ASEET TUTORIAL TIMELINE

TIME: ACTIVITY [PERSON]

000: TUTORIAL INTRODUCTION [J]
005: CC2020 PROJECT OVERVIEW AND COMPETENCY [J]
020: SWECOM MODEL OVERVIEW AND SKILLS [P]
030: INDUSTRY PERSPECTIVES, GRADUATE ATTRIBUTES [P]
040: SWE CC2020 DRAFT COMPETENCIES [N]
050: Q&A ON PRESENTATIONS [ALL]
055: ACTIVITY 1 INSTRUCTIONS [J,P,N]
060: BREAKOUT1: CC2020/SWECOM SAMPLES [A]
075: DISCUSSION: COMPETENCIES FOR SWE [ALL]
090: PAUSE/BREAK
100: ACTIVITY2 INSTRUCTIONS [N, P, J]
105: BREAKOUT2: CREATE SWE COMPETENCIES [A]
135: REPORT BACK [ALL]
155: FUTURE SWE EDUCATIONAL ACTIVITIES [ALL]
170: CLOSING REMARKS [P, N, J]
180: ADJOURNMENT [ALL]

A=Audience, N=Nancy, P=Pierre, J=John, All=Everyone

CC2020 Report (2020)

http://www.cc2020.net/
https://cc2020.nsparc.msstate.edu/

SWECOM Report (2014)

https://ieeecs-media.computer.org/media/education/swebok/swecom.pdf

The CC2020 Competency Model

The CC2020 project specifies competency to be three components within the performance of a task in context.

Competency = [Knowledge + Skills + Dispositions] in Task/Context

- Figure A illustrates the meaning of competency taken from the CC2020 report.
- It illustrates a structure of knowledge, skills, and dispositions that are observable in the accomplishment of a task in context.
- Competency emphasizes the overlapping or intersection of knowledge, skills, and dispositions. That is, competency must involve all these three components.

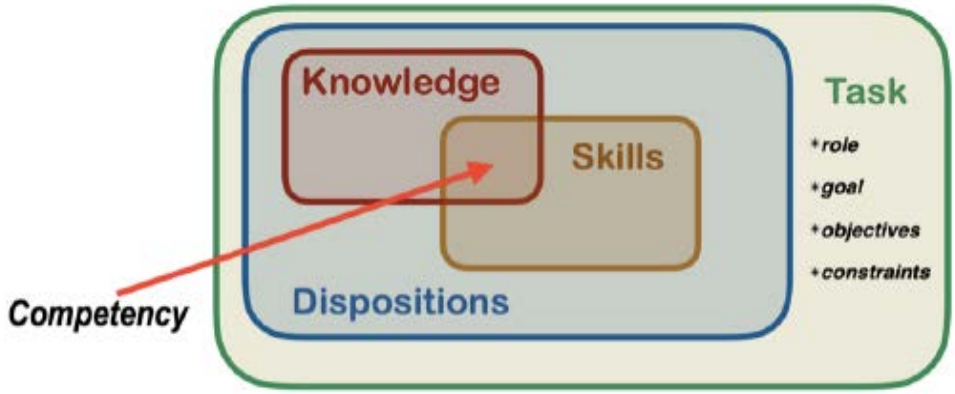

Figure A

Knowledge is the "know-what" dimension of competency that is factual.

Skills express the "know-how" and usually develop over time and with practice.

Dispositions frame the "know-why" dimension of competency, which prescribes a requisite character or quality in task performance.

Task is the construct that frames the skilled application of knowledge and makes dispositions concrete.

Software Engineering Knowledge Elements

Table A identifies software engineering knowledge elements from CC2020 (Appendix C.2.5) which are derived notably from SE2014. Appendix C.2.5 appears at the end of this document.

Table A
Software Engineering Knowledge Elements (in alphabetical order)

Behavioral Attributes	Software Quality
Human-Computer Interaction	Software Requirements
Project Management	Software Safety
Software Configuration Management	Software Security
Software Construction	Software Sustainment
Software Design	Software Systems Engineering
Software Measurement	Software Testing
Software Process and Life Cycle	

Professional Knowledge Elements

Table B shows thirteen professional knowledge elements with their meaning as presented in the CC2020 report.

Table B
Professional Knowledge Elements from CC2020

Knowledge Elements	Meaning
Analytical and Critical Thinking:	A mental process of simplifying complex information into basic parts and evaluating results to make proper decisions.
Collaboration and Teamwork:	Apportion challenging tasks into simpler ones and then work together to complete them efficiently.
Ethical and Intercultural Perspectives:	Ethical perspectives are the different viewpoints someone uses to view a problem in the context of individual human values.
Mathematics and Statistics:	Use of numbers and theories abstractly especially in the collection and analysis of numerical data
Multi-Task Prioritization and Management:	Processing several issues or tasks at once while arranging them according to importance to do specific one first.
Oral Communication and Presentation:	Conveying a message orally using real-time presentations with visual aids related to audience interests and goals.
Problem Solving and Trouble Shooting:	A logical and orderly search for the source of a unit problem and making the unit operational again.
Project and Task Organization and Planning:	A process to provide decisions about a project concerning organization and planning to achieve a successful result.
Quality Assurance / Control:	Use of techniques, methods, and processes to identify and prevent defects according to defined quality standards.
Relationship Management:	A strategy to maintain an ongoing level of engagement usually between a business and its customers or other businesses.
Research and Self-Starter/Learner:	Someone who begins or undertakes work or a project without needing direction or encouragement to do so.
Time Management:	An ability to use a person's time effectively or productively to work efficiently in multiple environments.
Written Communication:	Use of a written form of interaction between people and organizations that provides an effective way of messaging.

Levels of Cognitive Skills

Table C describes six skill levels as they appear in the CC2020 report. These skill levels are commensurate with modern taxonomies to describe levels of performance.

Table C
Levels of Cognitive Skills from CC2020

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Exhibit memory of previously learned materials by recalling facts, terms, basic concepts, and answers	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions.	Solve problems to new situations by applying acquired knowledge, facts, techniques, and rules in a different way	Examine and break information into parts by identifying motives or causes; make inferences and find evidence to support solutions.	Present and defend opinions by making judgments about information, validity of ideas, or quality of material.	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions.

Dispositional Elements

Table D describes eleven dispositional elements as they appear in the CC2020 report. The set of dispositions is an essential characteristic of a well-structured competency.

Table D
Elements of Dispositions from CC2020

Element	Elaboration	Element	Elaboration
Proactive:	With initiative, self-starter, independent	Adaptable:	Flexible; agile, adjust in response to change
Self-directed:	Self-motivated, determination, independent	Collaborative:	Team player, willing to work with others
Passionate:	Conviction, strong commitment, compelling	Responsive:	Respectful; react quickly and positively
Purpose-driven:	Goal-driven, achieve goals, business acumen	Meticulous:	Attentive to detail; thoroughness, accurate
Professional:	Professionalism, discretion, ethical, astute	Inventive:	Exploratory. Look beyond simple solutions
Responsible:	Use judgment, discretion, act appropriately		

SWECOM Skill Areas

Table E lists the Software Engineering Life Cycle Skill Areas and Software Engineering Cross Cutting Skill Areas as they appear in the SWECOM report.

Table E
Software Engineering Skill Areas from SWECOM

Software Engineering Life Cycle Skill Areas	**Software Engineering Crosscutting Skill Areas**
Software Requirements Skills	Software Process and Life Cycle Skills
Software Design Skills	Software Systems Engineering Skills
Software Construction Skills	Software Quality Skills
Software Testing Skills	Software Security Skills
Software Sustainment Skills	Software Safety Skills
	Software Configuration Management Skills
	Software Measurement Skills
	Human-Computer Interaction Skills

Competency Cluster Template

It is possible to take the suggested elements to create a competency cluster based on knowledge (computing and professional knowledge), cognitive skills, dispositions, and SWECOM attributes. The elements derive from relevant tables A, B, C, D, and E in Figure B, combined from CC2020 and SWECOM.

Name of Area	
Competency Statement *A sentence to describe the competency*.	
Knowledge Elements [Tables A and B]	**Skills Level** [Table C]
Software Engineering Knowledge Element 1	Skill level 1
Software Engineering Knowledge Element 2	Skill level 2
Software Engineering Knowledge Element 3	Skill level 3
Software Engineering Knowledge Element 4	Skill level 4
- -	- - - - - - - - -
Software Engineering Knowledge Element *m*	Skill level *m*
Professional Knowledge Element 1	Skill level 1
Professional Knowledge Element 2	Skill level 2
Professional Knowledge Element 3	Skill level 3
- -	- - - - - - - - -
Professional Knowledge Element *n*	Skill level *n*

Dispositions [Table D]			
Disposition Element 1	Disposition Element 2	- - - - - - - - - -	Disposition Element *k*

SWECOM Skill Areas [Table E]
SWECOM Skill Area 1
SWECOM Skill Area 2
SWECOM Skill Area 3
- - - - - - - - - -
SWECOM Skill Area *p*

Figure B. Template structure of a SWE competency cluster

Sample SWE Competency: Example 1

Software Construction	
Competency Statement *As a member of a project team, evaluate a software system against modern software practices such as defensive programming, error and exception handling, and accepted fault tolerances in a runtime mode that considers state-based table-driven constructions on a large project*.	
Knowledge Elements [Tables A and B]	**Skills Level** [Table C]
Software Construction	Evaluating
Software Quality	Analyzing
Software Security	Applying
Software Configuration Management	Understanding
Software Design	Understanding
Software Testing	Understanding
Software Sustainment	Applying
Software Systems Engineering	Understanding
Analytical and Critical Thinking	Evaluating
Collaboration and Teamwork	Applying
Quality Assurance and Control	Evaluating
Written Communication	Applying

Dispositions [Table D]			
Professional	Responsible	Collaborative	Meticulous

SWECOM Skill Areas[Table E]
Software Design Skills
Software Construction Skills
Software Testing Skills
Software Sustainment Skills
Software Systems Engineering Skills
Software Quality Skills
Software Security Skills
Software Configuration Management Skills

Figure C. Sample SWE Competency Cluster 1

Sample SWE Competency: Example 2

Software Requirements	
Competency Statement *Verify and validate the requirements using standard techniques, including inspection, modeling, prototyping, and test case development, as a contributing member of a requirements team.*	
Knowledge Elements [Tables A and B]	**Skills Level** [Table C]
Software Requirements	Evaluating
Software Quality	Evaluating
Software Testing	Applying
Human-Computer Interaction	Understanding
Project Management	Understanding
Software Design	Understanding
Software Process and Life Cycle	Understanding
Software Safety	Understanding
Software Security	Understanding
Software Systems Engineering	Understanding
Analytical and Critical Thinking	Evaluating
Collaboration and Teamwork	Applying
Quality Assurance / Control	Evaluating
Written Communication	Applying

Dispositions [Table D]			
Professional	Responsible	Collaborative	Meticulous

SWECOM Elements [Table E]
Software Requirements Skills
Software Quality Skills
Software Testing Skills
Human-Computer Interaction Skills
Software Design Skills
Software Process and Life Cycle Skills
Software Security Skills
Software Safety Skills
Software Systems Engineering Skills

Figure D. Sample SWE Competency Cluster 2

Instructions for Activity 1

- Split into teams as directed
- Read ASEET - Background-Instructions
- Read Sample SWE competencies: Example 1 and Example 2, pp. 9-10
- Use the next slide to reflect on the sample competencies
- Identify a team leader who will report the result of the team's discussions

Consider the Following Points

Read the two example competencies and consider the following:

1. When you read each example competency, do you nod your head in agreement or scratch your head trying to figure out the point?
2. Are the competencies written at a consistent level with one another?
3. How well do they relate to the SWECOM material presented earlier?

C.2.5: Software Engineering Draft Competencies

Software Requirements

1. Identify and document software requirements by applying a known requirements elicitation technique in work sessions with stakeholders, using facilitative skills, as a contributing member of a requirements team.
2. Analyze software requirements for consistency, completeness, and feasibility, and recommend improved requirements documentation, as a contributing member of a requirements team.
3. Specify software requirements using standard specification formats and languages that have been selected for the project, and be able to describe the requirements in an understandable way to non-experts such as end users, other stakeholders, or administrative managers, as a contributing member of a requirements team.
4. Verify and validate the requirements using standard techniques, including inspection, modeling, prototyping, and test case development, as a contributing member of a requirements team.
5. Follow process and product management procedures that have been identified for the project, as a contributing member of the requirements engineering team.

Software Design

1. Present to business decision makers architecturally significant requirements from a software requirements specification document.
2. Evaluate and compare tradeoffs from alternative design possibilities for satisfying functional and non-functional requirements and write a brief proposal summarizing key conclusions for a client.
3. Produce a high-level design of specific subsystems that is presentable to a non-computing audience by considering architectural and design patterns.
4. Produce detailed designs for a client for specific subsystem high-level designs by using design principles and cross-cutting aspects to satisfy functional and non-functional requirements.
5. Evaluate software testing consideration of quality attributes in the design of subsystems and modules for a developer/manufacturer.
6. Create software design documents which communicate effectively to software design clients such as analysts, implementers, test planners, or maintainers.

Software Construction

1. Design and implement an API using an object-oriented language and extended libraries, including parameterization and generics on a small project.
2. Evaluate a software system against modern software practices such as defensive programming, error and exception handling, accepted fault tolerances, in a runtime mode that considers state-based table-driven constructions on a large project, as a member of a project team.
3. Develop a distributed cloud-based system that incorporates grammar-based inputs and concurrency primitives for a medium-size project and then conduct a performance analysis to fine-tune the system, as a member of a project team.

Software Testing

1. Perform an integrative test and analysis of software components by using black-box and use case techniques in collaboration the clients.
2. Conduct a regressive test of software components for a client that considers operational profiles and quality attributes specific to the application in accordance with empirical data and the intended usages.
3. Conduct a test utilizing appropriate testing tools focused on desirable quality attributes specified by the quality control team and the client.
4. Plan and conduct process to design test cases for an organization using both clear- and black-box techniques to measure quality metrics in terms of coverage and performance.

Software Sustainment

1. Describe the criteria for transition into a sustainment status and assist in identifying applicable systems and software operational standards.
2. Relate to the needs of operational support personnel for documentation and training, and help develop software transition documentation and operational support training materials.
3. Help in determining the impacts of software changes on the operational environment.
4. Describe the elements of software support activities, such as configuration management, operational software assurance, help desk activities, operational data analysis, and software retirement.
5. Perform software support activities; and interact effectively with other software support personnel.
6. Assist in implementing software maintenance processes and plans, and make changes to software to implement maintenance needs and requests.

Software Process and Life Cycle

1. Engage with a team to translate a software development process into individual areas of responsibility.
2. Commit to and perform tasks related to assigned or agreed upon areas of responsibility. [MEE: Would it make sense to designate "on time, or "with a reasonable explanation for and plan for addressing delays"]
3. Propose and provide a justification for software lifecycle process improvements based on team capacity, project progress data, and quality analysis as part of a software development team's retrospective activities.

Software Systems Engineering

1. Provide a description of system engineering concepts and activities to identify problems or opportunities, explore alternatives, create models and test them.
2. Develop the big picture of a system in its context and environment in order to simplify and improve system architectures for supporting system designers.
3. Develop interfaces, which interact with other subsystems. Use information hiding to isolate the contents and collaborations within subsystems, so that clients of the subsystem need not be aware of the internal design of subsystems.
4. Work effectively with engineers and developers from other disciplines to ensure effective interaction.

Software Quality

1. Distinguish quality attributes that are discernable at run-time (performance, security, availability, functionality, usability), from those not discernable at run-time (modifiability, portability, reusability, integrability, and testability), and those related to the intrinsic qualities of architecture and detailed design (conceptual integrity, correctness, and completeness). [Based on SWEBOK 2014]
2. Design, coordinate, and execute, within a project team, software quality assurance plans for small software subsystems and modules, considering the way in which quality attributes are discernable. Correspondingly, measure, document, and communicate appropriately the results.
3. Perform peer code reviews for evaluating quality attributes that are not discernable at run-time.
4. Explain the statistical nature of quality evaluation when performed on software execution; develop, deploy and implement approaches to collect statistical usage and testing outcome data; compute and analyze statistics on outcome data.
5. Interact with external entities including clients, users, and auditing agencies in conveying quality goals for processes and products.

Software Security

1. Apply the project’s selected security lifecycle model (e.g. Microsoft SDL), as a contributing member of a project team.
2. Identify security requirements by applying the selected security requirements method, as a contributing member of a software project team.
3. Incorporate security requirements into architecture, high-level, and detailed design, as a contributing member of a software project team.
4. Develop software using secure coding standards.
5. Execute test cases that are specific to security.
6. Adhere to the project’s software development process, as a contributing member of a software project team.
7. Develop software that supports the project’s quality goals and adheres to quality requirements.

Software Safety

1. Describe the principal activities with the development of software systems, which involve safety concerns (activities related to requirements, design, construction, and quality);
2. Create and verify preliminary hazard lists; perform hazard and risk analyses, identify safety requirements;
3. Implement and verify design solutions, using safe design and coding practices, to assure that the hazards are mitigated and the safety requirements are met;
4. Be aware of the consequences of the development of unsafe software, that is, the negative affect on those who use or receive services from the software.

Software Configuration Management

[None]

Software Measurement

1. Develop and implement plans for measurement of software processes and work products using appropriate methods, tools, and abilities.

Human-Computer Interaction

[None]

Project Management

1. Explain the principal elements of management for a small project team;
2. Assist in the managerial aspects of a small project team, including software estimation, project planning and tracking, staffing and resource allocation, and risk management;
3. Develop and implement plans for measurement of software processes and work products using appropriate methods and tools.
4. Work effectively with other team members in project management activities.

Behavioral Attributes

1. Engage with team members to collaborate in solving a problem, effectively applying oral and/or written communication skills. Work done towards team effort is accomplished on time; it is in compliance with the role played in the team: it uses established quality procedures; and it advances the team effort.
2. Assist in the analysis and presentation of a complex problem, taking into account the needs of stakeholders from diverse cultures, needs, and/or geographic locations. Help in developing a solution for the problem and presenting it to stakeholders, explaining the economic, social and/or environmental impact of the proposed solution. Identify areas of uncertainty or ambiguity, and explain how these have been managed.
3. Analyze software employment contracts from various social and legal perspectives, ensuring that the final product conforms to professional and ethical expectations, and follows standard licensing practices.
4. Locate and make sense of learning resources, and use these to expand knowledge, skills, and dispositions. Reflect upon one's own learning and how it provides a foundation for future growth.

Number of Draft Competencies = 56

Software Engineering Subgroup Members who are Task Force Members

Nancy Mead (Leader)
Hala Alrumaih
Marisa Exter
Rich LeBlanc
John Impagliazzo
Barbara Viola

Software Engineering Subgroup Members who are not Task Force Members (Contributors)

Kai H. Chang, Auburn University
Dick Fairley, Software and Systems Engineering Associates
Kevin Gary, Arizona State University
Thomas Hilburn, Embry-Riddle Aeronautical University
Gabriel Tamura, Universidad Icesi, Colombia
Chris Taylor, Milwaukee School of Engineering
Jim Vallino, Rochester Institute of Technology
Norha M. Villegas, Universidad Icesi, Colombia