Introduction to Competency Framework

Competence Model - Definitions and Terminology

Competency is defined as the sum of knowledge, skills, and dispositions in IT2017 [1], wherein dispositions are defined as *cultivable behaviors desirable in the workplace* [3].

Competency = Knowledge + Skills + Dispositions *in context*

In CC 2020 [2], competency was further elaborated as the sum of the three within the performance of a task. Instead of the additive model of IT 2017, CC2020 defined competency as the intersection of the three:

Competency = Knowledge ∩ Skills ∩ Dispositions

In CS2023, competency is treated as a point in a 3D space with knowledge, skills, and dispositions as the three axes of the space (Figure 1): all three are required for proper execution of a task.

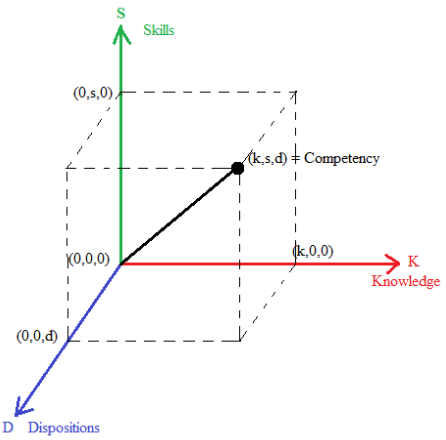

Figure 1. Competency as a point in a 3D space [3].

For a given learner at a given moment and a given task, competency is represented as a point in this 3D space. For a set of tasks, the competency of a learner is a point cloud in this 3D space.

A competency model of a curriculum is a set of competency specifications:

Competency Model = { Competency Specifications }

A competency specification consists of a competency statement and enumeration of the knowledge, skills, and dispositions needed to complete the task stated in the competency statement.

Competency Specification = Competency Statement + Knowledge + Skills + Dispositions

An ITiCSE working group has tried to design sample competency statements for computer science [4]. A process has also been proposed for converting a knowledge model to a competency model for computer science [5].

CS2023 Competency Model

In the specification of a competency, the task is the sole independent variable. The knowledge, skills, and dispositions needed to complete a task depend on the task and vary from one task to another. So, in CS2023, the description of the task was separated from the competency statement in a competency specification:

Competency Specification = Task + Competency Statement + Knowledge + Skills + Dispositions

In a competency specification:

- a task is a statement of what someone in a given role and context might be expected to do and is expressed in layman terms;
- the competency statement describes how a graduate might go about completing the task and is expressed in technical terms;
- knowledge is specified in terms of knowledge units in knowledge areas in the *Body of Knowledge* (Section 3);
- Skills are one or more of *Explain*, *Apply*, *Evaluate,* and *Develop*; and
- Dispositions are one or more of those identified as appropriate for the knowledge areas needed to complete the task.

The format used for competency specifications in CS2023 is as follows.

- **Task:** *What* someone in a given role and context might be expected to do.
- **Competency statement:** *What* a graduate might bring to bear in terms of knowledge and skills to attempt the task.
- **Required knowledge:** List of knowledge area-knowledge unit pairs needed to complete the task.
- **Required skills:** Explain / Apply / Evaluate / Develop
- **Desirable professional dispositions:** Adaptable / Collaborative / Inventive / Meticulous / Persistent / Proactive / Responsive / Self-Directed / Other

The first step in designing a competency model is to identify the tasks for which competency specifications will be written. But computer science being a general-purpose discipline of the study of solving problems with computers, the range of tasks for which it prepares graduates is vast. Not only is a comprehensive enumeration of tasks intractable, but it is also further complicated by the variability of the granularity and composition of tasks. Finally, local customization is an essential ingredient of competency models. So, a complete competency model of computer science, even if it were tractable, would be of limited off-the-shelf utility. Given these reasons, in CS2023, a competency framework has been proposed instead of an exhaustive competency model. The framework consists of 1) a framework for systematically

identifying tasks (described next); 2) a format for competency specifications (see previous box); and 3) an algorithm to design a customized competency model using the competency framework (described last in this Introduction to Competency Framework). Examples of sample tasks and competency specifications are included in Section 3.

CS2023 Framework for Systematically Identifying Tasks

The framework for systematically identifying tasks focuses on atomic tasks that can be combined to create compound tasks to suit local needs. The framework consists of three dimensions: component, activity, and constraint. In a task statement, the component is typically the noun, the activity the verb, and the constraint either an adjective or adverb.

The framework is elaborated for the three competency areas proposed in the knowledge model, that is, Software, Systems, and Applications. The following is an initial set of **components** in these three competency areas.

- Software: Program, algorithm, and language/paradigm
- Systems: Processor, storage, communication, architecture, I/O, data, and service
- Applications: Input, computation, output, and platform

An initial list of **activities** includes design, develop, document, evaluate, maintain, improve, humanize, and research. While most of the activities are self-explanatory, humanize refers to activities that address issues of society, ethics, and the profession, and research refers to activities that study theoretical underpinnings.

Constraints are categorized as follows:

- Problem constraints, e.g., task size (small versus large), problem (well-defined versus open-ended), task agent (solo versus in a team);
- Solution constraints, e.g., functional and non-functional requirements such as performance, availability, security, scalability, efficiency, reliability, cost; and
- Implementation constraints, e.g., parallel, distributed, virtualized.

The components, activities, and constraints listed above are representative, not prescriptive, or comprehensive. They may be visualized in three-dimensional space as shown in Figure 1 for the three competency areas. In the figure, the three axes use nominal scale, with no ordinality implied.

Each atomic task is a point in the three-dimensional space of component x activity x constraint. At the bottom-right of Figure 1 are the following three tasks mapped on software competency area:

- Develop a program for an open-ended problem (blue star);
- Evaluate the efficiency of a parallel algorithm (green star);
- Research language features for writing secure code (red star).

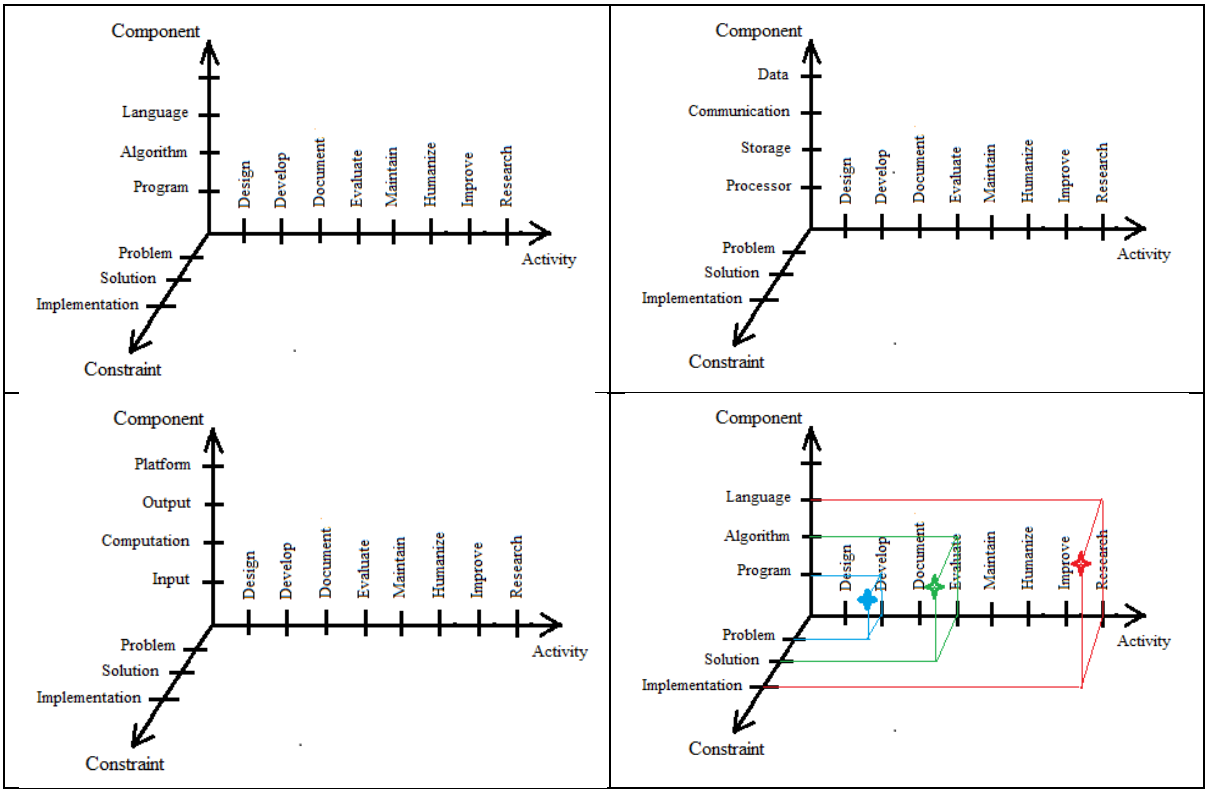

Figure 1: Task design space: Software competency area (top-left); Systems competency area (top-right); Applications competency area (bottom-left) and three tasks mapped on software competency area (bottom-right)

The framework is offered as a starting point for **generating** atomic tasks, not "classifying" them. One may want to add other components, activities, and constraints to the framework as appropriate for their local needs. It is expected that most competency specifications will be written for compound tasks created by combining two or more atomic tasks, e.g., "Design, implement, and document a parallelized scheduling program."

Designing/Revising a Curriculum Using the Competency Framework

A process for designing or revising a computer science curriculum using the CS2023 competency framework is as follows.

1. Identify the **competency area(s)** to be targeted by the curriculum in consultation with local stakeholders (academics, industry representatives, policy makers, etc.).
2. For each targeted competency area, use the component x activity x constraint task design space of the competency area to identify the atomic tasks for which the curriculum must prepare graduates – the task design spaces of Software, Systems, and Applications

competency areas are shown in Figure 1. The targeted atomic tasks will each be a point in the three-dimensional task design space as illustrated at the bottom-right in Figure 1.

3. Create compound tasks by combining two or more related atomic tasks.
4. Write a **competency specification** for each atomic or compound task identified in the previous two steps. In each specification,
    a. Exhaustively identify all the knowledge areas and knowledge units needed for the task.
    b. Identify the **skills** needed to complete the task.
        i. For the tasks that require CS and KA Core topics, re-design the tasks to require the skill levels recommended in the *Table of Core Topics* in Section 3.
    c. Identify the **dispositions** needed for the task – use the dispositions listed for the knowledge areas in step 4a as the starting point.

    The set of competency specifications for all the identified atomic/compound tasks constitutes the **competency model** of the curriculum.
5. Aggregate the required **knowledge areas** and **knowledge units** identified in the competency specifications of the competency model.
    a. Eliminate duplicate knowledge unit entries in the aggregation.
    b. Include additional knowledge units that are prerequisites for the knowledge units listed in the aggregation.
    c. Include additional knowledge units in the aggregation as necessary to ensure that all the **CS Core** topics are covered.
6. Package the knowledge units in the aggregation into courses. Adapt the **course packaging suggestions** of knowledge areas.
7. Sequence the courses to form a curriculum with prerequisites and co-requisites based on the following:
    a. **Mathematics requirements** listed for the courses identified in step 6;
    b. Software competency is a prerequisite for most other competency areas.
8. Complete the curriculum design loop.
    a. Verify that the knowledge areas aggregated in step 5 include all the knowledge areas that are part of the competency area(s) identified in Step 1.
    b. Add to the list of tasks identified in step 2, the tasks supported by the knowledge units added in steps 5b and 5c.

References

1. Sabin, M., Alrumaih, H., Impagliazzo, J., Lunt, B., Zhang, M., Byers, B., Newhouse, W., Paterson, W., Tang, C., van der Veer, G. and Viola, B. Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology. Association for Computing Machinery, New York, NY, USA, (2017).
2. Clear, A., Parrish, A., Impagliazzo, J., Wang, P., Ciancarini, P., Cuadros-Vargas, E., Frezza, S., Gal-Ezer, J., Pears, A., Takada, S., Topi, H., van der Veer, G., Vichare, A., Waguespack, L. and Zhang, M. Computing Curricula 2020 (CC2020): Paradigms for Future Computing Curricula. Technical Report. Association for Computing Machinery / IEEE Computer Society, New York, NY, USA, (2020).

3. Kumar, A. N., Becker, B. A., Pias, M., Oudshoorn, M., Jalote, P., Servin, C., Aly, S.G., Blumenthal, R. L., Epstein, S. L., and Anderson, M.D. 2023. A Combined Knowledge and Competency (CKC) Model for Computer Science Curricula. *ACM Inroads* 14, 3 (September 2023), 22–29. https://doi.org/10.1145/3605215
4. Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., Lunt, B., Maiorana, F., Pears, A., Pitt, F., Riedesel, C. and Szynkiewicz, J. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century. In Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '20). Association for Computing Machinery, New York, NY, USA, (2020), 211–246.
5. Clear, A., Clear, T., Impagliazzo, J. and Wang, P. From Knowledge-based to Competency-based Computing Education: Future Directions. In 2020 IEEE Frontiers in Education Conference (FIE). IEEE, New York, (2020), 1–7.