Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century

Alison Clear*
Eastern Institute of Technology
New Zealand
aclear@eit.ac.nz

Tony Clear*
Auckland University of Technology
New Zealand
tony.clear@aut.ac.nz

Abhijat Vichare*
ACM India
India
abhijatv@gmail.com

Thea Charles
Siegel Family Endowment
USA
thea.charles@siegelendowment.org

Stephen Frezza
Gannon University
Erie, PA, USA
frezza001@gannon.edu

Mirela Gutica
British Columbia Institute of Technology
Vancouver, Canada
mirela_gutica@bcit.ca

Barry Lunt
Brigham Young University
USA
luntb@byu.edu

Francesco Maiorana†
Universitá di Urbino Carlo Bo, Italy
Urbino, Italy
fmaioran@gmail.com

Arnold Pears
KTH Royal Institute of Technology
Stockholm, Sweden
pears@kth.se

François Pitt
University of Toronto
Ontario, Canada
fpitt@cs.utoronto.ca

Charles Riedesel
University of Nebraska, Lincoln
USA
chuckr@unl.edu

Justyna Szynkiewicz
Nord University
Norway
justyna.szynkiewicz@nord.no

ABSTRACT

The broadly influential document Computing Curricula 2005 (CC2005) is in the process of being updated through a project called Computing Curricula 2020 (CC2020). CC2020 provides a vision for the future of computing education, including a comprehensive report that contrasts curricular guidelines, and contextualizing those guidelines within the broader landscape of computing education. In the process, a framework of competency-based educational principles has been developed which is closely aligned with other skills and qualifications frameworks. This working group report demonstrates one way in which the transition from current learning-outcomes-based practices to the competency-based practices can be approached. Further, the paper discusses the challenges and insights that have emerged as the learning outcomes for various Knowledge Areas in the CS2013 report were re-expressed in terms of competencies.

CCS CONCEPTS

• **Social and professional topics** → **Model curricula**; *Computing education programs*; Student assessment; Testing, certification and licensing.

KEYWORDS

Competencies, Curriculum guidelines, CS2013, CC2020

ACM Reference Format:
Alison Clear, Tony Clear, Abhijat Vichare, Thea Charles, Stephen Frezza, Mirela Gutica, Barry Lunt, Francesco Maiorana, Arnold Pears, François Pitt, Charles Riedesel, and Justyna Szynkiewicz. 2020. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century. In *2020 ITiCSE Working Group Reports (ITiCSE-WGR '20), June 17–18, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 36 pages. https://doi.org/10.1145/3437800.3439208

*Working group leader
†Also with Kansas State University, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ITiCSE-WGR '20, June 17–18, 2020, Trondheim, Norway

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-8293-9/20/06…$15.00
https://doi.org/10.1145/3437800.3439208

1 INTRODUCTION

In 2016 the Association for Computing Machinery (ACM) initiated, and along with the IEEE Computer Society, sponsored the Computing Curricula 2020 project (in-press) to update the influential, Computing Curricula 2005 document [33]. The CC2020 project examines the current state of curricular guidelines for academic programs granting bachelor degrees in computing. It also provides a vision for the future of computing. The project intends to produce a comprehensive report that contrasts curricular guidelines and to contextualize these guidelines in the landscape of a competency-based educational framework for computing education.

As a part of its vision, the CC2020 project inspired the ITiCSE 2018 working group [24] that formulated the "CoLeaF model" (Competency Learning Framework) as a mechanism to express competencies within the computing disciplines. This work was further refined as a competency-based educational framework [25, 67]. Competencies, in essence, place the knowledge units and the skills to be developed within the context of a task, and a set of associated dispositions. This working group uses CoLeaF to frame and expound in a concrete manner a set of competencies based on the content of the CS2013 curriculum.

In contrast to a competency approach, learning outcomes tend to be expressed in terms of knowledge units and skills only. They thus fall short when we are faced with challenges like assessing the development of professional graduate attributes, or performing comparisons of educational programs. The increasing demands posed by professional contexts on graduates also place pressures on academic curricula to support the creation of what have been termed "adaptive experts" [13]. Burke and Bailey recommend strategies such as emphasising meta-cognitive and collaborative activities, and providing opportunities for the transfer of knowledge and a focus on growing learner adaptability.

The CS2013 Curriculum Report [1] provides curricular guidelines centred around a *Body of Knowledge* that is subdivided into 18 *Knowledge Areas* (KAs), corresponding to topical areas of study in computing. Each KA is further subdivided into *Knowledge Units* (KUs), corresponding to a coherent group of *Topics* and associated *Learning Outcomes* (LOs). Each LO within a KU is labelled with a level of *Mastery*, a rough indicator of the LO's level on the Revised Bloom's Taxonomy [3]: *Familiarity* (corresponding to Bloom's Remembering and Understanding), *Usage* (corresponding to Bloom's Applying), or *Assessment* (corresponding to Bloom's Analyzing, Evaluating and Creating).

In addition to this structure, each topic within the Body of Knowledge is identified as either *Core/Tier-1*, *Core/Tier-2* or *Elective*. The intention is that every curriculum should include all Core/Tier-1 topics, almost all Core/Tier-2 topics and a significant number of Elective topics. (Each topic is also assigned a number of *Curricular Hours*, representing the nominal number of lecture hours required to cover the topic.)

The goal of our Working Group was to build upon the work of an ITiCSE 2018 working group [24] to redevelop the CS2013 learning-outcomes-based statements in terms of the new CC2020 competency-based approach. This work attempts to be an "application of CoLeaF for curricula purposes…as a means to specify competencies in specific contexts, for example, a degree curricula definition or at an educational institution." [24] This effort offers concrete examples of "how to" make a transition from a traditional learning outcomes oriented computing curriculum to one based upon competencies.

The aim of this report, therefore, is not to create a full model CS curriculum in a competency-based format. Rather, it is to illustrate the process for developing a competency-based curriculum, and to compare the significant differences between a knowledge/learning-outcome-based approach and a competency-based one.

The result includes a range of sample competency statements at various levels of detail, process(es) for developing competency statements, observations about the quality of statements developed, and observations about how competency-based statements might be applied by curriculum developers. The intent is that these results be useful to Computing Curriculum developers at multiple levels, from those assigned to global curricular efforts (such as planned review and revisions to CS2013), to those with the task of applying competency modeling to their individual programs of study.

1.1 Our ITiCSE 2020 Working Group

The virtual nature of ITiCSE2020 required innovative processes and adoption of new technologies in order to communicate, coordinate, archive the underlying empirical work, and manage the process of authoring the report.

1.1.1 Working Virtually. ITiCSE working groups have traditionally had three stages of activity—before the conference (remotely), at the conference (face to face) and after the conference (remotely again). So, aspects of remote and global working were innate in the working group model [16]. However, this year with the global COVID-19 pandemic raging, the working groups had to adapt to a fully virtual mode. As a virtual group we had a combination of the aspects of an "ad-hoc group" [41], as some of us had never met, and aspects of an "established group" (ibid.) as many of us had worked, and continue to work together, on a range of projects. The members of the group spanned several continents—India, Europe and Scandinavia, North America, Australasia including New Zealand. This made synchronous sessions challenging as "global distance" [5] impacted the work of the group. Members joined our joint Zoom video-conferencing sessions variously from midnight, 5:30am, 7:30pm and 10:30am–2:30pm. We established a pattern of weekly Zoom sessions from six weeks prior to the conference, to help plan and review progress, resolve issues, and set direction; members worked on the evolving project tasks during that time. To support remote group members and help give one another a prod to undertake their assigned tasks, we allocated pairs who could work together.

1.1.2 Infrastructure. In addition to the use of email for general, asynchronous communication, and *Zoom* as a synchronous technology for videoconferencing, chat and screen sharing, a range of supporting and mostly asynchronous "collaborative technologies" [19] also had to be established as critical infrastructure to support our work. As a repository for resources, a shared *Google Drive* with a set of directories was set up to hold shared documents such as: the CS2013 Curriculum document (CS2013), draft CC2020 report (CC2020) and related publications, our work in progress competency statements, templates, *MS-Word*, *Adobe PDF* documents and *MS-Excel* spreadsheets. The draft structure and report itself was set up in *Overleaf* as a shared online document, with the bibliography held in *Zotero* fed through to BibTeX. To ease the process for contributors and enable self-contained changes and ready restructuring of the report, the LaTeX document was set up with a separate file for each section. LaTeX also provided good support for the new ACM templates. To support project management we set up a *Trello* board to provide visibility on work allocations of sections of the report to individuals and pairs, by managing stages of workflow with lists of: "in progress," "current sprint," "next up" and "done" showing the status of work in progress.

1.1.3 During the Virtual Conference. During the conference proper, we migrated to a mode involving plenary zoom sessions every second day to pick up on developing issues, help determine direction, track progress on report sections and offer support to each other. We also broke into two smaller but more timezone compatible groups (e.g., NZ and Scandinavia, who met at the start or beginning of their respective days) for a synchronous daily meeting. In addition, activity continued asynchronously using our repertoire of collaborative technologies to develop competency statements (in shared spreadsheets) and document our efforts (in the shared draft report). The original pairs also continued to work together in approaches that best suited them for writing and reviewing report sections, and compiling and peer-reviewing competency statements.

2 BACKGROUND

The Competency Learning Framework (CoLeaF) was developed with the intention to facilitate the integration of competency concepts into computing education. Frezza et al. [24] provides a comprehensive analysis of the research establishing a case for competencies and their use in computing education, outlining a significant body of relevant education and computing literature on competencies, competency models, and the relevant theories involved. One of the major contributions of the CoLeaF work was the view of a competency as being something more than the sum of its components, and additionally as being a virtual or invisible characteristic that emerges in performance in relation to a disciplinary challenge [24].

The CoLeaF work notes that the application of competency modeling to curricula is not a new endeavour, see for instance [11, 28, 37, 51]. However, the development work for the CoLeaF model proposed a revised definition of competency rooted in the scholarly education and computing literature. Here, competency is viewed as a means of modeling educational goals. Frezza et al. [24] defines competency as "The personal qualities causally related to effective performance in an area of work", where an individual competency integrates Knowledge, Skills and Dispositions in a professional Context. The CoLeaF formulation appears unique, in that it suggests a different, more unified view of the *integration* of {Knowledge + Skill + Disposition}. Leveraging educational theory, it explicitly layers the professional context as an element that should remain explicit, rather than implicit [36, 42, 49]. Subsequently, this work has been refined [25, 67] to make the model more useful in practice in the context of ongoing efforts around the world.

Competency statements can be seen as a means to describe the value of education for participating stakeholders. Besides digital competencies for learners [17, 45, 57], frameworks have been highlighted for educators [10, 14, 59], surveyed among teachers [62], developed in respect to other elements of society ranging from citizens [64], educational organizations [34] and consumers [12]. Considering the rapid expansion and evolution of computing into an omnipresent educational priority, the application of competencies as a potent means of defining and communicating computing curricula is worthy of research and implementation. There are strong efforts around the world to push for the mandatory introduction of Computing/Computer Science, either as a stand-alone discipline or in the context of other disciplines [15], from primary school through higher education; the development of computing has even been envisaged in a playful environment from early development [18]. In this respect, competencies should go hand in hand with the curricula that indicate how to develop these competencies. In recent years, a great deal of research and work has been carried out, leveraging the ACM/IEEE computing curriculum[1] [1], and a new draft standard is under development by the CC2020 project[2]. Besides this work, many curricula are under development in other related subject areas, ranging from data science [20] to cybersecurity [58], with concurrent efforts to introduce competency into other domains, such as Information Literacy [30].

Alongside intended curricula, there are enacted curricula by teachers and educators [23] with complete resources [39] suitable for curriculum proposal [43]. Ways to organize, search for and deploy learning resources hierarchically that highlight relations such as prerequisites, deepening and repurposing have been explored in [27], and should leverage an ontological representation, such as the BBC Curriculum[3] at different levels of granularity, spanning learning resources through curricula content and competencies.

2.1 Musings About Knowledge and Learning

Below are listed a number of observations, disclaimers and other musings about the nature and structure of knowledge that exemplify the challenges in reducing and expressing computing as curriculum models, even with the power of competencies. The effort of this Working Group in this process was very substantial. Indeed, attempting to do so may seem like trying to nail jelly to a wall.

(1) There are many kinds of knowledge that need to be dealt with. The Knowledge Structure[4] lists 9 that are relevant to learning: conceptual, factual, representational, strategic, meta-cognitive, self, operational, procedural and problem-state. In order to compile a comprehensive teaching and learning framework, all of these should be considered. Simply the "know-what", "know-how", and "know-why" employed here play significantly different roles in delivery and assessment and connect to competency-based models differently than knowledge-based approaches.
(2) Curriculum models are by definition intended to encompass and delineate the knowledge elements needed to be taught, and this may give the impression that the teaching process involves transferring the elements from the instructors' brain into the students' brain. This may impel educators to be comprehensive in their teaching, and increases concerns about sequencing with prerequisites.
(3) Competencies are defined to add a component of "disposition" onto the knowledge units, along with describing capabilities of the student (future employee) needed to accomplish tasks that would be expected in the workforce (or later classes).
(4) Educators do not have enough contact teaching hours to be comprehensive, yet students clearly do achieve "success" without strictly following prerequisites or acquiring specific skills and knowledge.

[1] https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf
[2] https://cc2020.nsparc.msstate.edu/
[3] https://www.bbc.co.uk/ontologies/curriculum
[4] https://srri.umass.edu/topics/knowledge-structure/

(5) If our teaching is to succeed, our students must be prepared to learn and discover all those things that were missed in formal education, either through self study and/or creative inventive thought.
(6) Teaching computing can start with mathematical models of computation (of which there are several logically equivalent to each other), advance through the hardware and through higher level programming. However, going in the other direction also works (this is done in some departments at one of the author's home institution, for example). This implies that prerequisites are not adequate in ordering subject matter. We can begin with functional programming, OOP, etc. and they all can be made to work. Learning trajectories, i.e., the journey bringing learners to achieve the competencies, must be explored up to individual learner personalization.

2.2 Underpinning Learning Theories

It is crucial at this point to briefly distinguish between the two approaches to curriculum design adopted in our curriculum transformation process. Basing curriculum specification on *Learning Outcomes* drawing upon a defined *body of knowledge* as used in the CS2013 curriculum model, follows in a long line of educational thought and a search to resolve the inherent muddiness of goal driven approaches. The notion of learning outcomes is said to have "evolved from rational curriculum design and the work of Eisner" [2], whereby "the use of outcomes emphasises student achievement and affirms that curriculum planning should begin with what is learnt rather than what is taught" (ibid.). While learning outcomes are said to encompass the three elements of "core subject based outcomes, personal transferable outcomes and generic academic outcomes" (ibid.), undergraduate education as Allan has observed, typically has a "subject orientation". The CS2013 curriculum with its strong content focus, can thus be so viewed. The risk with such an approach is that the specification of curriculum goals becomes overly reductionist, and further assumes that students may unproblematically learn what is taught. Yet as Barnett (cited by [2]) has observed: "the outcome is to a considerable degree unpredictable: the tutor cannot fully control it, for it is the student that is, or is not, successful."

As a comparison, an equally substantial body of academic literature supports competency-based approaches [28, 32, 52, 63]. As Magnusson (1990) stated: "CBE [Competency Based Education] can be characterized as being an amalgamation of the work of several leading learning theorists: It contains elements of programmed instruction, clearly specified behavioral objectives, hierarchical methods of knowledge acquisition, and social learning techniques." [38]

Cognitive theories of teaching and learning focus on acquisition of knowledge and development of internal mental models. Engineering, mathematics and science education at the undergraduate level is based on knowledge acquisition and cognitivism [40]. However, using constructivist learning theories is most efficient in the development of professional competence [40]. Constructivist theories of teaching and learning state that learners actively build knowledge gained through experiences that are meaningful to them individually (theories based on Piaget's cognitive constructivism) or collectively (theories based on Vygotsky's social constructivism) [48, 50, 65]. Constructing meaning requires holistic understanding. Knowledge builds recursively and new cognitive structures are created on existing knowledge [6]. Opposed to the belief that formality and formalism are fundamental to computer science, Beynon argues that constructivist teaching and learning theories apply to computer science as "few disciplines potentially engage more with every aspect of life, in all its breadth and informality, than computing." [7] By designing a computer science curriculum based on competencies, we go beyond a knowledge-based curriculum and position the new curriculum into the realm of cognitive and social constructivism. As competencies integrate knowledge, skills and dispositions and are context-situated, we respond to the need of alignment of the intellectual and practical domains and integrate theory with professional practice [60, 61], including computer science education [24, 67].

In this transformation of the curriculum, we aim to avoid some of the weaknesses of learning-outcomes-based approaches such as undue subject focus and reductionist specification of fragments of learning [31], and benefit from the strengths of a competency-based approach as conceived in the CC2020 report. This is not to deny that competency-based approaches have been criticised for reductionism themselves. However as conceived by Gonczi (1994), competency can be viewed from a professionalism lens:

> The notion of competence is relational. It brings together disparate things—abilities of individuals (deriving from combinations of attributes) and the tasks that need to be performed in particular situations. Thus competence is conceived of as complex structuring of attributes needed for intelligent performance in specific situations. Obviously it incorporates the idea of professional judgment. This approach has been called the 'integrated' or holistic approach to competence. [29]

One of the more essential aspects of a holistic approach is the integration of context as a necessary feature of learning goals [35, 46]. One of the more difficult aspects of the knowledge-based approach common to CS2013 and other efforts at curriculum development is that decontextualised knowledge is seen to be universally applicable. However, its role in competence can only be observable in association with the application of professional skills in the engagement with a task or problem [24, 67]. This makes the specification of task essential to competency modeling. The advantage of the CoLeaF-based approach is that it distinguishes the contextual, natural language statement of a learning goal from its decontextualized details, details that disambiguate the meaning of the contextual statement. This being asserted, it leaves open the question of how to express competency statements that are general enough to be applicable but specific enough to be useful—and not too specific to the culture and educational context in which they were developed. This exploration is a key part of this exercise and is elaborated on in Section 6.

Our focus in the rest of this report is on documenting *how* to reframe existing curricular guidelines in the form of competencies,

and on identifying and describing the range of challenges that emerge in the process.

3 UNDERSTANDING COMPETENCY STATEMENTS

Competencies as a conceptual framework for valuing the outcomes of higher education can be traced back to the 1970s in the U.S. [28]. The resulting approach to studying skills to copy behaviour did not attract much of a following. It was only at the end of the 1990s that higher education began to show interest and renew its participation in the conversation of Klink, Boon and Schlusmans [24, 61].

The educational literature on competencies and their role in structuring the learning process is vast, encompassing as it does ideas about epistemology, agency, skill-acquisition, and the comparative claims of different competency frameworks. In fact, a literature study by Stoof, Martens and Van Merriënboer [56] places the word in the "wicked words" category, meaning that its limits are hard to determine, which makes complete agreement on its meaning elusive. Despite its continuing fuzziness, the term promises to be useful in bridging the gap between educational outcomes and job requirements [35, 60]. The development of the CoLeaF model in 2018 represented a major advancement in these definitions [24] and forms the basis for the understanding of competency presented here.

3.1 Competency Modeling for Computing Education

Learning goals, particularly at the program level, are intended to be future-looking and supportive of effective transactional teaching and learning. Well-modelled competencies provide the opportunity for learning goals to support both paradigms, as the modelling task requires uncovering unspoken expectations and documenting them in a way that supports both course delivery and future assessment [24].

The CS2013 curriculum guidelines are knowledge-based and are expressed through learning outcomes and skills. The new curriculum CC2020 recommends guidelines based on competencies which encompass knowledge (know-what), skills (know-how) and dispositions (know-why) all included in a task as part of a professional context.

CC2020 articulates a notion of competency as a practical educational goal that refines the Knowledge-Skill-Disposition (K-S-D) theorized in the 2018 CoLeaF model [24], where the specification of a competency integrates designations of knowledge, skills and dispositions that are observable in a work context. Figure 1 presents an overview of the CoLeaF model at the basis of the CC2020 approach to competency.

While the knowledge dimensions of computing have been extensively explored in various computing curricula, what is meant by skill and disposition have had significantly less focus. CC2020 adopts a practical refinement, where competency can be specified as being composed of a collection of K-S-D dimensions observed within the performance of a task [67]. Thus, in normal parlance, a competency specification consists of a natural-language competency statement, paired with a more disambiguated K-S-D-T specification leveraging well-understood K+S+D vocabularies [25, 67].

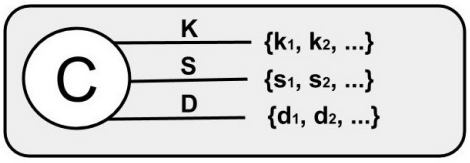


Figure 1: CoLeaF Model: Knowledge, Skill, Disposition components of a competency [24]

Competency = [Knowledge + Skills + Dispositions + Task]

Knowledge is the "know-what" dimension of competency that can be understood as factual. An element of knowledge designates a core concept essential to a competency. This dimension reflects the enumerated subject matter that teachers catalog as topics in their syllabi, departments distribute and balance among the courses they develop in an academic program, accreditation organizations stipulate in their accreditation criteria, and employers identify in job descriptions of their workers. Traditionally, curriculum guidelines for computing education have been predominated by the designation of knowledge elements composed of facts based upon scientific derivation or proof. This work takes on the CS2013 KA/KU/Topic/Learning Outcome statements as the knowledge vocabulary for competency specification. How these terms are organized is described in sub-section 3.2.1 below.

Skills refer to the capability and strategy for applying "know-what" to perform a task in context. Competency is realized when "know-what" knowledge is applied in action to accomplish a task, hence in application, skills express "know-how". Skills are most often developed over time and with practice. Consequently, skill development often requires engagement in a progressive hierarchy of higher-order cognitive process. For example, CC2020 reasonably adopts Bloom's levels of cognitive process [3] to specify the degree of skill expected in successful task accomplishment.

The skills dimension of competency is often assessed indirectly, through observation of the process or quality of work produced. The activation of "know-what" animated by "know-how" fuses the knowledge and skills dimensions. For that reason, the usefulness of any element of knowledge in a competency specification is only understandable when applied at a level of skillfulness, e.g., specified or observed as a level of Bloom's cognitive process. Therefore, an element of knowledge and the level of skill with which it is applied are necessarily and naturally conjoined as paired in the specification of a competency. In this way, the CC2020 competency model realizes a performance-based epistemology that animates an element of knowledge in achieving a task.

Dispositions frame the "know-why" dimension of competency, and prescribe a requisite character or quality in task performance. Dispositions shape the discernment of skillful engagement of "know-what" and "know-how". Specific to the task at hand, dispositions exert a moderating or controlling influence on a practitioner's choices by proposing or projecting a desirable quality onto the outcome. How dispositions moderate knowledge and skill could be thought of as the "extent that it accounts for the relation between the predictor and the criterion" [4] in that dispositions connect the

'better' or 'correct' application of knowledge and skill to the context in which they are applied. For example, dispositions moderate a practitioner's capabilities to discern a task as "professionally accomplished" rather than simply "completed". In this sense, dispositions are able to reflect the professional values behind a competency.

Dispositions characterize the socio-emotional tendencies, predilections and attitudes that characterize the inclination to carry out tasks and the sensitivity to know when and how to engage in those tasks [22, 47]. Hence, dispositions denote the values and motivation that guide applying knowledge while designating the quality of knowing commensurate with a standard of professional performance. "Know-why" exhibits as enacted values and because of the difficulty of assessing values and intention, disposition is typically assessed indirectly, through the observance of patterns of behavior or reflective practice.

Task is the construct that frames the skilled application of knowledge and makes dispositions concrete. Task instantiates a context for observing competency and provides the setting in which to manifest dispositions, *e.g.*, where individuals moderate their choices, actions and effort necessary to pursue and succeed in an efficient and effective manner. In this sense, task enfolds the purposeful context of competency, exposing the integral nature of knowledge, skills and dispositions. To this end, task definitions stipulate pragmatic engagement that reflect professional practice relevant to the particular vision for the program. In this sense, task descriptions provide an explicit context for the program to develop pedagogy and for graduates to demonstrate competency as a computing professional. The fact that each competency is framed with respect to a single task should *not* be construed as implying that competency = task. For example, a single task could lead to multiple competencies by focusing on different dispositions associated with its performance. Alternatively, the same competency components could be instantiated in different tasks.

3.1.1 Hierarchical Competency. An atomic competency is a competency specification that is independent of other specified competencies. However, a competency may be composite, e.g., dependent upon one or more other competency statements. Figure 2 depicts three compound competencies that all build upon/extend a shared atomic competency (the arrows go *from* the compound competencies *to* the atomic competency they all depend on). Composite competency specifications unfold as tree structures with branch and leaf nodes.

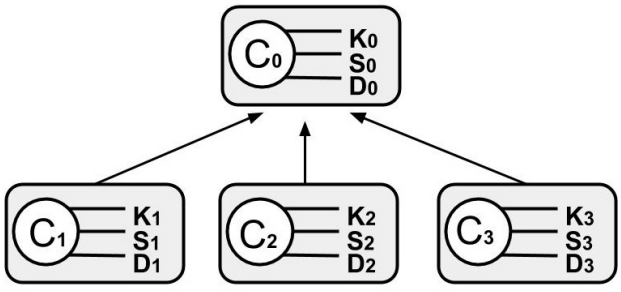


Figure 2: CoLeaF Model: Competency hierarchy [24]

Establishing a dependency among competencies would carry with it several implications about the components of the parent (presumably lower-level) competency statement(s) and the constituent/child (presumably higher-level) competency statement(s). A dependency implies the following about the components of the constituent to the parent:

(1) **Knowledge**: the parent knowledge elements are (in whole or in part) foundational for one or more knowledge elements in the constituent/child
(2) **Skill**: the parent competency is at the same or lower level of skill than the constituent/child

While the knowledge and skill levels carry necessary implications, it is not clear that the dispositional elements or task are necessarily dependent. This provides for a rich model that allows for differing levels of detail that can, but are not required to, depend on one another.

Figure 3 depicts both atomic and compound competencies, consistent with the CoLeaF and CC2020 models. The object of developing competency statements is to fill the model, that is to create natural-language competency statements and define the K+S+D+T components to disambiguate the statement.

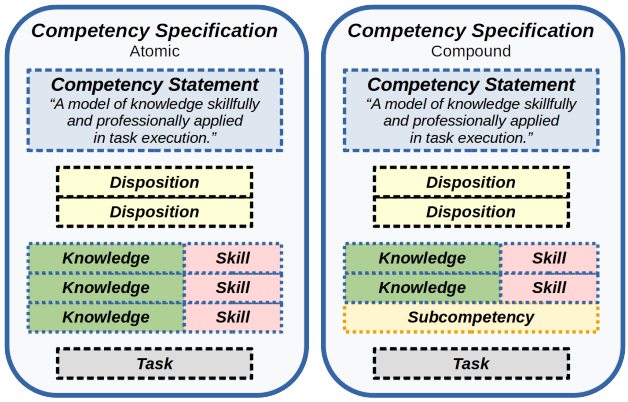


Figure 3: Atomic and Compound Competency Statements

3.2 Vocabularies

The ability to disambiguate competency statements is accomplished through the use of understood vocabulary for each of the K-S-D components. As suggested by Waguespack et al. [67], these vocabularies should be drawn from community-managed dictionaries that relate knowledge, skills and dispositions relevant to that domain. In computer science, the best generally accepted vocabulary for terms describing what can/should be known by a professional practitioner is the Computer Science Curricula 2013 (CS2013), particularly the structured Knowledge Areas (KAs) and Knowledge Units (KUs) [1]. Similarly, as computing relies heavily on applied cognitive skills, the Revised Bloom's Taxonomy [3] provides a significantly useful vocabulary. While not nearly as well studied as either knowledge or skill, recent work on the CC2020 project has yielded a useful set of dispositional statements applicable to computing competencies [25, 67].

3.2.1 Knowledge Vocabulary for Computer Science Competencies. The CS2013 document divides computer science knowledge into 18 KAs subdivided into 225 KUs. For example, the Software Engineering KA is divided into ten KUs:

(1) SE/Software Processes
(2) SE/Software Project Management
(3) SE/Tools and Environments
(4) SE/Requirements Engineering
(5) SE/Software Design
(6) SE/Software Construction
(7) SE/Software Verification and Validation
(8) SE/Software Evolution
(9) SE/Software Reliability
(10) SE/Formal Methods

Each of these KUs are themselves then broken down into Topics and Learning Outcomes that are expressed at a level of expectation for tertiary computer science programs: Tier-1, Tier-2 and Elective. To illustrate the depth of detail provided in CS2013, the following two tables present the detailed topic and learning outcome information for one of the KUs within the SE KA.

Table 1: CS2013 Tier-1, Tier-2 and Elective Topics defined for the SE/Requirements Engineering KU

Tag	Tier	Topic
SE-RE-a	1	Describing functional requirements using, for example, use cases or user stories
SE-RE-b	1	Properties of requirements including consistency, validity, completeness, and feasibility
SE-RE-c	2	Software requirements elicitation
SE-RE-d	2	Describing system data using, for example, class diagrams or entity-relationship diagrams
SE-RE-e	2	Non-functional requirements and their relationship to software quality (cross-reference IA/Secure Software Engineering)
SE-RE-f	2	Evaluation and use of requirements specifications
SE-RE-g	E	Requirements analysis modeling techniques
SE-RE-h	E	Acceptability of certainty/uncertainty considerations regarding software/system behavior
SE-RE-i	E	Prototyping
SE-RE-j	E	Basic concepts of formal requirements specification
SE-RE-k	E	Requirements specification
SE-RE-l	E	Requirements validation
SE-RE-m	E	Requirements tracing

Table 1 presents the topics defined in CS2013 for the *SE/Requirements Engineering* KU. All topics and learning outcomes are classified as *Tier-1*, *Tier-2* or *Elective*, depending on how they are expected to appear within a CS program.

The *SE/Requirements Engineering* KU also defines a number of learning outcomes (LOs) associated with these topics, presented in Table 2. Each LO is assigned a unique tag for traceability, a tier level (Tier-1, Tier-2 or Elective) as well as a mastery level (one of F: Familiarity, U: Usage or A: Assessment). While these mastery levels are explained in more detail in the CS2013 document, summary definitions are provided in Table 3. In that sense, CC2013's Mastery Levels are the counterpart to CC2020's Skill Levels. This is described in more detail in sub-section 3.2.4. Consistent with the CC2020 approach to competency modeling, these LOs provide a traceable means of disambiguating natural-language competency statements.

Table 2: CS2013 Tier-1, Tier-2 and Elective Learning Outcomes for the SE/Requirements Engineering KU

Tag	Tier	Level	Learning Outcome
SE-RE-1	1	F	List the key components of a use case or similar description of some behavior that is required for a system.
SE-RE-2	1	F	Describe how the requirements engineering process supports the elicitation and validation of behavioral requirements.
SE-RE-3	1	F	Interpret a given requirements model for a simple software system.
SE-RE-4	2	F	Describe the fundamental challenges of and common techniques used for requirements elicitation.
SE-RE-5	2	F	List the key components of a data model (e.g., class diagrams or ER diagrams).
SE-RE-6	2	U	Identify both functional and non-functional requirements in a given requirements specification for a software system.
SE-RE-7	2	U	Conduct a review of a set of software requirements to determine the quality of the requirements with respect to the characteristics of good requirements.
SE-RE-8	E	U	Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system.
SE-RE-9	E	F	Compare the plan-driven and agile approaches to requirements specification and validation and describe the benefits and risks associated with each.
SE-RE-10	E	U	Use a common, non-formal method to model and specify the requirements for a medium-size software system.
SE-RE-11	E	U	Translate into natural language a software requirements specification written in a formal specification language.
SE-RE-12	E	U	Create a prototype of a software system to mitigate risk in requirements.
SE-RE-13	E	F	Differentiate between forward and backward tracing and explain their roles in the requirements validation process.

3.2.2 Knowledge Areas Beyond CS2013. One of the weaknesses of the CS2013 effort in its application to competencies is the omission

Table 3: CS2013 Mastery Levels and their Skill-Level Equivalents [1]

	CS2013 Mastery Level	Bloom's Equivalent
Familiarity	The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept as opposed to expecting real facility with its application. It provides an answer to the question: "What do you know about this?"	B-II
Usage	The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question: "What do you know how to do?"	B-III
Assessment	The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question: "Why would you do that?"	B-IV or B-V

of knowledge areas expected of computing students. Most notably, CS2013 omits professional knowledge areas describing what can/should be known by a professional practitioner that extend broadly beyond computer science specifically. For example, a usable list of professional and foundational knowledge areas for computing has been included in computing curricular documents developed shortly after the CS2013 effort: IT2017 [52]. A sample set of distinct professional "knowledge-skill" categories are presented in Table 4 [25, 67].

Surprisingly few of these more general professional & foundational knowledge areas were explicitly identified during our mapping process. Most notable were PK-4 *Project/Task Organization & Planning* which parallels many aspects of the Software Project Management KU in the Software Engineering KA (SE-SPM). Similarly PK-11 *Quality Assurance / Control* parallels the SE-VV Verification and Validation KU, while PK-1 and PK-2 *Oral & Written Communication* parallel the Social Issues and Professional Practice (SP) Professional Communication KU.

When mapped as part of competency statements, these professional knowledge areas would be used in exactly the same way as the 18 knowledge areas defined in CS2013. They necessarily involve not just knowledge of (or know-about) something, but also skill in applying that knowledge. Although they may be coined as 'skills' or 'soft skills' in common parlance, as learned/developed these act like other KAs. In the context of competency development and description, these professional knowledge areas, like the other CS2013 KAs,

Table 4: Professional & Foundational Knowledge Areas extending CS2013

Tag	Knowledge Area
PK-1	Oral Communication & Presentation
PK-2	Written Communication
PK-3	Problem Solving and Trouble Shooting
PK-4	Project/Task Organization & Planning
PK-5	Collaboration and Teamwork
PK-6	Research and Self-Learning
PK-7	Multi-Task Prioritization & Management
PK-8	Relationship Management
PK-9	Analytical and Critical Thinking
PK-10	Time Management
PK-11	Quality Assurance / Control
PK-12	Mathematics and Statistics
PK-13	Ethical Intercultural Perspectives

necessarily imply that this knowledge (and skill) can be observed being applied at different levels of performance, or mastery/skill level. This aspect of learning was included in the CS2013 work, and served as a very useful key in developing competency statements.

This raises a key point. The need for competency with professional and foundational knowledge is a well-identified problem in computing education [8, 21, 26, 44, 67]. One of the key factors is distinguishing these as proper knowledge areas vs. 'skills' or recognizing that some of the desirable competencies of CS graduates are not a knoweldge area per-se, but are more properly modeled as competencies. For example, 'leadership' is not included in Table 4 [67], as leadership is more properly a competency, comprising knowledge, skill-level and dispositions and is observable in a proper context.

3.2.3 Mastery Levels in CS2013. Consistent with effective learning outcome usage, CS2013 included a definition of, and significant application of mastery levels. Learning outcomes were defined for each of the 119 KUs defined. Appropriately, each of the learning outcomes (LOs) defined in CS2013 also included the assertion of the mastery level that was expected of students with respect to that LO. Table 3 depicts the mastery levels defined in CS2013 and the corresponding levels from the revised Bloom's taxonomy that were used.

These mastery levels map well to other forms of skill assessment, most notably the cognitive domain of Bloom's taxonomy [9]. This taxonomy, developed by Benjamin S. Bloom and his collaborators in 1956, was intended to classify learning objectives and skills into levels of complexity with the purpose of structuring educational objectives and goals. Since then, Bloom's taxonomy has been used by teachers, instructors and professors in K-12, college and university education. This type of skill level vocabulary is essential to competency modeling.

3.2.4 Skill Level Vocabulary for Computer Science Competencies. Bloom's taxonomy encompasses three domains: affective (emotion-based), cognitive (knowledge-based) and sensory (action-based);

however, the cognitive domain was the most used in formal education. Within the cognitive domain, the six cognitive-learning levels are: knowledge, comprehension, application, analysis, synthesis and evaluation [9]. The six categories were described with thinking skills evolving from low- to high-level. Anderson and Krathwohl [3] identified nineteen specific cognitive processes that clarify the six categories: remember (recognizing and recalling), understand (interpreting, exemplifying, classifying, summarizing, inferring, comparing and explaining), apply (executing and implementing), analyze (differentiating, organizing and attributing) and create (generating, planning and producing).

This work applies a simplification (from the CoLEaF model) of the K-S-D-C competency theory [25, 67]: rather than distinguish conceptual knowledge (know-of/know-about) from procedural knowledge (know-how), the CC2020 approach models knowledge and skill as applied knowledge/skill (KS). This follows from the nature of competency modelling and assessment, in the sense that in developing competencies, all conceptual knowledge (know-of/know-about) and procedural knowledge (know-how) is observed as that knowledge is applied in the context of the task specified in the competency statement.

This paired modelling of knowledge/skill (KS) from the CoLeaF model as *applied* knowledge demonstrated with some skill is well suited for competency-based education. It permits connecting identified KS elements to useful observed skill levels (SLs). As in their application to well-stated learning outcomes, the revised Bloom's Cognitive domain [3] vocabulary for cognitive learning levels [9] works well for SLs, as it is in common use among computing educators. Table 5 presents the set of cognitive skill terms used in our work.

Following the typical ordering, this taxonomy for cognitive skill levels is applied to competency statement development. This approach has an added benefit in that it reduces the difficulty of distinguishing knowledge and skill components from each other (which would otherwise be required for specification), enabling a richer vocabulary of knowledge goals to be leveraged and established understandings of cognitive skill levels to be employed in the definition and assessment of competency or competency component. Consequently, this work takes the position that when expressing learning goals and/or their components, knowledge and skill normatively act as paired terms, either explicitly or implicitly. Following the pairing model depicted in Figures 3, we identify the intended skill-level in the application of that knowledge.

3.2.5 Disposition Vocabulary for Computer Science Competencies. Dispositions can be defined as patterns of behaviours that are exhibited frequently and intentionally, not forced [25, 47, 53, 54]. Dispositions express themselves in acting, reasoning and feeling in certain ways. These ways of acting, reasoning and feeling are not items that can be assessed directly; also, single action or reflected reasoning or feeling does not make up a disposition; rather the pattern is observable, albeit indirectly. In this sense, disposition can be observed through collections of actions or reflections (what they say or were observed to do), but not assessed directly [22, 26, 54]. This is a helpful observation for distinguishing dispositional elements from cognitive skills.

In curricula, the concept of disposition observable in a task presents the opportunity to enhance the comprehension of knowledge and skills as they relate to a computing discipline or academic program. Competency statements also offer an opportunity for students to realize more synthesis in their computing education. Applying relevant dispositions informs the students' educational experience by providing an approach explicit in purpose to the content they learn. Table 6 depicts disposition categories derived from relevant job descriptions [25].

These three sets of vocabulary terms are used in forming a competency specification. This is accomplished by using vocabulary terms to meta-tag portions of a competency statement by making explicit the {KS, SL, D, T} descriptive mechanism. Their use aims to expose the intended skill-level of the applied knowledge, to identify the contextual/task-related aspects of the statement, and more particularly, to make more explicit the dispositional aspects expected of the learner.

3.3 Competency Specification

An effective specification of competency is a hybrid of natural prose and component structure that enumerates the meaning of the statement. As a vernacular expression of a competency, the competency statement succinctly conveys the essential intention of curricular details. These statements express a model of knowledge that is skillfully and professionally applied in some task execution. These are free-form vernacular statements that express the competency in terms familiar and comprehensible to a wide audience, typically using a vocabulary familiar to, or that resonates with, that audience.

While the vernacular favors a public audience, the competency component structure is more formal as it enumerates the components, e.g., knowledge elements demonstrated at a skill level and moderating dispositions determined necessary for effective accomplishment of the competency in task. This structural enumeration of components is essential for automating comparative analyses and visualization of curricula. Having both a vernacular statement of the competency alongside the more formal component-specific enumeration corroborates that the two perspectives align. Any divergence perceived in these perspectives would encourage a closer reflection upon the efficacy of one or both representations.

The CC2020 K+S+D+T competency model establishes a framework designed to comprehensively describe criteria to support both understanding curricular subject matter in pedagogy and the requisites of the workplace from which the subject matter emanates. A competency statement is a vernacular expression of the modeled competency that more explicitly expresses the component structure.

In their most simple form, a singular (atomic) competency statement might address the goals for a solitary course or even an individual module within a course, constructed from suitable K+S+D+T components [67]. The KS, SL and D vocabulary provides a structure for analysing competency statements. A particular competency statement can have a number of KS components with paired SLs. Similarly, one or more KS-SL pairs can be moderated by one or more D labels. This conceptualisation adopted for CC2020 extends the CoLeaF model. This implies a relationship structure for the components (tagged phrases) of a competency statement. Figure 4

Table 5: Sample CC2020 Skill Level Vocabulary [3]

	I Remembering	II Understanding	III Applying	IV Analyzing	V Evaluating	VI Creating
Definitions	Exhibit memory of previously learned materials…	Demonstrate understanding of facts and ideas…	Solve problems in new situations by applying…	Examine and break information into parts…	Present and defend opinions by making judgments about information…	Compile information together in a new way…
Verbs	Choose, define, find, how, label, list, match, name, omit, recall, relate, select, show, spell, tell, what, when, where, which, who, why	Classify, compare, contrast, demonstrate, explain, extend, illustrate, infer, interpret, outline, relate, rephrase, show, summarize, translate	Apply, build, choose, construct, develop, experiment with, identify, interview, make use of, model, organize, plan, select, solve, utilize	Analyze, assume, categorize, classify, compare, contrast, discover, dissect, distinguish, divide, examine, function, infer, inspect, list, relate, simplify, survey, take part in, test for	Agree, appraise, assess, award, choose, conclude, criteria, criticize, decide, deduct, defend, determine, disprove, estimate, evaluate, explain, interpret, judge, justify, measure, prioritize, prove, rate, recommend, select	Adapt, build, change, combine, compose, construct, create, design, develop, elaborate, estimate, improve, invent, make up, maximize, minimize, modify, optimize, originate, plan, predict, propose solution, solve, test theory

depicts a block diagram for this tagged information for a single (atomic) competency.

The structure of this information is a key to both the formulation and interpretation of a competency statement. The core of the specification is the vernacular statement, which communicates the K+S+D+T aspects of the competency. This is then associated with a set of KS pairs, and a set of dispositions that make more explicit the intent of the competency statement.

Independent of the process by which competency specifications are developed, the following example illustrates a natural-language competency statement and explains the mapping of this statement to its constituent K+S+D+T parts.

3.3.1 Example Competency Specification: Requirements Engineering. Consider this competency statement, illustrating a significant learning task related to Software/Systems Engineering:

> *Identify and document system requirements by applying a known requirements elicitation technique in work sessions with stakeholders, using facilitative skills, as a contributing member of a requirements team.*

This natural-language text can be parsed into three constituent phrases for analysis. Each of these statements implies the demonstration of a particular set of computing knowledge and skill, as well as a context, or aspects of a context in which these components of competency should be observed. The list that follows suggests one examination of the explicit and implicit K-S pairs, as well as the implied context of the statement as a whole. This work leverages the abstract vocabulary found in Appendix A of CS2013, examples of which are provided in Tables 1 and 2. Combined with the Skill-level language of Table 5, this results in the following four mappings:

(i) "Identify and document system requirements" (somewhat) explicitly expects students to be applying [B-III] use cases and behavioral models (SE-RE-1), elicitation (SE-RE-c, SE-RE-2 and SE-RE-4), eliciting data models (SE-RE-d and SE-RE-5), functional and non-functional requirements (SE-RE-e and SE-RE-6), describing functional requirements (SE-RE-a), requirements properties (SE-RE-b).

(ii) "applying a known requirements elicitation technique in work sessions with stakeholders" explicitly expects students to be applying [B-III] producing a set of requirements (SE-RE-8), using common methods (SE-RE-10), translating into natural language specifications (SE-RE-11), with requirements analysis techniques (SE-RE-g) and considering the certainty and uncertainty considerations involved (SE-RE-h).

(iii) "using facilitative skills, as a contributing member of a requirements team" explicitly expects students to be applying [B-III] the SE-RE knowledge and skills, but also leveraging aspects of the SE/Software Project Management (SE-SPM) and SE/Tools and Environments (SE-TE) knowledge areas. Particularly, the expectation is for analyzing [B-IV] collaboration and teamwork skills (SE-SPM-a and related LOs and subtopics).

Extending this to include the relevant dispositional elements (e.g., Table 6) adds an additional mapping:

(iv) In context, these behaviors are expected to be moderated by students demonstrating that they are Purposefully engaged (D-4), with Judgement (D-6) and demonstrating to themselves and each other that they can choose to operate in a Collaborative (D-8) manner.

This statement provides an example of a competency-based approach to describing a possible program or course-level goal or outcome. The statement appears complete, in that it reasonably captures all four K-S-D-T elements of a useful competency statement at a level of abstraction consistent with the vocabulary of

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	*With Initiative / Self-Starter.* Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	*Self-motivated / Self-Directed.* Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	*With Passion / Conviction.* Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	*Purposefully engaged / Purposefulness.* Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	*With Professionalism / Work ethic.* Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	*With Judgement / Discretion / Responsible / Rectitude.* Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	*Adaptable / Flexible / Agile.* Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	*Collaborative / Team Player / Influencing.* Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	*Responsive / Respectful.* Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	*Attentive to Detail.* Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	*Exploratory / Inventive.* Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

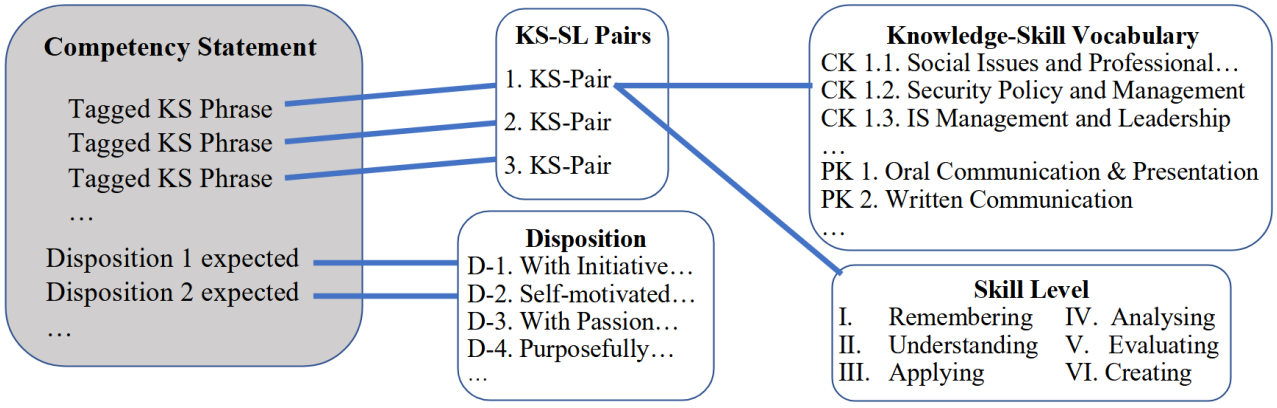


Figure 4: Knowledge, skill, disposition and task components of a competency specification [25]

interest. Assessing completeness also warrants ensuring that the task is at least implied. In this case, the statement is explicit about having a particular (though unspecified) task in which these aspects of competency in requirements elicitation are demonstrated. With the additional information elicited from this level of analysis, the statement expands into an atomic competency statement that other competencies could leverage/expect.
This competency statement focuses on a central aspect of systems (and/or) software engineering. If employed in a course or program, it sets up the opportunity for (and challenges) the educator teaching a systems and/or software requirements unit (or course) to set up a learning situation whereby the students are challenged to engage in the context, and the instructor can observe student behaviour to assess to what extent the students demonstrate the K-S-D components.

3.4 Writing Competencies and Bloom's Taxonomy Levels

The CS2013 curriculum learning outcomes are expressed based on three mastery levels (familiarity, usage and assessment) and skills that are mapped on Bloom's six taxonomy levels (B-I remembering, B-II understanding, B-III applying, B-IV analysing, B-V evaluating and B-VI creating). Upon analyzing CS2013 learning outcomes, we found that a large number of them are based on Bloom's taxonomy level B-II (understanding). This can be a problem when writing competencies, as competencies should model observable tasks in a professional context. Therefore, competencies should be written at level B-III and above, and careful attention should be given to the professional context.

Consider a practical example from the Architecture and Organization KA of the CS2013 curriculum that illustrates why B-II level learning outcomes and skills alone are not suitable for competencies expression. In a Computer Architecture and Organization course, an instructor found that students had learned the rules of 2's complement in a pre-requisite mathematics course. This instructor further found that 120 students were able to apply the rules to a sequence of binary digits and met the B-II skill level. However, when queried on the professional context of the use of binary, e.g., what was represented in base 10 by a sequence of binary digits, only 2 of the 120 students answered correctly as they did not apply the 2's complement rules in the context of data representation (students applied 2's complement rules even to positive numbers). This observation resulted in re-designing the mathematics content to be taught not just as a set of rules, but as practical examples and problem solving in the context of data representation. As this example illustrates, Bloom's Taxonomy, particularly level B-II and below, is when learners exhibit the cognitive level of understanding (B-II) and even usage in a task (B-III). This may be observable or not, but even if observable, it does not incorporate a professional context. By contrast, competencies focus on *observable* demonstration in a professional context.

3.4.1 Differentiating competency statements and learning outcomes. When comparing learning outcomes and this style of competency modelling, the use of Bloom's levels for cognitive process differs.

Learning outcomes focus on measurement, not just observation. The levels in Bloom's Taxonomy are intended to be not just observed, but assessed. In this context, a phrase like "understands" would be poor use, as measuring a learner's understanding is difficult and/or ambiguous. However, in the context of competency statements, the use of Bloom's Taxonomy is an indicator of a cognitive skill level of a particular knowledge element observed in a context; its use serves as an indicator. The purpose of detailed K-S components tagged to a competency statement is disambiguation. What is meant by a broad statement of competency, especially in natural language terms, is necessarily ambiguous. Consequently, the addition of knowledge elements that are expected to be observed/included in the demonstration of competency actually need a specification of the skill level expected [24, 25, 67].

4 PROCESS

The process adopted by the team evolved as the work progressed. Broadly speaking, we had conceived the approach as adopting an *input-process-output* model. The inputs were the CS2013 curriculum model and CC2020 conception for modeling an effective competency statement. The process involved mapping from CS2013 to CC2020 expressions of competency. The outputs would be the report indicating how the creation of competency statements operated and how they could be developed, complemented by a transformation of the competency statements into a format that would readily feed into the CC2020 prototype visualization tool, to serve as exemplars of competencies to be visualised by this prototype.

4.1 From CS2013 to CC2020

As described in the introduction and in the previous section, the CS2013 Curriculum Report provides curricular guidelines centred around Knowledge Areas (KAs), Knowledge Units (KUs), Topics, and Learning Outcomes (LOs). By contrast, CC2020 specifies competencies as a composition of *Knowledge* and *Skill* elements associated with *Disposition* elements, all within the context of performing a specific *Task*. The CS2013 Body of Knowledge provides us with Knowledge elements and associated Skill elements (through the LO levels of mastery). But it does not provide **any** of the Disposition or Task elements. Thus, mapping individual KAs or KUs to Competency Statements requires us to envision a specific professional context (the Task and Disposition elements) within which the KA/KU is demonstrated.

Having established work partners, we set to the task of writing up sample Competency Statements for individual KAs/KUs in CS2013. Each pair worked independently to generate four statements or more (a minimum of two statements for each of two KAs/KUs). Some partners were able to carry out this work synchronously; others used a divide-and-conquer approach: each partner wrote up two statements, which were then reviewed by the second partner.

4.2 Level of Granularity

When we regrouped to review and discuss our draft statements, we quickly identified key differences in each pair's work.

- Approaches ranged from capturing a small number of related topics/LOs in a statement to capturing entire KAs in a single statement or multiple statements; after discussion, we agreed

KA-Ref#	Title	Competency Statement	Dispositions	Knowledge-Skill Pairs				Traced Competencies	Notes
				KU / Broader Knowledge Description	Topic/LO Description	Knowledge Element	Paired Skill Level		
Our reference number goes here. E.g., "KA-##"	Short Title	Competency statement goes here (Natural Language) that embeds the task/context and suggests the K-S-D breadth and depth	List top applicable dispositions: <ID+descriptor>. Note that dispositions are not aligned with KS-pairs.	CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level	List other competency reference #s that this competency expects or depends upon. Often null.	Any notes or comments
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Often null
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		
				CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	New IDs for extensions to CS2013	Bloom's Level		Note deviations from Bloom's level in CS2013
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		

Figure 5: Competency specification in a tabular format

that it was most appropriate to express competencies at the level of entire KAs, from the point of view of a graduate or professional. When we "designed" a competency statement, we focused on a particular knowledge area and looked for collections of topics and/or learning outcomes (LOs) that could be observed in tasks and at particular skill levels. While seemingly straightforward, there was significant variance in how this was approached, and how the resulting competency was formulated. Dispositions were often implied at first, but refinement of the statements made these both more clear and more relevant.

- The format ranged from simple textual lists to more complex tables linking Knowledge-Skill pairs with individual Dispositions; after discussion, we agreed on a tabular format for ease of integration with the CC2020 prototype. This also aligns with an anticipated need from practitioners/curriculum designers for an easy-to-use instrument in a tabular format, and precise instructions.
- We discussed writing dispositions based on individual LOs vs. dispositions based on observable tasks that encompass several LOs. As per the CoLeaF model, we agreed that dispositions should be defined based on observable tasks.
- We argued that all eleven dispositions are important and expected for a computer science student or professional. However, not all dispositions are equally relevant for a task, and therefore we agreed to include only the most pertinent dispositions to the task at hand.
- We discussed at what level competencies should be written (e.g., module, course, speciality, or program). We concluded that competencies should be written at any level decided by practitioners/curriculum designers.
- We discussed the importance of providing the context in which tasks are carried out, in addition to expressing measurable/observable achievement.
- Context specification turned out to be highly subject to granularity issues. Over-specifying the context tended to leave the competency statement looking more like a classroom assignment; under-specifying the context/task could easily disrupt the observability of the competency in a way that made it difficult to know whether a particular instantiation of the context/task actually met the intent of the statement.

Appendix B show samples of the original statements produced by each pair. This demonstrates the wide variation we had to accommodate to come up with a common format.

4.3 Common Format

Both textual and tabular statements are generated by starting from a particular topic/LO, or a group of closely-related topics and LOs from one KU or an entire KA, imagining a concrete task where a student or professional would be expected to demonstrate the target topics/LOs, and writing up a clear informal statement of this situation. Samples of these LO and topic statements from CS2013 are provided in Tables 1 and 2 (Section 3).

Using the informal statement as a starting point, we then review other topics and LOs (mostly in the same KA but this could also cross KA boundaries) to identify the ones most relevant to the statement and generate a list of Knowledge-Skill pairs. In these pairs, the topic/LO constitute the Knowledge component, the level at which it is involved constitute the Skill component—note that this level may differ from the one indicated in CS2013, most often by being higher up in Bloom's hierarchy. After all, while a topic in CS2013 might be phrased in terms of students "being aware" of a particular topic, genuine situations where students need to demonstrate that awareness will most often involve some sort of application (and therefore be at least at Bloom's Application level). Disposition elements are then added for the entire statement; they are not tied to specific Knowledge-Skill pairs because they apply to the overall situation being described.

After much discussion, we settled on the tabular format as being the most appropriate: it encapsulates every key element that makes up a competency in a compact form, and it is especially well-suited for automated processing—in particular, by the CC2020 prototype system. Figure 5 presents the abstract format we settled on for tabular competency specifications. Section 5 contains a detailed description of the final format and the sample competency statements obtained.

4.4 Validation Methodology

To develop a common language and follow a consistent and robust process when defining our competencies, we used a two-step validation strategy. Following our first attempt at producing draft competency statements (described in the preceding sub-sections),

individual drafts were subjected to a thorough review. First, a rubric was agreed upon (see sub-section 4.4.2) to evaluate the consistency and validity of each statement. Next, each pair of original authors reviewed their statements (*self-evaluation*), and larger groups were formed from two pairs to validate each other's statements (*peer-validation*).

4.4.1 Competency Statement Self-Evaluation. Each pair of original authors reviewed their competencies and re-wrote their statements. Each statement was reviewed to identify issues, undertake revisions, and consider any limitations found for inclusion in the discussion section. The revision process had side-effects besides expressing competencies in a more robust and complete format. In many cases, the authors identified and included more statements. In addition, the process of self-evaluation resulted in the following improvements.

- The most common improvement was to make the dispositions expected in the competency more explicit within the statement. For example, the first drafts of the IS, SE and SP statements used more implicit language to indicate the dispositional elements expected.
- Some authors identified and included more competency statements as a direct result of their revisions. For example, including more LOs missed in the initial pass, or pulling in topics or LOs from other KAs that were clearly required to support the desired competency demonstration. This often included expanding a single-KA statement to include aspects of other KAs.
- Context improvement: we discussed how to express the context when writing competencies that apply to topics taught at different levels and/or using different technologies, such as an HCI course taught using mobile technologies vs. one using web development as the vehicle of instruction. This is illustrated in the differences between statements HCI-3 and HCI-3+ listed in Section 5. We found that the context should be neither over- nor under-specified, but remain sufficiently open. This has the merit that it allows for future formulations of competencies in a more specific context to a particular program or course, and the drawback that the formulation could be perceived as too general and vague.
- We found that some competencies overlap (e.g., CS-SE-3 parallels CS-SE-1 to a certain extent). This led to a discussion of the way in which some competencies could subsume others. While the CoLeaF model allows for competencies to form a hierarchy (where some competencies are components of others), we consider that some amount of overlap is nonetheless acceptable and needed as competencies could map differently depending on the particularities of a program, speciality or course.
- The level of the competency statements and the context in which these statements are employed became a point of discussion. For example, numerous statements seemed closely associated with a particular instantiation of a particular course (e.g., HCI-3, several of the OS statements). This was most obvious as statements incorporated skill levels less than three. Similarly, the statement could be at a program level, where the skill levels were above, or routinely above, level four.

4.4.2 Competency Statement Peer-Validation. Within the sub-groups of four, each statement was reviewed by non-author peers. This step aimed to develop team norms within the sub-group about the statements, and to establish a common approach and coverage. It is interesting that the sub-groups report that, even if peer validation resulted in valuable improvement of the competency statements, there were more similarities than differences of opinion within the sub-group. This gave us more confidence that the process we employed is effective in developing a common language.
Our peer validation involved developing a rubric around sets of questions that could be assessed for the competency specification, examining both the free-form statement and the mapping to its constituent K-S-D components.

- Comprehensible & Clear? {Yes | Maybe | No}
- Context provided? {Yes | Implied | No}
- Observable achievement? {Yes | Maybe | No}
- Embodies Dispositions? {Yes | Implied | No}
- Incorporates Professional KAs? {Yes | Implied | No}
- Good Coverage of KA/KU Content? {Yes | Maybe | No}
- Effective mapping of statement to connected K+S+D? {Yes | Maybe | No}
- Level of CS2013 Competency Statement {Program | Multiple KA | Single KA}
- KU Count {number}
- Topic Count {number}
- LO Count {number}

Other information collected from the review included *Notes specific to this statement*, *Who conducted the review and when*, *General thoughts about the authoring process* and *Suggested rewrite* information. Some, but not all statements were rewritten following review.

4.4.3 Validation Examples. Of the 36 competency statements subjected to review, four have been included in an appendix to illustrate the validation process. These four examples highlight different aspects of the statement writing quality, and of the review process itself, that contributed to our lessons learned. The four competency statements {AR-1, HCI-3, IS-4 and OS-4} are listed below, and the detailed breakdown for each of them is available in Section 3. AR-1 is presented because it is representative of a situation where the distinction between Dispositions and Professional & Foundational Knowledge Areas is particularly relevant. The findings around HCI-3 led to some good comments and discussions about what a single competency statement is not intended to capture, and how the hierarchical structure of competencies really needs to be taken advantage of better in our modelling, to allow single competencies to be more focused. Similarly, OS-4 was quite interesting because it shows a very narrowly-focused, low-level competency yet it brings up a question that the team often struggled with: should dispositions always be worked into the vernacular statement (at the risk of making these statements sound slightly less natural), or is it fine to leave them implicit in the vernacular statement because they are spelled out in the rest of the table (just like the precise K-S elements are not all spelled out in the statement but are listed separately)? IS-4 was selected as one of the few competencies to include explicit Professional & Foundational Knowledge Areas in the competency statement.

Table 7: IS-4: IS/Basic Machine Learning

Statement	Dispositions
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.	D-6 (Responsible) D-10 (Meticulous)

Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
IS/Basic Machine Learning	List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised.	IS-BML-1	B-II	
	Identify examples of classification tasks, including the available input features and output to be predicted	IS-BML-2	B-II	
	Explain the difference between inductive and deductive learning	IS-BML-3	B-II	
	Describe over-fitting in the context of a problem	IS-BML-4	B-II	
	Apply the simple statistical learning algorithm such as Naive Bayesian Classifier to a classification task and measure the classifier's accuracy	IS-BML-5	B-III	
Prof. & Foundational Knowledge	Analytical and Critical Thinking	PK-9	B-III	
	Mathematics and Statistics	PK-12	B-III	
IS/Basic Machine Learning	Definition and examples of broad variety of machine learning tasks, including classification	IS-BML-a	B-II	
	Inductive Learning	IS-BML-b	B-II	
	Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees	IS-BML-c	B-III	
	The over-fitting problem	IS-BML-d	B-II	
	Measuring classifier accuracy	IS-BML-e	B-III	

AR-1 For a given modern pipelined computer architecture, coherently explain in sufficient and correct detail 1) the execution of a sequence of code at the instruction-level (fetch, decode, execute, fetch operands, save results), 2) the bus transfer of instruction and data from and to cache and main memory, 3) the data integrity (e.g., error-correction), and 4) the impact on/use of memory organization. [This statement's full specification is presented in Table 22.]

HCI-3 As a small independent development team, leverage a wireframe UI design to design, develop, test and evaluate a prototype interactive mobile application. Contrast the mobile development process to that for non-mobile applications. [This statement's full specification is presented in Table 24.]

IS-4 Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier. [This statement's full specification is presented in Table 7.]

OS-4 Explain the abstractions involved and the resources used to print a text file on a printer. [This statement's full specification is presented in Table 21.]

4.4.4 Validation Findings. The validation results for these four statements, presented in Table 8, show one single- and three multi-KA statements. However, this is not representative of the 36 statements, where 27 of the 36 were single-KA focused, while only nine were multiple-KA focused. This discrepancy corresponds with our observation that the multiple-KA competency statements were both more interesting and more challenging to write.

The following notes were specific to the four sample statements as written:

AR-1 Grammar is tough. Numbered items helps. This seems to rely on CS-SP-PC skills. Added these connections and adjusted the dispositional phrases.

HCI-3 This competency is targeted to a particular course and consequently easily over-specified. [The last sentence (contrast…) may be perceived as a bit of an add-on: the implicit task described by the first sentence is already complete, so the second sentence gives a slight "kitchen sink" feeling: "while we're at it, we should add that the student also contrasts mobile and non-mobile development, oh and they could also comment on their choice of data structures and algorithms, and …" This raises the question of which task elements are central to the competency being described and which are peripheral. We currently don't have a good mechanism for making these sorts of distinctions.]

IS-4 Broad coverage of the full KU, explicit about dispositions, context provided by problem addressed. Term "abstractions" maybe interpreted in different ways.

OS-4 Dispositions appear not to be embodied in the competency statement.

Table 8: Validation Summary for Four Competency Statements

KA-Ref	Short Title	Observable?	Dispositions?	KS Pairs?	Professional KA?	Good Coverage?	Effective mapping?	Level	KU Count	Topic Count	LO Count
AR-1	Instruction Execution	Y	Y	Y	I	Y	Y	M	3	16	12
HCI-3	Interactive Systems Development	Y	I	Y	I	Y	Y	M	8	7	24
IS-4	IS/Basic Machine Learning	Y	Y	Y	Y	Y	Y	M	2	5	7
OS-4	Operating Systems Principles	Y	N	Y	N	Y	Y	S	1	4	4

One of the differences that these four statements presented was a variance in completeness. Completeness here is with respect to the CoLeaF model, where Task, Disposition and Knowledge-Skill pairs are expected. Of these four, the OS-4 statement did not clearly identify dispositional elements, whereas AR-1 and IS-4 explicitly did. The absence of dispositional terms raised suggestions for how to improve statement OS-4, as indicated in the general comments that follow. The general comments that these four reviews generated included:

- AR-1 Considered adding a new disposition category ("Articulate") more suitable for B-II-level LOs.
- IS-4 Word-for-word formulations of competencies are necessarily problematic. These are intended for adaptation. Over-specification is a significant risk in competency authoring.
- HCI-3 Assessments and skill levels would need to be tiered for students at different stages of the degree.
- OS-4 Should we always use “dispositional verbs”? One answer is: yes, at least until the ideas of competency statements stabilise. For example, restating this competency statement somewhat like: “Meticulously list the abstractions . . . ” could work better. The context here is quite narrow in the sense that it focuses on the technical aspects only and does not seem to involve professional knowledge areas, e.g., shared printers. Do we always need to deal with professional knowledge?

Another area of differentiation is the inclusion of Professional KAs called out in CC2020. While these are part of the SE and SP knowledge areas in CS2013, the legitimate question is how and where these should be called out explicitly, as they were in IS-4 and implicit in the AR-1 and HCI-3 statements. Note that these three statements were also among the nine multi-KA statements. Similarly among the 36 sample statements that this team reviewed in detail, only 9 (all single-KA) statements did not imply demonstration of the Professional KA. This was explicitly called out as one of the more general questions/concerns raised with OS-4.

The discussions and lessons learned surrounding HCI-3 were particularly useful. What we found was a situation where the statement authors were working in a vacuum, which leads to a tendency to write slightly "bloated" competencies. As the work progressed, a major contributor was that the author team did not have access to simpler, lower-level competencies that we should be able to depend on. This showed up in other competencies, particularly those that appear more integrative such as the Software Engineering (SE) competencies. This issue of dependency was itself dependent on the instantiation of the task context; dependency of one competency on another implies (requires?) a particular ordering of learning/experience. While this ordering is easy to observe in the instantiation of a particular computer science degree program, it is less easy to make assumptions about ordering of material and experiences in the broader, general case. The result was competencies with large numbers of dependent KUs, or a specification that assumes a particular ordering of tasks/competency. The former makes specifying a complete competency more difficult, while the latter makes the resulting collection of competencies less able to be reused in another context. This raises questions about the ordering of material in computer science programs: what competencies should come first / must come first?

For example, in Table 24, sets of Software Engineering (SE) topics and learning outcomes are traced to the statement. Are they all required for that context? To the extent that they are not required to be demonstrated to meet the intent of the statement, then they over-specify the competency. On the other hand some of these (such as SE-SD-11 ‘apply patterns’ and SE-SD-13 ‘select components’) are very reasonable and appropriate extensions to the other traced topics/LOs (e.g., HCI-PIS-1 ‘explain patterns’, HCI-PIS-2 ‘create modern GUI’, etc.). Others, such as the Software Verification & Validation (SVV) may (as the comments specific to HCI-3 above suggest) be an add-on. As we discovered in these validation attempts, the distinction between bloat/over-specification and synergy among KA’s in a context was not always obvious.

The HCI-3 example raised other questions in validation. A similar point of KA overlap arose over the inclusion of the Platform Based Development (PBD) KA. The context of HCI implementation is normatively on some platform. Which platform? This same competency could be re-written to be more generic (see HCI-3+) and then not be able to specifically link the relevant PD knowledge unit (such as web platforms, industrial platforms or game platforms specified in CS2013), or as in this example, be contextualized through its task specification to a specific to a particular knowledge unit (in this case PBD-MP). Again, from the perspective of the author of competencies for a particular program, the latter case is probably more appropriate. From the perspective of CoLeaF or developing a replacement for CS2013 using competencies, probably a mechanism to allow conditional specification of knowledge to a particular

competency specification (e.g., to better communicate the intent of HCI-3+) would be warranted. This and other possible extensions to competency modeling are left for future work.

As a whole, the Intelligent Systems KA came out very clean, and was among the best examples generated. The IS-4 example was very interesting as it was clearly improved by the addition of the Professional and Foundational Knowledge Areas listed in Table 4. This example also showed success in the struggle to balance the statement of the competency to most accurately hit the Tier-1 and Tier-2 mastery levels of CS2013.

5 SAMPLE COMPETENCY STATEMENTS

In this section, we provide sample competency statements developed for Knowledge Areas in CS2013. At the time of writing this report, 43 sample competency statements have been developed spanning 10 of the 18 KAs defined in CS2013 (AL, AR, HCI, IS, NC, OS, PL, SDF, SE and SP). Note that we did not attempt to ensure either breadth or depth of coverage of the CS2013 KAs. As mentioned earlier, our efforts were focused on the *process* of developing competency statements, rather than on the resulting statements themselves. Once the initial draft statements were ready, significant work was required for validation and to reflect on our process. Unfortunately, this did not leave time for us to "fill in the gaps" and cover more KAs, or try to ensure KA coverage proportional to the size of the KA (in terms of number of topics and/or LOs).

Following the pattern provided in Figure 5, these statements have each had competency specifications developed [25, 67], samples of which are provided in Appendix D. Additionally, other draft competency statements were developed covering aspects of the DS, NC and PL KAs, but we were unable to complete quality review on these contributions. These are included in Appendix C. As part of the broader effort to understand how the LO/Topic approach to curricular definition (like CS2013) maps to a competency-based approach, the 43 statements from the sample KAs have been traced to the topics and learning outcomes for completeness/coverage.

Table 9: Summary of CS2013 KAs Addressed by 43 Competency Specifications

Code	Knowledge Area	Statement Count	Tiers 1 & 2 KUs
AL	Algorithms	8	6
AR	Architecture and Organization	2	7
HCI	Human Computer Interaction	4	10
IS	Intelligent Systems	4	11
NC	Networking & Communications	1	7
OS	Operating Systems	6	6
PL	Programming Languages	6	6
SDF	Software Development Fundamentals	2	4
SE	Software Engineering	4	6
SP	Social Issues and Professional Practice	4	7
		43	100

Table 9 lists the different CS2013 KAs that were addressed, and how many competency statements were specified. This also shows the number of KUs for which there were one or more Tier-1 and Tier-2 topics and/or learning outcomes defined. These topics and LOs were the initial coverage target, as these represent the minimum learning expected of a CS2013-compliant degree program. The potential 87 Tier-1 and Tier-2 KUs represent 68% of the 127 Tier-1 and Tier-2 KUs in the CS2013.

For a more specific example, Table 10 illustrates the coverage that the complete specification for the two HCI competencies provide. These two statements are intended to provide complete coverage for 10/10 topics and 4/5 LOs defined for the HCI-F (Foundations) KU. Additionally, they provide coverage for 8/8 topics and 1/4 LOs defined for the HCI-DI (Designing Interaction) KU. No coverage is provided for topics or LOs from the other eight KUs documented in the HCI KA. At first glance this assessment appears to be very lax; only two of ten KUs are addressed. However, this is misleading, as only these two KUs (HCI-F and HCI-DI) have any Tier-1 or Tier-2 expectations. All other eight KUs are identified as elective topics. Consequently, these two statements provide near-complete coverage of HCI/Foundations and some coverage of the HCI/Designing Interaction KUs. In total, four statements addressed 24 of 27 tier-1 and tier-2 topics and LOs for the HCI Knowledge Area.

5.1 The Competency Statements

The 43 competency statements (not the full specifications) are listed in the subsections that follow. These subsections list each of the free-form text statement of the preliminary competencies developed with that KA in mind. Note that more statements were developed; these 43 sample statements were subject to quality review, and many were improved in response to the issues uncovered.

5.1.1 AL—Algorithms and Complexity.

AL-1 Given a problem to program for which an inefficient solution is not sufficient, research and implement a solution that is sufficient for reasonable instances, without straining other resources or beeing to complex to implement.

AL-2 Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior.

AL-3 Given a problem that involves multiple actions (such as insertions, deletions, and searches) that may or may not have predictable occurrences, properties, or relationships, devise a data structure that optimally supports the actions.

AL-4 Present to an audience of co-workers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops.

AL-5 Given partial specifications for a program that is to include a small user-interface language, devise a syntax that accommodates the expressiveness of the interface, is simple to use and not prone to user mistakes, and can be implemented using a small finite state machine solution. Present the syntax to the users as regular expressions.

AL-6 Prepare a presentation that introduces first year students to the basic concepts of algorithmic complexity, such as

Table 10: Topic and LO Coverage for three HCI Competency Specifications

KU Code	Knowledge Unit	KU		Coverage	
		Topics	LOs	Topic	LO
HCI-F	HCI/Foundations	10	5	10	5
HCI-DI	HCI/Designing Interaction	8	4	8	1
HCI-PIS	HCI/Programming Interactive Systems	12	4	0	0
HCI-UCDT	HCI/User-Centered Design & Testing	4	5	0	0
HCI-NIT	HCI/New Interactive Technologies	6	3	0	0
HCI-CC	HCI/Collaboration & Communication	7	4	0	0
HCI-SM	HCI/Statistical Methods for HCI	7	4	0	0
HCI-HFS	HCI/Human Factors & Security	9	5	0	0
HCI-DO	HCI/Design-Oriented HCI	11	3	0	0
HCI-MAVR	HCI/Mixed, Augmented and Virtual Reality	22	6	0	0

notation, characteristics, complexity classes, time and space, empirical measurement, and impact on practical problems.

AL-7 After watching algorithms' animations, provide in natural language, different descriptions, starting from the high-level idea and the algorithmic strategy down to details covering implementation, execution time, and the underlying mathematical model.

AL-8 In different context from pre-university to undergraduate and graduate level up to job interviews, apply a algorithm design strategies to solve an assigned puzzle and prove the correctness and eventually the optimality of your solutions.

5.1.2 AR—Architecture and Organization.

AR-1 For a given modern pipelined computer architecture, coherently explain in sufficient and correct detail 1) the execution of a sequence of code at the instruction-level (fetch, decode, execute, fetch operands, save results), 2) the bus transfer of instruction and data from and to cache and main memory, 3) the data integrity (e.g., error-correction), and 4) the impact on/use of memory organization.

AR-2 Provide a responsible comparison/contrasting of the differences between/among two or more computer architectures that coherently describes the equivalent representations of computer functionality, instruction sequences/code (e.g., instructions, addresses) and data representation (e.g., numbers, characters, data structures, images, audio files, etc.) across multiple levels of abstraction (mathematical, digital-logic, machine-code, assembly code, HOL, etc.).

5.1.3 CN—Computational Science. None developed as part of this report. Left for future work.

5.1.4 DS—Discrete Structures. Preliminary statements were developed but not subject to review. See Appendix C.

5.1.5 GV—Graphics and Visualization. None developed as part of this report. Left for future work.

5.1.6 HCI—Human-Computer Interaction.

HCI-1 Purposefully follow a user-centered design process and adaptively engage users to develop and document an analysis of the user needs for a human-computer interaction with a proposed computing application, device or system.

HCI-2 Work independently and rigorously adhere to a user-centered design process, adaptively engaging non-peer real users in the development of an interactive application.

HCI-3 As a small independent development team, leverage a wireframe UI design to design, develop, test and evaluate a prototype interactive mobile application. Contrast the mobile development process to that for non-mobile applications.

HCI-3+ As a small independent development team, leverage a wireframe UI design to design, develop, test and evaluate a prototype interactive application for a particular interactive computing platform[5]. Contrast the interactive development process to other development.

5.1.7 IAS—Information Assurance and Security. None developed as part of this report. Left for future work.

5.1.8 IM—Information Management. None developed as part of this report. Left for future work.

5.1.9 IS—Intelligent Systems. Additional to the CS2013 topics and LOs mapped, these competency statements were explicitly mapped to CS2020 PK KAs (Table 4).

IS-1 Demonstrate commitment and insight in showing how awareness of ethical applications of AI and the nature of intelligent behavior and reasoning, support a professional attitude to determining whether an AI approach is appropriate to a given problem, and justifying or otherwise its merits.

IS-2 Show the ability to independently and carefully frame, select, implement and judge the most appropriate approach by distinguishing both suitable algorithms to meet the goals set in a given search problem and balance the trade-offs of completeness, optimality, time complexity and space complexity.

IS-3 Show care in flexibly constructing differing representations of knowledge, while applying multiple approaches to logical reasoning to draw well evidenced conclusions.

IS-4 Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned

[5]This variant of HCI-3 provides more ambiguity to the platform, which would necessarily change the platform topics/LOs that would be traced.

classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.
(A detailed expansion showing how this competency statements maps to its specification is provided in Table 7, in Section 4.)

5.1.10 NC—Networking and Communications.

NC-1 Analyse and propose approaches to dealing with the different situations that arise in managing data transfer and communication over a lossy medium or channel based on contextually relevant quality parameters and mindful of performance priorities and desired outcomes.

5.1.11 OS—Operating Systems.

OS-1 Given an unfamiliar OS proactively find descriptive and functional information about it. Descriptive information captures adjectives like networked OS while functional information is about convenience features, security issues etc.
OS-2 Contrast the logical layering strategies for different OS architectures.
OS-3 List the steps to obtain two processes out of a single program executable image for two different OSes.
OS-4 Explain the abstractions involved and the resources used to print a text file on a printer.
OS-5 Clearly demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.
OS-6 Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and problems beyond computing.
OS-7 Design a memory usage, optionally including caching strategy, over all the available memory hierarchies for various applications (e.g. web application).
OS-8 Carry out simple system administration tasks according to a security policy, for example creating accounts, setting permissions, applying patches, and arranging for regular backups (cross-reference IAS/Security Architecture and Systems Administration).

5.1.12 PBD—Platform-based Development. No competency statements were developed explicitly as part of this report. Some PBD topics and LOs were included in HCI Competency Statements. See statement HCI-3 above.

5.1.13 PD—Parallel and Distributed Computing.

PL-1 Correctly demonstrate understanding of the basic components of object-oriented programming by writing a code segment with properties including state, behavior, class hierarchy, inheritance, fields, encapsulation, methods, constructors, iteration, and dynamic dispatch.
PL-2 Correctly demonstrate understanding of functional programming by writing a code segment with properties including taking and returning functions, function closures, functional encapsulation, and iteration.
PL-3 Accurately and effectively write a code segment which demonstrates all the concepts of event-driven and reactive programming.
PL-4 Correctly demonstrate understanding basic type systems by writing a code segment which correctly uses typing.
PL-5 Correctly demonstrate understanding of program representation options by writing a small program to process some representation of code for some specific purpose, such as an interpreter.
PL-6 Correctly demonstrate understanding of language translation and execution by writing a program to compile, interpret, or translate a small program.

5.1.14 PL—Programming Languages. Preliminary statements were developed but not subject to review. See Appendix C.

5.1.15 SDF—Software Development Fundamentals.

SDF-1 Consider various algorithms to solve a problem, along with their key properties, and select the most appropriate algorithm based on careful consideration of their strengths and weaknesses.
SDF-2 Carefully use appropriate, standard data structures (such as arrays, linked lists, strings, stacks, queues, sets, maps) to write a program that carries out a well-specified task correctly and efficiently.

5.1.16 SE—Software Engineering.

SE-1 As a team, purposefully develop a project proposal document for a team software development/enhancement project based on requirements; responsibly estimate the team development tasks and effort as well as the technical, development and security risks anticipated with meticulous attention to quality.
SE-2 As a team, purposefully follow a software engineering methodology; proactively select and responsibly use the appropriate tools and demonstrate good practices for designing, developing, integrating and testing a project according to a documented proposal, requirement documents or statement of work.
SE-3 Identify and document system requirements by applying a known requirements elicitation technique in work sessions with stakeholders, using facilitative skills, as a contributing member of a requirements team.
SE-4 Responsibly leveraging a suitable requirements management process, responsively elicit, document, verify and negotiate the requirements for a software or systems development project. Professionally communicate the process and results of the effort.

5.1.17 SF—Systems Fundamentals. None developed as part of this report. Left for future work.

5.1.18 SP—Social Issues and Professional Practice.

SP-1 Using empirical data with judgement and ethically focused approach evaluate a given (software/hardware) design and its implementation using criteria such as:
(a) developers' assumptions and values embedded in the design
(b) design usability for diverse population (including underrepresented population and the disabled)
(c) under-representation of diverse population in the computing profession

SP-2 For a current computing issue with global implications, leveraging professional codes of ethics (e.g., ACM, IEEE Computer Society, and others), develop an insightful analysis of the role(s) of the community, professionals and government(s) for addressing this issue.

SP-3 Collaboratively design and deploy software or hardware using sustainable design guidelines and make a critical prediction of local or global social and environmental impact of this software or hardware (use and e-waste disposal).

SP-4 With attention to detail, research the state of art and predict future possible usage of pervasive computing in promotion of sustainable behaviour in the following areas: smart energy systems, social networking, transportation, agriculture, supply-chain systems, environmental monitoring and citizen activism. Base on you critical analysis and guidelines for sustainable design standards make a computational design that foster sustainable behaviour in one area.

5.2 Preliminary Analysis

Initially, 32 sample competency statements were developed for eleven KAs (AL, AR, DS, HCI, IS, NS, OS, PL, SDF, SE and SP). Further reviews, quality assessment and statement development continued for eight of these KAs, as well as the inclusion of the CC2020 Professional & Foundational Knowledge elements outlined in Table 4. From the second round of statement analysis and refinement, 43 competency statements were developed addressing the AL, AR, HCI, IS, NC, OS, PL, SDF, SE and SP CS2013 KAs and the CC2020-PK knowledge area. Table 11 depicts how these ten KAs were 'covered' by the 43 competency statements authored. Text and summary information for the original competency statements drafted for the 11 KAs is presented in Appendix C.

For the 43 competency specifications summarized in Table 11, the KU coverage was computed as the sum of the Tier-1 and Tier-2 topics and Learning Outcomes for that KA, divided by the sum of the Tier-1 and Tier-2 topics and LOs that were addressed by one or more competency statements for that KA. The Topic/LO Coverage is shown as that ratio expressed as a percentage (%). The last column of Table 11 shows the ratio of topics & LOs per statement, which provides an overview of the complexity of the competency statements for that KA. Lower ratios indicate that the competency statement focused on fewer CS2013 Topics/LOs. Higher ratios indicate that the statement has higher/broader expectations and or a higher expectation for the synthesis of knowledge in the demonstration of that competency.

As Table 11 suggests, the competency statements developed for Programming Languages (PL) did not include tracing to the related topics; this shows that a different level of granularity was used by these authors who only specified LOs. This differed from authors of the other KAs who traced both topics and LOs. Consequently the coverage for these statements appears significantly lower (26%) than most other KAs.

As a whole, these competency statements represent ten of eighteen KAs, and 50% of the Tier-1 and Tier-2 topics defined in the CS2013 for these ten KAs. At a glance, these 43 statements represent over 300 Topics and LOs, or roughly a seven-fold reduction in the amount of information conveyed. The notion of coverage, and the relationship of competency statements to each other and to the underlying knowledge and skills expected of a program are points of expansion that we leave to future work (See the next section).

6 DISCUSSION

Formulating competencies which embody the expected learning outcomes (LOs) of earlier ACM/IEEE curricula requires a combination of interpretation and expert judgement. Our point of departure as a working group was to explore how the CS2013 curriculum, in combination with our professional and disciplinary insight, might be amenable to re-expression. The goal was to transform the knowledge-and-learning-outcomes-based CS2013 curriculum into formulations of expectations of graduate proficiency expressed in terms of competencies.

One key output was, thus, to develop a set of steps (a process) illustrating how existing curricula might be recast in terms of a learning-outcomes based collection of competencies. This section offers these steps and summarises the lessons learned in that process. It seems likely that similar efforts in other fields and sub-fields of computing will directly benefit from these insights. We hope that the processes and quality assurance rubrics developed in this project will serve the profession well, as communities of educators in the computing sub-disciplines engage in related exercises in the future.

6.1 Steps for Re-expressing LO-based Curricula in terms of Competencies

Our recommendation, based on the work described here, is that the following general process provides good support for re-expressing LO-based approaches in terms of competencies.

- Start by identifying the knowledge areas. Often these would be identified during the development of the LO-based approach. It is best to use generally accepted vocabulary, such as that used in the CS2013 report.
- Use the LO itself to identify the skill level. See the discussion below.
- Developing the right set of dispositions is most easily accomplished by drawing on the expert judgement capacity of colleagues (for instance at curriculum committee or board of studies), whereby the overall curriculum intent and the knowledge area plus the learning outcome can be used to identify a relevant set of dispositions.
- Identify a task to frame the competency statement. For a guidelines report like CS2013, the tasks can quite often be "verbs that describe professional activity". Some other techniques are discussed in the earlier parts of the paper.
- Develop rubrics to ensure good quality. Use them to review the resulting competency statements and validate them in terms of process and knowledge unit completeness.

This work has explored a variety of mechanisms to execute these steps, and has also taken a self-critical perspective. We offer some concrete suggestions for how to carry out the broad steps above. See, for example, Section 4.3 for the details of the common format we developed for writing competency statements, and Section 4.4.2 for a discussion of our validation approach.

Table 11: CS2013 Topic and LO Coverage for Final 43 Competency Specifications

KA Code	CS2013 KA	Competency Statement Count	Tier 1 & Tier 2 Topic/LO Count	Topics/LOs Addressed	Topic/LO Coverage	Topic+LOs / Statement
CS-AL	Algorithms	8	76	36	47%	4.5
CS-AR	Architecture and Organization	2	75	40	53%	20.0
CS-HCI	Human-Computer Interaction	4	27	24	89%	6.0
CS-IS	Intelligent Systems	4	40	37	93%	9.3
CS-NC	Networking and Communication	1	55	22	40%	22.0
CS-OS	Operating Systems	5	12	6	50%	1.2
CS-PL	Programming Languages	6	103	27	26%	4.5
CS-SDF	Software Development Fundamentals	2	85	24	28%	12.0
CS-SE	Software Engineering	3	138	62	45%	20.7
CS-SP	Social Issues and Professional Practice	4	92	27	29%	6.8
	Total	32	703	305	50%	10.1
PK	Professional & Foundational Knowledge	(4)	13	5	38%	1.3

6.2 What was Learned During the Process?

(1) Transformation of a traditional curricula document cannot be done mechanically. Competencies capture dispositions explicitly. Learning outcomes do not explicitly address the relevance of dispositions for professionalism and integrity. Developing competency statements thus often requires the introduction of dispositions that may not have been easy to consider, or include, in a learning-outcomes-based approach. Knowledge areas that demand significant systems level work, for example the competency OS-2, require considerable attention to detail, a disposition that needs to be identified and explicitly included when working with our transformative approach.

(2) Competencies derived from CS2013 are not universally valid, nor necessarily directly appropriate to all computing subfields. Competencies expected of graduates in relation to a particular knowledge area differ and are dependent on degree program objectives. Consider the operating systems knowledge area. One expects a science-oriented graduate to gravitate mainly towards correctness and efficiency of an OS, while an engineering interest would be concerned with efficiency and implementation challenges, and systems support personnel would look for reliable deployment. The curricular statements for different degree programs would steer the same knowledge areas towards their intent by distinguishing the level of the skills to be acquired and their disposition. In this sense competencies provide the ability to further tailor more traditional curricula to the local context of degree programmes.

(3) Given that competency statements describe the educational value for participating stakeholders, there are two natural views to a given set of competency statements: the curriculum design view and the curriculum consumption view. The design view is concerned with building up the competencies through the curriculum design of an educational program and typically exhibits a locally relevant context that is used to frame competency statements. The consumption view consists of the final competencies available to the stakeholders through an educational program. A guidelines report like the CS2013 might be expected to focus on the consumption view to be globally relevant. Due care must be exercised to ensure that the appropriate sense of context is incorporated into the competency statements.

(4) It is possible that a stated learning outcome is almost identical to a competency statement. For example, the competency statement OS-6 is almost identical to the learning outcome in CS2013, since the outcome is inextricably linked with connotations of disposition. Interestingly, it also requires the support of a few other learning outcomes from the same knowledge area to formulate a complete competency. While developing a competency statement for this learning outcome, we also observed that it required a higher Bloom's level than that specified in CS2013 in order to capture what we judged to be the intent.

(5) Formulating competencies also leads to a discussion of the level of abstraction at which a competency could, or should, be formulated. It is not clear that all KS-pairs delineated in a curriculum, such as CS2013, are obligatory when formulating a competency specification. In addressing the level of abstraction at which competencies are formulated we recommend the architects of future curricula to consider if earlier curricula contain KS pairs that may be optional; and where optional KS pairings might exist, upon what criteria they could be included, or excluded, from a given higher level competency. In addition, we recommend that the issue of whether optional pairings can/should be included in a competency be explicitly addressed.

Developing complete, observable, contextualized competency statements around skill level B-II turned out to be difficult, even to the point where we briefly considered extending the dispositions of Table 6 to include the concept of "Articulate". This proposal was discussed and rejected because the

term 'Articulate' is more of a quality statement about the application of professional knowledge than a proper disposition [25]. In particular, the concept of 'articulate' related to B-II topics and LOs is more correctly related to the demonstration of professional communication (e.g., PK-1, PK-2 in Table 4).

(6) One of the most important insights gained in our transformation exercise was the identification of what we consider to be some missing LOs and topics in CS2013. Therefore, it is essential to divulge that CS2013 should not be seen to represent the sole input for this work. Rather, our work includes additional disciplinary expertise and captures new insights that should inform future revision of CS guidelines, in particular with respect to completion of missing LOs and/or topics. In particular, the absence of professional knowledge areas, such as problem-solving, relationship management, etc., or professional attributes such as leadership.
In addition to this challenge, many competencies assume knowledge of areas such as inter-cultural communication, technical writing and rhetoric, which in CC2020 have been termed foundational and professional knowledge areas (see the forthcoming CC2020 report, Table 4.2 or the draft provided in Table 4). These foundational and professional knowledge areas undergird many competencies and are needed to augment the CS2013 document in order to help to define competencies that cannot be well formulated otherwise. This was an issue specifically identified in the AR-1 example presented in Section 4.
Our observations regarding Professional Knowledge are important at many levels. The issues related to foundational and professional knowledge areas extend well beyond more effective competency specification. The broad KAs of Table 4 are really much more like other 'cross-cutting' KAs described in the SWECOM [55]. These KAs represent a relatively complete collection of what are colloquially (and incorrectly) referred to as 'soft skills'. The need for CS students to develop skill and disposition in these KAs is a well-identified problem in computing education [8, 21, 26, 44, 67]. This appears to be an area where a future version of the CS2013 could be improved.
The CS2013 document, though immensely useful, demonstrates a paucity of discussion of what foundational and professional knowledge is required of computer science graduates, with only professional communication and some aspects of teamwork having been included. (See the related topics and LOs in the SE-SPM Software Process Management and SP-PC Professional Communication KAs in CS2013). This work suggests that how to frame this language dealing with professional and foundational knowledge needs more attention as we move towards a competency-based approach to CS education.

(7) The level of mastery at which LOs are written is also linked to levels of abstraction, and ultimately to the issue of whether a competency can be assessed. Familiarity-level LOs are needed; in fact they are prerequisites of the LOs at the usage and assessment level. In our work, we found the need to re-write some LOs as we found that student should be above the mastery level specified by CS2013. These were needed so they could connect to the task and be observable. For example, we extended the learning outcome AR-ALMO-4 'Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler' written at the Bloom's taxonomy level II with an enhanced LO 'Compare and contrast how instructions are represented at different levels of representation starting at the machine level through to a higher level representation' that is written at a higher Bloom's taxonomy level (IV).

(8) Unsurprisingly, some LOs and topics span a range of aspects of professional competence, and can be included in more than one competency. This may appear counter intuitive, but is actually desirable, as students can be exposed to (or use) the same concepts and/or learning material in different modules or courses. In our competency schema, we aimed to meet the expected Tier-1 and Tier-2 LOs as a baseline. By doing this, we believe that we can contribute best to the community through presenting this competency modeling view of CS2013. Practitioners and curriculum designers should find guidance and inspiration in this report which will help them to write their own competencies (e.g., SE statements in Section C.1.16).

(9) Combining a top-down and bottom-up approach to competency statement development was sometimes difficult. It was far easier to develop statements that were aimed at a graduate—using higher Bloom's levels, more dispositional terminology, etc. However, these were often more difficult to connect to the LOs and topics of CS2013. It was particularly difficult to connect well-written program-level competency statements (e.g., NC-1 in Section C.1.10).

(10) As Table 11 implies, we have only mapped a subset of CS2013 KUs. One of the disadvantages of working on a sub-set of the CS2013 is that identifying a clear hierarchical structure between competency statements has not been possible. Such a structure might potentially have allowed us to write smaller (or at the very least, more focused) competencies with clear dependencies between them. As it is, we end up with some competency statements that are slightly bloated because we are forced to include a number of "prerequisite" K-S pairs that might be better expressed through dependencies among competencies.

(11) We discovered that the "purpose-driven" disposition seemed applicable for several competencies, but at a higher Bloom's taxonomy level—at level IV or V. How to avoid adding all of the dispositions to most or all of the statements was occasionally difficult. We concluded that retaining an emphasis on the few key dispositions, considered most applicable for the given competency, was the most viable approach.

This work has explored a number of aspects of competency writing. In some stages of the process we adopted a curriculum-developer's perspective. The latter had to be introduced in an otherwise curriculum-developer agnostic document like CS2013 to augment the LOs with the additional information that competencies demand. These processes, lessons-learned, rubrics and coverage metrics all could be used to develop authoring tools for generating/recording

competency statements that are part of computing curricula. Similarly, the formats employed in this work, the connecting free-form text to linked knowledge-areas, topics and learning outcomes from formal curricular documents (like CS2013) may well support the visualization development efforts being explored for comparing computing programs using competencies [66].

6.3 Future Work

This work constitutes but one set of steps towards implementing a competency-based curriculum. A critical future step will involve assessment design and implementation, building upon the competency statements provided here for computer science, or as potentially applied to a separate computing or alternative disciplinary curriculum. No doubt issues relating to the gaps between the competency statements proposed here and a set of accompanying assessments will emerge. In addition the issue of hierarchies of competencies, and the implications this might have for writing more concise competencies associated with cognitive performances at the lower Bloom's levels, deserves further exploration.

However, more positively, these statements may provide a sound basis for assessment design and learning, as the process of moving towards more competency-based curriculum delivery evolves. Even with these notes, there remain many lessons learned that still require team reflection.

A further key step of this work relates to incorporation of the example competency statements developed here into the CC2020 tool, which is still under development. We aim to demonstrate how the population of the tool with concrete examples of competency-based curriculum materials, will support the intention of the tool to enable visualisation of curricula by various stakeholders. These statements (to be made available through an open repository of the full set of competency specifications and their review spreadsheets at zenodo.org, (https://doi.org/10.5281/zenodo.4270438) will also provide a useful validation of the interfaces required for input of data into the tool, and a set of test data to illustrate the operation of those interchange formats [66].

A particularly valuable additional activity could be the codification of context-free questions and/or a good-practice guide for developing competency statements. These types of questions were useful in identifying tension in the expression of competency statements as being too general or too specific. For instance, some context-free review questions we have used include:

- Are the dispositions expected referenced in the statement?
- Is the task specified well enough? Is it too specific?
- Is the breadth of knowledge and skills linked to the statement complete? Concise? Consistent? Too detailed?
- Is the statement sufficiently ambiguous to be used in multiple settings, or is it specific to the writers' context?

7 CONCLUSION

Codifying the knowledge and skills areas of the computer science 2013 curriculum into a format consistent with the CC2020 project competency framework has proven a considerable challenge. The guiding principle of the working group was to explore the implications of competency frameworks for the future development and comparison of programs worldwide.

This working group focused on CS2013 as a case study demonstrating the application of the CC2020 report methods to a specific curriculum. The outcome is a list of graduate competencies, however the important contribution is the process used to derive them. In developing this process for deriving and validating competencies, the working group demonstrates how CC2020 can be applied in practice. The competencies developed during this work provide an incomplete coverage of CS2013.

Developing descriptions of competency relevant to the knowledge areas of CS2013 presented significant challenges. Curricular documents typically focus on decontextualising knowledge, based on the assumption that this provides a universally relevant framing of knowledge and skill. Our work suggests, on the contrary, that context is vital to understanding the task, and the contextualisation is essential to being able to observe and assess competencies in educational practice.

In arriving at workable definitions of competencies, the working group members have extended the context of the 2013 knowledge and skill combinations. In doing this, the project takes into account the application area and the problem context with which a competency is closely linked. The resulting statements and their analysis provide unique insights into the nature of educational processes, and their implications for curriculum and instructional design and assessment.

This work stresses the importance of the re-contextualization of knowledge and skills by including important dispositional elements and a task to learning objectives published in CS2013. The report provides mechanisms for formulating competencies from older body-of-knowledge-centric curricula. This paves the way for systematic revision of existing curricula, in particular by identifying key challenges in the process, and providing potential resolution strategies. The guidance that this paper provides is rather an inspiration than a standard.

The redesign of existing curricula is one of many efforts needed to reshape computing education and consequently the method presented in this report is an important step in terms of helping to create a language and design principles for the application of the CC2020 report to existing and future curricula, helping stakeholders, students, employers and society appreciate the value of educational programs in the broad range of computing disciplines.

Redesigning the CS curricula based on competencies is consistent with social constructivism which doesn't stop at familiarity or application of knowledge, but allows learners to build practical and viable mental models and attitudes in the wide social context of the CS discipline. Leveraging competencies to document learning goals appears to be a clear improvement, and the extension of disposition as a required component of learning goals emphasises students' professional attitude development as well as potential foci for assessment [25].

The process for deriving and validating competencies employed by the working group was meticulous. It is important to highlight that besides scholarly expertise, the members of the working group have extensive practical experience in developing and teaching CS curricula. As the context of teaching and learning in post-secondary institutions around the globe could vary based on contextual, cultural or social differences, we consider that it is of paramount importance to evaluate and validate our process in a future empirical

study involving CS curriculum practitioners (designers, experts, and instructors) from various post-secondary institutions.

ACKNOWLEDGMENTS

Members of the Working Group would like to thank Leslie Waguespack, Jeff Babb, the CC2020 Tool Developers. the CS2013 Task Force. and the CC2020 Task Force.

REFERENCES

[1] ACM Computing Curricula Task Force (Ed.). 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM, Inc. https://doi.org/10.1145/2534860

[2] Joanna Allan. 1996. Learning outcomes in higher education. *Studies in higher education* 21, 1 (1996), 93–108. Type: Journal Article.

[3] Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths, and Merlin C. Wittrock. 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, abridged edition* (1 ed.). Pearson, New York, NY.

[4] R.M. Baron and D.A. Kenny. 1986. The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology* 52 (1986), 1173–1182. https://psych.wisc.edu/henriques/mediator.html

[5] S Beecham, T Clear, J Barr, M Daniels, M Oudshoorn, and J Noll. 2017. Preparing Tomorrow's Software Engineers for Work in a Global Environment. *IEEE Software* 34, 1 (2017), 9–12. https://doi.org/10.1109/MS.2017.16 Type: Journal Article.

[6] Mordechai Ben-Ari. 1998. Constructivism in computer science education. *ACM SIGCSE Bulletin* 30, 1 (March 1998), 257–261. https://doi.org/10.1145/274790.274308

[7] Meurig Beynon. 2009. Constructivist Computer Science Education Reconstructed. *Innovation in Teaching and Learning in Information and Computer Sciences* 8, 2 (June 2009), 73–90. https://doi.org/10.11120/ital.2009.08020073 Publisher: Routledge _eprint: https://doi.org/10.11120/ital.2009.08020073.

[8] Stephen Billett. 2009. Realising the educational worth of integrating work experiences in higher education. *Studies in Higher Education* 34, 7 (Nov. 2009), 827–843. https://doi.org/DOI:10.1080/03075070802706561

[9] Benjamin S Bloom, Max D Englehart, Edward J Furst, Walter H Hill, and David R Krathwohl. 1956. *Taxonomy of educational objectives.* Vol. 1: Cognitive domain. McKay, New York:. https://scholar.google.ca/scholar?q=Bloom,+B.+S.%3B+Engelhart,+M.+D.%3B+Furst,+E.+J.%3B+Hill,+W.+H.%3B+Krathwohl,+D.+R.+(1956).+Taxonomy+of+educational+objectives:+The+classification+of+educational+goals.+Handbook+I:+Cognitive+domain.+New+York:+David+McKay+Company.&hl=en&as_sdt=0&as_vis=1&oi=scholart

[10] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis, and Yves Punie. 2016. *Developing computational thinking in compulsory education.* JRC Working Papers JRC104188. Joint Research Centre (Seville site). 68 pages. https://publications.jrc.ec.europa.eu/repository/handle/JRC104188

[11] Claus Brabrand and Bettina Dahl. 2009. Analyzing CS Competencies Using the SOLO Taxonomy. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09).* ACM, New York, NY, USA, 1–1. https://doi.org/10.1145/1562877.1562879

[12] Barbara Brečko and Anusca Ferrari. 2016. *The digital competence framework for consumers.* Science for Policy EUR28133EN. Joint Research Center (JRC). 28 pages. http://www.acrmalta.com/wp-content/uploads/2017/04/DigCompConsumersfinalpublicationJRC103155.pdf DOI 10.2791/278444. Corpus ID: 114779981.

[13] Quinn Burke and Cinamon Sunrise Bailey. 2020. Becoming an 'adaptive' expert. *Commun. ACM* 63, 9 (2020), 56–64.

[14] Francesca Caena and Christine Redecker. 2019. Aligning teacher competence frameworks to 21st century challenges: The case for the European Digital Competence Framework for Educators (Digcompedu). *European Journal of Education* 54, 3 (2019), 356–369. Publisher: Wiley Online Library.

[15] Michael E Casrsen, Judith Gal-Ezer, Andrew McGettrick, and Enrico Nardelli. 2018. *Informatics for All The strategy.* ACM.

[16] T Clear. 2015. 'Follow the moon' development: writing a systematic literature review on Global Software Engineering Education [Keynote Presentation]. In *Proceedings of the 15th Baltic Sea Conference on Computing Education Research, (Koli Calling)*, P Kinnunen and J Sheard (Eds.). ACM, New York, 1–4. https://doi.org/10.1145/2828959.2835019 Type: Book Section.

[17] European Commission. 2018. Commission staff working document accompanying the document Proposal for a Council Recommendation on Key Competences for Lifelong Learning. (2018).

[18] Quintin Cutts, Elizabeth Patitsas, Elizabeth Cole, Peter Donaldson, Bedour Alshaigy, Mirela Gutica, Arto Hellas, Edurne Larraza-Mendiluze, Robert McCartney, and Charles Riedesel. 2018. Early developmental activities and computing proficiency. In *Proceedings of the 2017 ITiCSE Conference (Working Group Reports).* Association for Computing Machinery, Bologna, Italy, 140–157. https://doi.org/10.1145/3174781.3174789

[19] Mats Daniels, Åsa Cajander, Tony Clear, and Roger McDermott. 2015. Collaborative technologies in global engineering : New competencies and challenges. *International journal of engineering education* 31, 1 (2015), 267–281. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-240569

[20] Andrea Danyluk, Paul Leidig, Lillian Cassel, and Christian Servin. 2019. ACM Task Force on Data Science Education: Draft Report and Opportunity for Feedback. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* Association for Computing Machinery, Minneapolis, MN, 496–497.

[21] Michael Eraut. 2010. Knowledge, Working Practices, and Learning. In *Learning Through Practice: Models, Traditions, Orientations and Approaches*, Stephen Billett (Ed.). Springer Netherlands, Dordrecht, 37–58. https://doi.org/10.1007/978-90-481-3939-2_3

[22] Peter A. Facione, Noreen C. Facione, and Carol A. Giancarlo. 2000. The Disposition Toward Critical Thinking: Its Character, Measurement, and Relationship to Critical Thinking Skill. *Informal Logic* 20, 1 (2000), 61–84. https://www.insightassessment.com/content/download/755/4787/file/J_Infrml_Ppr+_2000+-+Disp+%26+Skls.pdf

[23] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M McGill, and Keith Quille. 2019. An International Study Piloting the MEasuring TeacheR Enacted Computing Curriculum (METRECC) Instrument. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education.* 111–142.

[24] Stephen Frezza, Åsa Cajander, Arnold Pears, Mats Daniels, Viggo Kann, Amanpreet Kapoor, Roger McDermott, Anne-Kathrin Peters, Charles Wallace, and Mihaela Sabin. 2018. Modeling global competencies for computing education. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018.* ACM Press, Larnaca, Cyprus, 348–349. https://doi.org/10.1145/3197091.3205844

[25] Stephen Frezza, Tony Clear, and Alison Clear. 2020. Unpacking Dispositions in the CC2020 Computing Curriculum Overview Report. In *Proceedings of the 50th Frontiers in Education Conference.* IEEE CS, Uppsala, Sweden, 9.

[26] Stephen Frezza, Mats Daniels, and Aaron Wilkins. 2019. Assessing Students' IT Professional Values in a Global Project Setting. *ACM Transactions on Computing Education* Special Issue on Global Software Eningeering Education, TBD (2019), 33.

[27] D Giordano, A Faro, F Maiorana, C Pino, and C Spampinato. 2009. Feeding back learning resources repurposing patterns into the "information loop": opportunities and challenges. In *9th International Conference on Information Technology and Applications in Biomedicine.* IEEE, Larnaca, Cyprus, 1–6. https://doi.org/DOI:10.1109/ITAB16117.2009

[28] Gerald Grant. 1979. On competence: a critical analysis of competence-based reforms in higher education. *ERIC* (1979).

[29] Paul Hager, Andrew Gonczi, and James Athanasou. 1994. General issues about assessment of competence. *Assessment and evaluation in higher education* 19, 1 (1994), 3–16. Type: Journal Article.

[30] Holly Hendrigan, Keshav Mukunda, and Diana Cukierman. 2020. A Case Study and Call to Action: Incorporating the ACRL Framework for Information Literacy in Undergraduate CS Courses. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education.* 198–204.

[31] Trevor Hussey, Patrick Smith, and Patrick Smith. 2012. *The Trouble with Higher Education : A Critical Examination of our Universities.* Routledge. https://doi.org/10.4324/9780203866344

[32] Leo Irakliotis and Sally Johnstone. 2014. Competency-based education programs versus traditional data management. *EDUCAUSE Review online* 1 (May 2014). https://er.educause.edu/articles/2014/5/competencybased-education-programs-versus-traditional-data-management

[33] Joint Task Force for Computing Curricula 2005. 2005. *Computing Curricula 2005.* Technical Report September. ACM / IEEE CS. 62 pages. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf Backup Publisher: The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), and The Computer Society (IEEE-CS) ISBN: 159593359X.

[34] Panagiotis Kampylis, Yves Punie, Jim Devine, and others. 2015. *Promoting effective digital-age learning-A European framework for digitally-competent educational organisations.* Technical Report. Joint Research Centre (Seville site).

[35] Declan Kennedy, Áine Hyland, and Norma Ryan. 2009. Learning outcomes and competencies. In *Introducing Bologna objectives and tools.* Number B 2.3-3. DUZ International, Cork, IRE, 59–76. https://pdfs.semanticscholar.org/423a/3e8656c0990f516ca78ace21ec79eacf8440.pdf

[36] Karoline Koeppen, Johannes Hartig, Eckhard Klieme, and Detlev Leutner. 2008. Current Issues in Competence Modeling and Assessment. *Zeitschrift für Psychologie / Journal of Psychology* 216, 2 (Jan. 2008), 61–73. https://doi.org/10.1027/0044-3409.216.2.61

[37] Matthias Kramer, Peter Hubwieser, and Torsten Brinda. 2016. A Competency Structure Model of Object-Oriented Programming. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*. IEEE CS, Mumbai, India„ 1–8. https://doi.org/10.1109/LaTiCE.2016.24

[38] Kris Magnusson and John Osborne. 1990. The rise of competency-based education: A deconstructionist analysis. *The Journal of Educational Thought (JET)/Revue de la Pensée Educative* 24, 1 (April 1990), 5–13. Type: Journal Article.

[39] Francesco Maiorana. 2019. Interdisciplinary Computing for STE (A) M: a low Floor high ceiling curriculum. *Innovations, Technologies and Research in Education* (2019), 37.

[40] A. L. Mchenry, D. R. Depew, M. J. Dyrenfurth, D. D. Dunlap, D. A. Keating, T. G. Stanford, and G. Deloatch. 2005. Constructivism: The Learning Theory That Supports Competency Development of Engineers For Engineering Practice and Technology Leadership Through Graduate Education. In *Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition*. ASEE, Portland, OR. https://peer.asee.org/collections/10

[41] B Mennecke and J Hoffer. 1992. The Implications of Group Development and History for Group Support System Theory and Practice. *Small Group Research* 23, 4 (1992), 524–572. Type: Journal Article.

[42] Martin Mulder. 2017. Competence Theory and Research: A Synthesis. In *Competence-based Vocational and Professional Education*. Springer, Cham, 1071–1106. https://doi.org/10.1007/978-3-319-41713-4_50

[43] Enrico Nardelli, Luca Forlizzi, Michael Lodi, Violetta Lonati, Claudio Mirolo, Mattia Monga, Alberto Montresor, and Anna Morpugo. 2017. *Proposal for a national Informatics curriculum in the Italian school*. Technical Report. National Interuniversity consortium for Informatics.

[44] A. Nylén, M. Daniels, V. Isomöttönen, and R. McDermott. 2017. Open-ended projects opened up - aspects of openness. In *2017 IEEE Frontiers in Education Conference (FIE)*. 1–7. https://doi.org/10.1109/FIE.2017.8190510

[45] Co-operation Organisation for Economic and Development. 2018. *The future of education and skills: Education 2030*. Directorate for Education and Skills, OECD Paris, France.

[46] Arnold Pears and Lauri Malmi. 2009. Values and Objectives in Computing Education Research. *ACM Transactions on Computing Education* 9, 3 (Sept. 2009), 1–6. https://doi.org/10.1145/1594399.1594400

[47] D. N. Perkins, Eileen Jay, and Shari Tishman. 1993. Beyond Abilities: A Dispositional Theory of Thinking. *Merrill-Palmer Quarterly* 39, 1 (1993), 1–21. http://www.jstor.org/stable/23087298

[48] Jean Piaget. 1976. Piaget's Theory. In *Piaget and His School: A Reader in Developmental Psychology*, Bärbel Inhelder, Harold H. Chipman, and Charles Zwingmann (Eds.). Springer, Berlin, Heidelberg, 11–23. https://doi.org/10.1007/978-3-642-46323-5_2

[49] Eetu Pikkarainen. 2014. Competence as a Key Concept of Educational Theory: A Semiotic Point of View. *Journal of Philosophy of Education* 48, 4 (Nov. 2014), 621–636. https://doi.org/10.1111/1467-9752.12080

[50] Katherine C. Powell and Cody J. Kalina. 2009. Cognitive and Social Constructivism: Developing Tools for an Effective Classroom. *Education* 130, 2 (2009), 241–250. Publisher: Project Innovation, Inc.

[51] Mihaela Sabin, Hala Alrumaih, and John Impagliazzo. 2018. A Competency-based Approach toward Curricular Guidelines for Information Technology Education. In *2018 Global Engineering Education Conference*. IEEE, Tenerife, 1214–1221. https://doi.org/10.1109/EDUCON.2018.8363368

[52] Mihaela Sabin, Svetlana Peltsverger, Bill Paterson, Ming Zhang, and Hala Alrumaih. 2017. IT2017 Report: Putting It to Work. In *Proceedings of the 18th Annual Conference on Information Technology Education (SIGITE '17)*. ACM, New York, NY, USA, 95–96. https://doi.org/10.1145/3125659.3125672

[53] Deborah L. Schussler. 2006. Defining Dispositions: Wading Through Murky Waters. *The Teacher Educator* 41, 4 (March 2006), 251–268. https://doi.org/10.1080/08878730609555387

[54] James Shiveley and Thomas Misco. 2010. "But How Do I Know About Their Attitudes and Beliefs?": A Four-Step Process for Integrating and Assessing Dispositions in Teacher Education. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* 83, 1 (Jan. 2010), 9–14. https://doi.org/10.1080/00098650903267669

[55] IEEE Computer Society. 2014. *Software Engineering Competency Model (SWECOM)*. Technical Report Version 1.0. https://www.computer.org/web/peb/swecom

[56] Angela Stoof, Rob L. Martens, and Jeroen JG Van Merriënboer. 2000. What is competence? A constructivist approach as a way out of confusion. *Onderwijs Research Dagen (ORD), Leiden, The Netherlands* (2000).

[57] Miho TAGUMA, Miho Senior Analyst, Eva FERON, and LIM Meow Hwee. 2018. Future of Education and Skills 2030: Conceptual Learning Framework. (2018).

[58] Cara Tang, Cindy Tucker, Christian Servin, Markus Geissler, and Melissa Stange. 2020. Curricular Guidance for Associate-Degree Cybersecurity Programs. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1285–1285.

[59] Scientific United Nations Educational and Cultural Organization (UNESCO). 2011. UNESCO ICT competency framework for teachers. (2011). Publisher: UNESCO, Paris, France.

[60] Marcel Van der Klink and Jo Boon. 2002. The investigation of competencies within professional domains. *Human Resource Development International* 5, 4 (2002), 411–424.

[61] Marcel Van der Klink, Jo Boon, and Kathleen Schlusmans. 2007. Competences and Vocational Higher Education: Now and in Future. *European journal of vocational training* 40, 1 (2007), 67–82.

[62] Rebecca Vivian and Katrina Falkner. 2019. Identifying Teachers' Technological Pedagogical Content Knowledge for Computer Science in the Primary Years. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 147–155.

[63] Richard A. Voorhees. 2001. Competency-Based Learning Models: A Necessary Future. *New Directions for Institutional Research* 2001, 110 (2001), 5–13. https://doi.org/10.1002/ir.7

[64] Riina Vuorikari, Yves Punie, Stephanie Carretero Gomez, Godelieve Van Den Brande, and others. 2016. *DigComp 2.0: The digital competence framework for citizens. Update phase 1: The conceptual reference model*. Technical Report. Joint Research Centre (Seville site).

[65] L. S. Vygotskiĭ and Michael Cole. 1978. *Mind in society: the development of higher psychological processes*. Harvard University Press, Cambridge.

[66] Leslie Waguespack and Babb. 2019. Toward Visualizing Computing Curricula: The Challenge of Competency. In *Information Systems Education Journal (ISEDJ)*, Vol. 17 n4. ISCAP, Cleveland, OH, 51–69. https://files.eric.ed.gov/fulltext/EJ1219541.pdf

[67] Leslie Waguespack, Heikki Topi, Stephen Frezza, Jeffry Babb, Linda Marshall, Shingo Takada, Gerrit van der Veer, and Arnold Pears. 2019. Adopting Competency Mindful of Professionalism in Baccalaureate Computing Curricula. In *2019 Proceedings of the EDSIGCON Conference*, Vol. 5 n4955. ISCAP, Cleveland, OH, 17. http://proc.iscap.info/2019/pdf/4955.pdf

A CS2013 KNOWLEDGE UNITS

The following provides a sample of the information provided in the CS2013 report that was used for tracing the learning outcome and topic information.

A.1 PL/Functional Programming

Tables 12 and 13 depict the full list of topics and learning outcomes for the Functional Programming KU.

Table 12: CS2013 Tier-1, Tier-2 and Elective Topics defined for the Knowledge Area "PL/Functional Programming"

Tag	Tier	Topic
PL-FP-a	1	Effect-free programming
PL-FP-a1	1	Function calls have no side effects, facilitating compositional reasoning
PL-FP-a2	1	Variables are immutable, preventing unexpected changes to program data by other code
PL-FP-a3	1	Data can be freely aliased or copied without introducing unintended effects from mutation
PL-FP-b	1	Processing structured data (e.g., trees) via functions with cases for each data variant
PL-FP-b1	1	Associated language constructs such as discriminated unions and pattern-matching over them
PL-FP-b2	2	Functions defined over compound data in terms of functions applied to the constituent pieces
PL-FP-c	2	First-class functions (taking, returning, and storing functions)
PL-FP-d	2	Function closures (functions using variables in the enclosing lexical environment)
PL-FP-d1	2	Basic meaning and definition—creating closures at run-time by capturing the environment
PL-FP-d2	2	Canonical idioms: call-backs, arguments to iterators, reusable code via function arguments
PL-FP-d3	2	Using a closure to encapsulate data in its environment
PL-FP-d4	2	Currying and partial application
PL-FP-e	2	Defining higher-order operations on aggregates, especially map, reduce/fold, and filter

Table 13: CS2013 Tier-1, Tier-2 and Elective Learning Outcomes for the Knowledge Area "PL/Functional Programming"

Tag	Tier	Level	Learning Outcome (LO)
PL-FP-1	1	U	Write basic algorithms that avoid assigning to mutable state or considering reference equality. [Usage]
PL-FP-2	1	U	Write useful functions that take and return other functions. [Usage]
PL-FP-3	1	A	Compare and contrast (1) the procedural/functional approach (defining a function for each operation with the function body providing a case for each data variant) and (2) the object-oriented approach (defining a class for each data variant with the class definition providing a method for each operation). Understand both as defining a matrix of operations and variants. [Assessment] This outcome also appears in PL/Object-Oriented Programming.
PL-FP-4	2	U	Correctly reason about variables and lexical scope in a program using function closures. [Usage]
PL-FP-5	2	U	Use functional encapsulation mechanisms such as closures and modular interfaces. [Usage]
PL-FP-6	2	U	Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language. [Usage] This outcome also appears in PL/Object-Oriented Programming.

A.2 DS/Sets, Relations, and Functions

Tables 14 and 15 summarize the full KU for the DS-SRF (Sets, Relations, and Functions) Knowledge Unit.

Table 14: CS2013 Tier-1, Tier-2 and Elective Topics defined for the Knowledge Area "DS/Sets, Relations, and Functions"

Tag	Tier	Topic
DS-SRF-a	1	Sets
DS-SRF-a1	1	Venn diagrams
DS-SRF-a2	1	Union, intersection, complement
DS-SRF-a3	1	Cartesian product
DS-SRF-a3	1	Power sets
DS-SRF-a4	1	Cardinality of finite sets
DS-SRF-b	1	Relations
DS-SRF-b1	1	Reflexivity, symmetry, transitivity
DS-SRF-b2	1	Equivalence relations, partial orders
DS-SRF-c	1	Functions
DS-SRF-c1	1	Surjections, injections, bijections
DS-SRF-c2	1	Inverses
DS-SRF-c3	1	Composition

Table 15: CS2013 Tier-1, Tier-2 and Elective Learning Outcomes for the Knowledge Area "DS/Sets, Relations, and Functions"

Tag	Tier	Level	Learning Outcome (LO)
DS-SRF-1	1	F	Explain with examples the basic terminology of functions, relations, and sets. [Familiarity]
DS-SRF-2	1	U	Perform the operations associated with sets, functions, and relations. [Usage]
DS-SRF-3	1	A	Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context. [Assessment]

B ORIGINAL STATEMENTS

Tables 16 to 18 show the variety of formats and level of detail in the original competency statements written by different members of the working group. Section 4 describes how we reconciled these differences.

Table 16: Draft textual statement #1: skills are not paired with knowledge elements; no labelling of individual LOs/ topics from CS2013.

Functional Programming

(1) Correctly write an algorithm that avoids assigning to a mutable state while considering reference equality in functional programming.
(2) Correctly write a function that takes and returns other functions as a part of functional programming.
(3) As an example of functional programming, accurately compare and contrast the functional approach and the object-oriented approach for each data variant in a given class definition, providing a method for each operation.
(4) Correctly describe variables and lexical scope in a program using function closures.
(5) Correctly write a method using functional encapsulation mechanisms such as closures and modular interfaces.

The codes used are:

- Knowledge Skills CS-PL Functional Programming (Tables 12 and 13)
- Skills B-2 Understanding; B-3 Applying (Table 3)
- Disposition D-9 Meticulous (Table 6)

Table 17: Draft textual statement #2: all the desired elements are present, except there is no skill level indicated for LOs.

DS-1. Given a problem involving analysis of populations (such as traits, geographic origins, social status, etc.), design a solution that utilizes sets and set operations in the processing and presenting of results in response to user queries.
Knowledge Elements:
Leverages LOs (minimally)

- CS-DS-SRF-1. Explain with examples the basic terminology of functions, relations, and sets.
- CS-DS-SRF-2. Perform the operations associated with sets, functions, and relations.
- CS-DS-SRF-3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.

Leverages Topics (minimally):

- CS-DS-SRF-a. Sets. (III)
- CS-DS-SRF-a1. Venn diagrams. (III)
- CS-DS-SRF-a2. Union, intersection, complement. (III)
- CS-DS-SRF-a3. Cartesian product. (III)
- CS-DS-SRF-b. Relations. (III)

Strongly related dispositions

- Passionate
- Purpose-Driven
- Responsible
- Meticulous

C FIRST DRAFT COMPETENCY STATEMENTS

In this section, we provide the first draft of competency statements developed for Knowledge Areas in CS2013. Initially, the author team co-wrote 36 sample competency statements, spanning 11 of the 18 KAs defined in CS2013 (AL, AR, DS, HCI, IS, NC, OS, PL, SDF, SE, SP). Following the pattern provided in Figure 5, these statements have each had competency specifications developed [25, 67], samples of which are provided in Appendix D. As part of the broader effort to understand how the LO/Topic approach to curricular definition (like CS2013) maps to a competency-based approach, the 36 statements from the 11 sample KAs have been traced to the corresponding topics and learning outcomes for completeness/coverage.

Table 19 lists the different CC2013 KAs that were addressed, and how many competency statements were specified. This also shows the number of KUs for which there were one or more Tier-1 and Tier-2 topics and/or learning outcomes defined. These topics and LOs were the initial coverage target, as these represent the minimum learning expected of a CS2013-compliant degree program. The potential 87 Tier-1 and Tier-2 KUs represent 68% of the 127 Tier-1 and Tier-2 KUs in the CS2013.

For a more specific example, Table 20 illustrates the coverage provided by the complete specification for the two HCI competencies listed below (C.1.6). These two statements are intended to provide complete coverage for 10/10 topics and 4/5 LOs defined for the HCI-F (Foundations) KU. Additionally, they provide coverage for 8/8 topics and 1/4 LOs defined for the HCI-DI (Designing Interaction) KU. No coverage is provided for the other eight KUs. At

Table 18: Draft tabular statement #1: dispositions are paired with knowledge and skill; missing CS2013 codes.

Ref#	Title	Competency Statement	Dispositions	Knowledge-Skill Pairs		Constituent Elements
CS-NC-RDD	Reliable Data Delivery	Capable of analysing and proposing approaches to dealing with the different situations that arise in managing data transfer and communication over a lossy medium/channel based on contextually relevant priorities and desired outcomes.	Purposefulness	Network layer protocols	Use network layer protocols to connect applications	
			Professional/ Inventive	Performance issues	UDP vs. TCP ability to analyse QoS in the context of an application area	
			Meticulous	Flow control	Managing reliable delivery and network capacity	
			Adaptable	Error control	Mitigation of the impact of data corruption on the operation of a service	

Table 19: Summary of CS2013 KAs Addressed by 36 Competency Specifications

Code	Knowledge Area	Statement Count	Tiers 1 & 2 KUs
AL	Algorithms	6	6
AR	Architecture and Organization	2	7
DS	Discrete Structures	2	6
HCI	Human Computer Interaction	2	10
IS	Intelligent Systems	4	11
NC	Networking and Communications	2	7
OS	Operating Systems	6	6
PL	Programming Languages	6	17
SDF	Software Development Fundamentals	2	4
SE	Software Engineering	2	6
SP	Social Issues and Professional Practice	2	7
		36	87

first glance this assessment appears to be very lax; only two of ten KUs are addressed. However, this is misleading, as only these two KUs (HCI-F and HCI-DI) have any Tier-1 or Tier-2 expectations. All other eight KUs are elective topics. Consequently, these two statements provide near-complete coverage of HCI/Foundations and some coverage of the HCI/Designing Interaction KUs. No elective topics or KUs were included.

C.1 The Draft Statements

The 36 competency statements (not the full specifications) are listed in the subsections that follow. These subsections list the free-form text statement of the preliminary competencies developed with that KA in mind, prior to their undergoing the review process described in Section 4.

C.1.1 AL—Algorithms and Complexity.

AL-1 Given a problem to program for which an inefficient solution is not sufficient, research and implement a solution that is sufficient for reasonable instances, without straining other resources.

AL-2 Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior.

AL-3 Given a problem that involves multiple actions (such as insertions, deletions, and searches) that may or may not have predictable occurrences, properties, or relationships, devise a data structure that optimally supports the actions.

AL-4 Present to an audience of co-workers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops.

AL-5 Given partial specifications for a program that is to include a small user-interface language, devise a syntax that accommodates the expressiveness of the interface, is simple to use and not prone to user mistakes, and can be implemented using a finite state machine solution. Present the syntax to the users as regular expressions.

AL-6 Prepare a presentation that introduces non-technical managers to the basic concepts of algorithmic complexity, such as notation, characteristics, complexity classes, time and space, empirical measurement, and impact on practical problems.

C.1.2 AR—Architecture and Organization.

AR-1 For a given modern pipelined computer architecture, explain in sufficient and correct detail the execution of a sequence of code at the instruction-level (fetch, decode, execute, fetch operands, save results), the bus transfer of instruction and data from and to cache and main memory, data integrity (e.g., error-correction), and the impact on/use of memory organization. [Note: MSOA and ALMO-focused]

AR-1 In the context of modern computer architectures, compare and contrast differences between equivalent representations of computer functionality, instruction sequences/code (e.g., instructions, addresses) and data (e.g., numbers, characters, data structures, images, audio files, etc.) across multiple levels of abstraction (mathematical, digital-logic, machine-code, assembly code, HOL, etc.) [Note: MLRD-Focused]

C.1.3 CN—Computational Science. None developed as part of this report.

C.1.4 DS—Discrete Structures.

DS-1 Given a problem involving analysis of populations (such as traits, geographic origins, social status, etc.), design a solution that utilizes sets and set operations in the processing and presenting of results in response to user queries.

DS-2 Given a problem, which in the general case has no efficient solution, and a proposed efficient solution for a contraction of the problem, prove that the solution is correct.

C.1.5 GV—Graphics and Visualization. None developed as part of this report.

C.1.6 HCI—Human-Computer Interaction.

HCI-1 Purposefully follow a user-centered design process and adaptively engage users to develop and document an analysis of the user needs for a human-computer interaction with a proposed computing application, device or system.

HCI-1 Work independently and rigorously adhere to a user-centered design process, adaptively engaging non-peer real users in the development of an interactive application.

C.1.7 IAS—Information Assurance and Security. None developed as part of this report.

C.1.8 IM—Information Management. None developed as part of this report.

C.1.9 IS—Intelligent Systems.

IS-1 Demonstrate commitment and insight in showing how awareness of ethical applications of AI and the nature of intelligent behavior and reasoning, support a professional attitude to determining whether an AI approach is appropriate to a given problem, and justifying or otherwise its merits.

Table 20: Topic and LO Coverage for two HCI Competency Specifications

KU Code	Knowledge Unit	KU		Coverage	
		Topics	LOs	Topic	LO
HCI-F	HCI/Foundations	10	5	10	4
HCI-DI	HCI/Designing Interaction	8	4	8	1
HCI-PIS	HCI/Programming Interactive Systems	12	4	0	0
HCI-UCDT	HCI/User-Centered Design & Testing	4	5	0	0
HCI-NIT	HCI/New Interactive Technologies	6	3	0	0
HCI-CC	HCI/Collaboration & Communication	7	4	0	0
HCI-SM	HCI/Statistical Methods for HCI	7	4	0	0
HCI-HFS	HCI/Human Factors & Security	9	5	0	0
HCI-DO	HCI/Design-Oriented HCI	11	3	0	0
HCI-MAVR	HCI/Mixed, Augmented and Virtual Reality	22	6	0	0

IS-2 Show the ability to independently and carefully frame, select, implement and judge the most appropriate approach by distinguishing both suitable algorithms to meet the goals set in a given search problem and balance the trade-offs of completeness, optimality, time complexity and space complexity,

IS-3 Show care in flexibly constructing differing representations of knowledge, while applying multiple approaches to logical reasoning to draw well evidenced conclusions.

IS-4 Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier (A detailed expansion showing how this competency statements maps to its specification is provided in Table 7, in Section 4.)

C.1.10 NC—Networking and Communications.

NC-1 Capable of analysing and proposing approaches to dealing with the different situations that arise in managing data transfer and communication over a lossy medium/channel based on contextually relevant priorities and desired outcomes.

NC-2 Capable of formulating appropriate ways of developing a sufficient representation of the internal topology, connectivity and properties of network entities and use that information and representations in order to manage data flow in the topology in a relevant manner consistent with a series of underlying ethical and culturally relevant principles.

C.1.11 OS—Operating Systems.

OS-1 Given an unfamiliar OS, proactively find descriptive and functional information about it.

OS-2 Explain the abstractions involved and the resources used to use a device (e.g., print a text file on a printer).

OS-3 Clearly demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks.

OS-4 Identify ways that the logic embodied in scheduling algorithms are applicable to other domains, such as disk I/O, network scheduling, project scheduling, and problems beyond computing.

OS-5 Design a memory usage, optionally including caching strategy, over all the available memory hierarchies for various applications (e.g., web application).

OS-6 Carry out simple system administration tasks according to a security policy, for example creating accounts, setting permissions, applying patches, and arranging for regular backups.

C.1.12 PBD—Platform-based Development. None developed as part of this report.

C.1.13 PD—Parallel and Distributed Computing. None developed as part of this report.

C.1.14 PL—Programming Languages.

PL-1 Correctly demonstrate understanding of the basic components of object-oriented programming by writing a code segment with properties including state, behavior, class hierarchy, inheritance, fields, encapsulation, methods, constructors, iteration, and dynamic dispatch.

PL-2 Correctly demonstrate understanding of functional programming by writing a code segment with properties including taking and returning functions, function closures, functional encapsulation, and iteration.

PL-3 Accurately and effectively write a code segment which demonstrates all the concepts of event-driven and reactive programming.

PL-4 Correctly demonstrate understanding basic type systems by writing a code segment which correctly uses typing.

PL-5 Correctly demonstrate understanding of program representation options by writing a small program to process some representation of code for some specific purpose, such as an interpreter.

PL-6 Correctly demonstrate understanding of language translation and execution by writing a program to compile, interpret, or translate a small program.

C.1.15 SDF—Software Development Fundamentals.

SDF-1 Consider various algorithms to solve a problem, along with their key properties, and select the most appropriate algorithm based on careful consideration of their strengths and weaknesses.

SDF-2 Carefully use appropriate, standard data structures (such as arrays, linked lists, strings, stacks, queues, sets, maps)

to write a program that carries out a well-specified task correctly and efficiently.

C.1.16 SE—Software Engineering.

SE-1 As a team, purposefully develop a project proposal document for a team software development/enhancement project based on requirements; responsibly estimate the team planning, effort, risks, security with meticulous attention to quality.

SE-2 As a team, follow a software engineering methodology, use the appropriate tools and demonstrable good practices for designing, developing, integrating and testing a project according to a documented proposal, requirement documents or statement of work.

C.1.17 SF—Systems Fundamentals. None developed as part of this report.

C.1.18 SP—Social Issues and Professional Practice.

SP-1 Using empirical data with judgement and ethically focused approach, evaluate a given (software/hardware) design and its implementation using criteria such as:
- (a) developers' assumptions and values embedded in the design
- (b) design usability for diverse population (including under-represented population and the disabled)
- (c) under-representation of diverse population in the computing profession

SP-2 For a current computing issue with global implications, leveraging professional codes of ethics (e.g., ACM, IEEE Computer Society, and others), develop an insightful analysis of the role(s) of the community, professionals and government(s) for addressing this issue.

D EXEMPLAR COMPETENCY STATEMENTS

This appendix contains the complete specification for three of the 43+ competency statements developed and reviewed as a part of the Working Group activities. These three specifications supplement the full specification for the IS-4 (*IS /Basic Machine Learning*) competency statement depicted in Table 7 in Section 4. Table 21 depicts the full specification for the Operating Systems KA for the OS-4 (*Operating Systems Principles*) statement. Tables 22 and 23 depict the full specification for the AR-1 (*Instruction Execution*) competency statement drawn from the Architecture and Organization KA.

Tables 24 and 25 depict the full specification for the HCI-3 (*Mobile/Interactive Systems Development*) competency statement drawn from the Human-Computer Interaction KA. These four examples are described in more detail in Section 4.

Table 21: OS-4: OS/Operating Systems Principles

Statement		Dispositions		
Explain the abstractions involved and the resources used to print a text file on a printer.		D-10 (Meticulous)		
Knowledge Unit	**Topic/LO Description**	**Knowledge Element**	**Skill Level**	**Notes**
OS/Operating Systems Principles	List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised	OS-OSP-1	B-II	
	Describe the value of APIs and middleware	OS-OSP-3	B-III	
	Describe how computing resources are used by application software and managed by system software	OS-OSP-4	B-II	
	Explain the use of a device list and driver I/O queue	OS-OSP-7	B-II	
	Structuring methods (monolithic, layered, modular, micro-kernel models)	OS-OSP-a	B-II	
	Abstractions, processes, and resources	OS-OSP-b	B-III	
	Concepts of application program interfaces (APIs)	OS-OSP-c	B-III	
	Device organization	OS-OSP-e	B-II	

Table 22: Full Competency Specification for *AR-1* (Instruction Execution)

Statement	Dispositions
For a given modern pipelined computer architecture, coherently explain in sufficient and correct detail 1) the execution of a sequence of code at the instruction-level (fetch, decode, execute, fetch operands, save results), 2) the bus transfer of instruction and data from and to cache and main memory, 3) the data integrity (e.g., error-correction), and 4) the impact on/use of memory organization. [Note: MSOA & ALMO-focused]	D-6 (Responsible) D-9 (Responsive) D-10 (Meticulous)

Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
AR/Assembly-Level Machine Organization	Explain the organization of the classical von Neumann machine and its major functional units.	AR-ALMO-1	B-II	
	Describe how an instruction is executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution.	AR-ALMO-2	B-II	
	Describe instruction level parallelism and hazards, and how they are managed in typical processor pipelines.	AR-ALMO-3	B-II	
	Summarize how instructions are represented at both the machine level and in the context of a symbolic assembler.	AR-ALMO-4	B-II	
	Compare and contrast how instructions are represented at different levels of representation starting at the machine level through to a higher level representation.	AR-ALMO-4+	B-IV	Extends CS2013 LO
	Demonstrate how to map between high-level language patterns into assembly/machine language notations	AR-ALMO-5	B-III	
	Basic organization of the von Neumann machine	AR-ALMO-a	B-III	
	Control unit; instruction fetch, decode, and execution	AR-ALMO-b	B-III	
	Instruction sets and types (data manipulation, control, I/O)	AR-ALMO-c	B-III	
	Assembly/machine language programming	AR-ALMO-d	B-III	
	Instruction formats	AR-ALMO-e	B-III	
	Addressing modes	AR-ALMO-f	B-III	
SP/ Professional communication	Reading, understanding and summarizing technical material, including source code and documentation	SP-PC-a	B-III	
	Write clear, concise, and accurate technical documents following well-defined standards for format and for including appropriate tables, figures, and references. [Usage]	SP-PC-1	B-III	Implies PK-2

Table 23: Full Competency Specification for *AR-1* (Instruction Execution) (Cont.)

Statement	Dispositions
For a given modern pipelined computer architecture, coherently explain in sufficient and correct detail 1) the execution of a sequence of code at the instruction-level (fetch, decode, execute, fetch operands, save results), 2) the bus transfer of instruction and data from and to cache and main memory, 3) the data integrity (e.g., error-correction), and 4) the impact on/use of memory organization. [Note: MSOA & ALMO-focused]	D-6 (Responsible) D-9 (Responsive) D-10 (Meticulous)

Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
AR/Memory System organization & Architecture	Identify the main types of memory technology (e.g., SRAM, DRAM, Flash, magnetic disk) and their relative cost and performance	AR-MSOA-1	B-II	
	Explain the effect of memory latency on running time	AR-MSOA-2	B-III	
	Describe how the use of memory hierarchy (cache, virtual memory) is used to reduce the effective memory latency	AR-MSOA-3	B-III	
	Describe the principles of memory management	AR-MSOA-4	B-II	
	Explain the workings of a system with virtual memory management	AR-MSOA-5	B-II	
	Storage systems and their technology	AR-MSOA-a	B-III	
	Storage systems and their technology	AR-MSOA-b	B-III	
	Storage systems and their technology	AR-MSOA-c	B-III	
	Latency, cycle time, bandwidth, and interleaving	AR-MSOA-d	B-III	
	Cache memories (address mapping, block size, replacement and store policy)	AR-MSOA-e	B-III	
	Multiprocessor cache consistency/Using the memory system for inter-core synchronization/atomic memory operations	AR-MSOA-f	B-II	
	Virtual memory (page table, TLB)	AR-MSOA-g	B-II	
	Fault handling and reliability	AR-MSOA-h	B-II	
	Error coding, data compression, and data integrity (cross-reference SF/Reliability through Redundancy)	AR-MSOA-i	B-II	

Table 24: Competency Statement *HCI-3* (Mobile /Interactive Systems Development)

Statement	Dispositions
As a small independent development team, leverage a wireframe UI design to design, develop, test and evaluate a prototype interactive mobile application. Contrast the mobile development process to that for non-mobile applications.	D-2 Self-Directed D-4 Purpose-Driven D-6 Responsible D-8 Collaborative

Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
HCI/Programming Interactive Systems	Interaction Design Patterns: visual hierarchy, navigational distance	HCI-PIS-a	B-III	
	Event management and user interaction	HCI-PIS-b	B-III	
	Geometry management (cross-reference GV/Geometric Modelling)	HCI-PIS-c	B-III	
	Presenting information: navigation, representation, manipulation	HCI-PIS-e	B-III	
	Interface animation techniques (e.g., scene graphs)	HCI-PIS-f	B-III	
	Explain the importance of Model-View controller to interface programming	HCI-PIS-1	B-II	
	Create an application with a modern graphical user interface	HCI-PIS-2	B-III	
	Identify commonalities and differences in UIs across different platforms	HCI-PIS-3	B-II	
	Explain and use GUI programming concepts: event handling, constraint-based layout management, etc.	HCI-PIS-4	B-III	
HCI/ fundamentals	Principles of good design and good designers; engineering tradeoffs	HCI-F-h	B-IV	
HCI/User-Centered Design & Testing	Approaches to, and characteristics of, the design process	HCI-UCDT-a	B-II	
	Use lo-fi (low fidelity) prototyping techniques to gather, and report, user responses	HCI-UCDT-2	B-III	
	Identify commonalities and differences in UIs across different platforms	HCI-UCDT-3	B-III	
	Explain and use GUI programming concepts: event handling, constraint-based layout management, etc.	HCI-UCDT-4	B-III	
SE/Software Design	Create appropriate models for the structure and behavior of software products from their requirements specifications	SE-SD-6	B-III	
	Explain the relationships between the requirements for a software product and its design, using appropriate model	SE-SD-7	B-III	
	Apply simple examples of patterns in a software design	SE-SD-11	B-III	
	Select suitable components for use in the design of a software product	SE-SD-13	B-III	
SE/Software Verification & Validation	Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance)	SE-SVV-4	B-II	
	Create and document a set of tests for a medium-size code segment	SE-SVV-6	B-III	
	Describe how to select good regression tests and automate them	SE-SVV-7	B-III	
PBD/Mobile Platforms	Create appropriate models for the structure and behavior of software products from their requirements specifications	PBD-MP-1	B-III	
	Discuss the constraints that mobile platforms put on developers	PBD-MP-2	B-II	
	Discuss the performance vs. power tradeoff	PBD-MP-3	B-II	
	Compare and contrast mobile programming with general purpose programming	PBD-MP-4	B-IV	

Table 25: Competency Statement *HCI-3* (Mobile /Interactive Systems Development) (cont.)

Statement	Dispositions
As a small independent development team, leverage a wireframe UI design to design, develop, test and evaluate a prototype interactive mobile application. Contrast the mobile development process to that for non-mobile applications.	D-2 Self-Directed D-4 Purpose-Driven D-6 Responsible D-8 Collaborative

Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
SE/Tools & Environments	Utilize version control to help manage software release management	SE-TE-2	B-III	Extends SE-SE-2
	Identify configuration items and use a source code control tool in a small team-based project	SE-TE-3	B-III	
	Demonstrate the capability to use software tools in support of the development of a software product of medium size	SE-TE-6	B-III	
SE/Software Process	Demonstrate through involvement in a team project the central elements of team building and team management	SE-SP-10	B-III	
	Create a team by identifying appropriate roles and assigning roles to team members	SE-SP-12	B-III	
	Assess and provide feedback to teams and individuals on their performance in a team setting	SE-SP-13	B-III	