the level of skill area or skill			
Note: Have, **Need**, and **Gap** indicate the number of individuals and the competency level			

Staffing Gap Analysis Worksheet			
Date Completed: [xxx]			
Organizational Unit: [xxx]			
Completed by: [names and titles of those completing the worksheet]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Managing Software Construction			
Detailed Design and Coding			
Debugging and Testing			
Integrating and Collaborating			
Software Testing Skills			
Software Test Planning			
Software Testing Infrastructure			
Software Testing Techniques			
Software Testing Measurement and Defect Tracking			
Software Sustainment Skills			
Software Transition			
Software Support			
Software Maintenance			
Software Process and Life Cycle Skills			
Software Development Life Cycle Implementation			
Process Definition and Tailoring			
Process Implementation and Management			
Process Assessment and Improvement			
Software Systems Engineering Skills			
System Development Life Cycle Modeling			
Concept Definition			
System Requirements Engineering			
System Design			
Requirements Allocation			
Component Engineering			
System Integration and Verification			
Note: Analysis may be at the level of skill area or skill			
Note: Have, **Need**, and **Gap** indicate the number of individuals and the competency level			

Staffing Gap Analysis Worksheet			
Date Completed: [xxx]			
Organizational Unit: [xxx]			
Completed by: [names and titles of those completing the worksheet]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
System Validation and Deployment			
System Sustainment Planning			
Software Quality Skills			
Software Quality Management (SQM)			
Reviews			
Audits			
Statistical Control			
Software Security Skills			
Requirements			
Design			
Construction			
Testing			
Process			
Quality			
Software Safety Skills			
Requirements			
Design			
Construction			
Testing			
Process			
Quality			
Software Configuration Management Skills			
Plan SCM			
Conduct SCM			
Manage Software Releases			
Software Measurement Skills			
Plan Measurement Process			
Note: Analysis may be at the level of skill area or skill			
Note: Have, **Need**, and **Gap** indicate the number of individuals and the competency level			

Staffing Gap Analysis Worksheet			
Date Completed: [xxx]			
Organizational Unit: [xxx]			
Completed by: [names and titles of those completing the worksheet]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Perform Measurement Process			
Human-Computer Interaction Skills			
Requirements			
Interaction Style Design			
Visual Design			
Usability Testing and Evaluation			
Accessibility			
Note: Analysis may be at the level of skill area or skill			
Note: Have, **Need**, and **Gap** indicate the number of individuals and the competency level			

Individual Gap Analysis Worksheet			
Date Completed: [xxx]			
Gap Analysis for: [name of individual]			
Names and Titles of Other Participants: [xxx]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Software Requirements Skills			
Software Requirements Elicitation			
Software Requirements Analysis			
Software Requirements Specification			
Software Requirements Verification and Validation			
Software Requirement Process and Product Management			
Software Design Skills			
Software Design Fundamentals			
Software Design Strategies and Methods			
Software Architectural Design			
Software Design Quality Analysis and Evaluation			
Software Construction Skills			
Software Construction Planning			
Managing Software Construction			
Detailed Design and Coding			
Debugging and Testing			
Integrating and Collaborating			
Software Testing Skills			
Software Test Planning			
Software Testing Infrastructure			
Software Testing Techniques			
Software Testing Measurement and Defect Tracking			
Software Sustainment Skills			
Software Transition			
Note: Have indicates that competencies for all activities for a skill at the indicated level have been demonstrated (e.g., L2); **Gap** indicates the difference			
Note: Need includes the activity numbers for which competency must be demonstrated to advance to the next level for that skill (e.g., L3: A2, A5, or perhaps L3: all)			

Individual Gap Analysis Worksheet			
Date Completed: [xxx]			
Gap Analysis for: [name of individual]			
Names and Titles of Other Participants: [xxx]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Software Support			
Software Maintenance			
Software Process and Life Cycle Skills			
Software Development Life Cycle Implementation			
Process Definition and Tailoring			
Process Implementation and Management			
Process Assessment and Improvement			
Software Systems Engineering Skills			
System Development Life Cycle Modeling			
Concept Definition			
System Requirements Engineering			
System Design			
Requirements Allocation			
Component Engineering			
System Integration and Verification			
System Validation and Deployment			
System Sustainment Planning			
Software Quality Skills			
Software Quality Management (SQM)			
Reviews			
Audits			
Statistical Control			
Software Security Skills			
Requirements			
Design			
Note: Have indicates that competencies for all activities for a skill at the indicated level have been demonstrated (e.g., L2); **Gap** indicates the difference			
Note: Need includes the activity numbers for which competency must be demonstrated to advance to the next level for that skill (e.g., L3: A2, A5, or perhaps L3: all)			

Individual Gap Analysis Worksheet			
Date Completed: [xxx]			
Gap Analysis for: [name of individual]			
Names and Titles of Other Participants: [xxx]			
Competencies (from Tables A and B of the SWECOM Skill Areas)			
Skills	**Have**	**Need**	**Gap**
Construction			
Testing			
Process			
Quality			
Software Safety Skills			
Requirements			
Design			
Construction			
Testing			
Process			
Quality			
Software Configuration Management Skills			
Plan SCM			
Conduct SCM			
Manage Software Releases			
Software Measurement Skills			
Plan Measurement Process			
Perform Measurement Process			
Human-Computer Interaction Skills			
Requirements			
Interaction Style Design			
Visual Design			
Usability Testing and Evaluation			
Accessibility			
Note: Have indicates that competencies for all activities for a skill at the indicated level have been demonstrated (e.g., L2); **Gap** indicates the difference			
Note: Need includes the activity numbers for which competency must be demonstrated to advance to the next level for that skill (e.g., L3: A2, A5, or perhaps L3: all)			