Chapter 49
Computational Thinking as an Emerging Competence Domain

Aman Yadav, Jon Good, Joke Voogt, and Petra Fisser

49.1 Introduction

Wing (2006) posited computational thinking (CT) as a problem-solving approach that draws on concepts fundamental to computer science by "reformulating a seemingly difficult problem into the one we know how to solve, perhaps by reduction, embedding, transformation, or simulation" (p. 33). Hence, computational thinking involves decomposing problems, using algorithms to solve problems, and abstracting and automating the problem-solving approach. Even though computational thinking includes a "range of mental tools that reflect the breadth of computer science," Wing argued that CT represents "a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (p. 33).

Though Wing's (2006) article popularized the term "computational thinking," Denning (2009) suggested that CT has a long history in computer science since the 1950s when it was known as "algorithmic thinking." Some aspects of algorithmic thinking approaches can be found in Polya's (1945) "How to Solve It." While working without the benefit of modern computing, Polya's work examined how everyday problems could be approached in a disciplined manner, decomposed into smaller problems, and solved using common techniques. More recently, Papert (1980, 1991) tied the practice of computer science to the act of thinking, which became a

A. Yadav (✉) • J. Good
Michigan State University, East Lansing, MI, USA
e-mail: ayadav@msu.edu

J. Voogt
University of Amsterdam, Amsterdam, The Netherlands

Windesheim University of Applied Sciences, Zwolle, The Netherlands

P. Fisser
Netherlands Institute for Curriculum Development, Enschede, The Netherlands

M. Mulder (ed.), *Competence-based Vocational and Professional Education*,
Technical and Vocational Education and Training: Issues, Concerns and
Prospects 23, DOI 10.1007/978-3-319-41713-4_49

central theme of his work with the LOGO programming language. Papert saw the students' creation of "microworlds" in LOGO as an epistemological apprenticeship to help develop students' thinking and problem-solving abilities. Papert believed that the LOGO environment provided students with opportunities to program the computer, which "could contribute to mental processes not only instrumentally but in more essential, conceptual ways, influencing how people think even when they are far removed from physical contact with a computer" (Papert 1980, p. 4). In our preferred conceptualization of computational thinking, we are attempting to shift the focus from the programming tools to the actual thinking skills Papert's methods were hoping to enhance.

So what constitutes computational thinking? What are the different components of computational thinking? Wing (2008) proposed computational thinking as two As: choosing the right *abstractions* and *automation* of those abstractions. In particular, Wing argued that computational thinking is reformulating complex problems into one which we know how to solve using abstractions and decomposition. The Royal Society in the United Kingdom (2012) similarly described computational thinking as "the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes" (p. 29). Barr and Stephenson (2011) expanded on these ideas and proposed nine core computational thinking concepts and capabilities, which included data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation.

These computational thinking ideas are fundamental parts of twenty-first century skills that are ubiquitous and can be applied across a wide range of fields. Wing (2006) argued that this new competence should be added to every child's analytical ability as a vital ingredient in the classroom given the pervasiveness of computational thinking in other disciplines, including statistics, biology, chemistry, and physics. Barr and Stephenson (2011) further asserted that given the importance of computing in the lives of students and how many of them will work in fields influenced by computation, it is critical that we begin to engage them in using algorithmic problem-solving and computational tools.

The following sections discuss key components of computational thinking competencies and how three countries (England, the Netherlands, and the United States) are addressing computational thinking in elementary and secondary classrooms. We also discuss the implications for those computational thinking competencies for vocational education and provide directions for future work in this area.

49.2 Three Country Cases

In this section, three cases of how national organizations in England, the Netherlands, and the United States are addressing the need for computational thinking in compulsory elementary and secondary education (i.e., K-12 schools) are presented.

49.2.1 Computing in the National Curriculum in England

In 2012, the Royal Society published a highly influential report with the title "Shut down or restart" about the state of computing in the UK schools. Computing was defined as a broad term and referred to information and communication technology (ICT) as it is used in schools and the use of information technology in the industry. The complaint expressed in the report was that ICT as a compulsory curriculum subject mainly focused on the acquisition of digital literacy, the general ability to use computers. The Royal Society argued that there is a need to also pay attention in the compulsory curriculum to information technology, the use of computers throughout society, including aspects such as the architecture of IT systems and human-computer interaction as well as computer science as an academic discipline that focuses on issues such as programming languages, data structures, and algorithms. For many students, computing was associated with low-level skills, because of the emphasis on digital literacy only. As a result, students did not have the opportunity to become interested in information technology or computer science as a potential area of interest for further studies, and only a few students continued their education in these fields. Without ignoring the importance of developing digital literacy skills in every child as a basis for being able to function in a society driven by digital technologies, the Royal Society argued for the recognition of information technology and computer science as important and foundational subjects in compulsory education. It was stated that "Every child should have the opportunity to learn concepts and principles from Computing (including Computer Science and Information Technology) from the beginning of primary education onward, and by age 14 should be able to choose to study toward a recognized qualification in these areas" (p. 44). In particular, they contended a high status for computer science, like other core subjects such as mathematics and history.

Core in England's computing curriculum is the importance attached to computational thinking. The definition of computational thinking the Royal Society adheres to emphasizes the importance of approaches, tools, and techniques from computer science to understand and reason about both natural and artificial systems and processes in the world around us. Computational thinking, in the view of the Royal Society, goes beyond computer science as a subject but is becoming part of many other disciplines and might even change other disciplines in a fundamental way. From an educational perspective, the following arguments support the need to position computer science as a core subject in the curriculum:

- Computer science develops key thinking skills in children: logical reasoning, modeling, abstraction, and problem-solving. Important aspects of computer science are the ability to be precise (a lack of precision makes a computer program fail and requires debugging) and the ability to break down a problem into subproblems and develop solutions for subproblems and interactions between these solutions.

- The ultimate goal of computer science is to create solutions for problems. It requires students to be creative, often in teams, and they need to apply "principles of quality, workmanship, fitness for purpose, and considerations of project management" (p. 29) to be able to solve complex computational problems.
- And finally, computation is a fundamental part of today's digital world. So for children to become an active participant in our society, they need to understand computing concepts to be able to contribute to highly sensitive societal issues involving computing, such as privacy and cyber criminality.

Based on the report of the Royal Society, England has commenced the implementation of a new compulsory curriculum for computing as of September 2014 for 5–16 year-olds. Although the new computing curriculum comprises three subdomains – computer science, information technology, and digital literacy – it is obvious that computer science forms a major part of the curriculum. The curriculum has four main goals (Department of Education 2013). All students should be able to:

- Understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms, and data representation
- Analyze problems in computational terms and have repeated practical experience writing computer programs in order to solve such problems
- Evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- Act as responsible, competent, confident, and creative users of information and communication technology

The national curriculum in England has four key stages: Key Stage 1 (5–7-year-olds), Key Stage 2 (7–11-year-olds), Key Stage 3 (12–14-year-olds), and Key Stage 4 (14–16-year-olds). We focus on the CT competencies that are part of the new computing curriculum (based on the elaboration of the curriculum by Naace and CAS 2014).

In Key Stage 1, students become acquainted with simple algorithms (e.g., recipes, describing the way home) and use simple programming tools to develop algorithmic notion (e.g., using LOGO or similar programs). Students should understand that there are many algorithms to solve real-world problems and learn to understand the advantage of using good algorithms. Students should also be encouraged to apply logical reasoning (in terms of cause/effect) instead of simply guessing and be able to create and manipulate digital content with the idea that students have control over their own digital product, and that this can be done in several ways.

In Key Stage 2, students learn how to design, write, and debug programs for diverse contexts using visual or graphical environments and to understand basic structures of computer programs (sequence, selection, and repetition) with the help of flow diagrams. They also need to understand what inputs and outputs are and how they can be used in controlling or simulating physical systems, such as programmable toys. Students at this age need to understand what a variable is and what the effect is when changing a variable in a program or a simulation. To further their logical reasoning skills, students need to be able to explain how simple algorithms

work, regardless the starting situation. They also need to develop a basic understanding of computer networks.

In Key Stage 3, students have to design, use, and evaluate computational abstractions that model the state and behavior of real-world problems, because in this way it becomes clear that computation provides concrete insights in natural and artificial systems. In addition, they need to further their understanding that there are often several algorithms that can solve the same problem and need to be able to compare different algorithms on a variety of indicators (e.g., efficiency, flexibility, generalization). Furthermore, students need to become acquainted with at least two programming languages (of which one is textual) and learn to use appropriately data structures and procedures. In addition, they develop a more in-depth understanding of how computers work and how they communicate with each other. Finally, they experience the interconnected nature of the different aspects of computing through the use of extended projects, extending the knowledge and skills developed at Key Stage 2.

Finally, attention for computing is still compulsory in Key Stage 4, but the content is less prescriptive. The rationale is that students should be able to make an informed choice to further specialize in different aspects of computing. Students may either choose to prepare for academic routes or for continuation in vocational education. Also when they do not plan to continue studies in computing, they need to get the opportunity to further develop the creativity, capability, and knowledge acquired in the previous key stages. The ultimate aim is to provide students with opportunities to continue studies in the field of computer science or to continue in a professional career.

49.2.2 Computational Thinking in the Netherlands

Currently, there is no official national policy in relation to computational thinking in the curriculum of the Netherlands, and CT does not have an explicit position in the Dutch curriculum. The Royal Netherlands Academy of Arts and Sciences (an advisory body to the Dutch government) published a report on digital literacy in 2013, which caused much discussion in the Netherlands. The Academy referred to digital literacy as the ability to make prudent use of digital information and communication and to evaluate the consequences of that use critically (Lenstra et al. 2012). They consider computational thinking as part of the digital literacy skills along with critical thinking skills when using Information and Communication Technologies (ICT), and applying (ethical) rules when using ICT, including notions of privacy and security. The Academy related digital literacy to general secondary (havo) and pre-university (vwo) levels and concludes that information science and informatics as subjects have a marginal status in the current curriculum, their quality insufficient, and their content outdated. They argued that the current teaching of digital information and communication in secondary schools should be completely revised;

otherwise, the Netherlands will lag behind. The Academy made four recommendations to the Minister of Education, Culture and Science:

(a) Introduce a new compulsory subject information and communication in the lower years of havo and vwo, as a broad and compact introductory subject, covering the essential facets of digital literacy
(b) Completely redesign the optional subject informatics in the upper years of havo and vwo
(c) Encourage interaction between these subjects and other school subjects
(d) Make it a priority to raise a new generation of teachers with new skills and attitudes

The Academy's report led to discussions on several issues, including whether digital literacy (and thus CT) should only be part of informatics and whether digital literacy was equally important in prevocational and primary education. Based on the report of the Academy, the Dutch Ministry of Education asked the Netherlands Institute for Curriculum Development to carry out two studies: one on the outdatedness of the subjects information science and informatics and another broader study on the twenty-first century skills, of which computational thinking can be seen as a critical aspect.

The intention of the study on the current state of information science and informatics was to see whether the current upper secondary school teachers that teach information science and/or informatics agreed that these subjects were outdated. Looking at the results related to CT, the study concluded that teachers did not think there was enough attention to CT as a thinking skill, and also that there was not enough time allocated in the curriculum to teach computational thinking skills (Tolboom et al. 2014). The teachers did see opportunities in the current curriculum, but many indicated that CT should have a more explicit position in the curriculum and that CT should be part of teacher training and professional development programs.

The second study (Thijs et al. 2014) was carried out in the context of primary and lower secondary education to examine whether twenty-first century skills were currently part of the formal curriculum (the core objectives for all subjects that should be taught at school and the common framework of reference for language and mathematics), the intended curriculum (analyzed by studying existing learning materials), and the implemented curriculum (analyzed through a questionnaire for teachers and case studies in schools). Computational thinking was seen as a subskill of digital literacy (one of the twenty-first century skills). The results suggested that teachers in primary and lower secondary education were familiar with twenty-first century skills and found it was important to pay attention to it; however, it got the least attention. CT at primary schools was almost nonexistent, except for a small number of schools that experimented with programming languages. Some attention to CT was paid in lower secondary schools, but compared to other twenty-first century skills, it scored very low. The case studies exhibited that even though teachers

were familiar with the twenty-first century skills, the intentions to do something in their classroom lacked focus. Thijs et al. (2014) concluded that to support teachers in integrating these skills in their teaching practice, four types of support were important: curriculum development (specification of the skills in the curriculum and exemplary teaching materials), testing instruments (development of useful frameworks and tools for monitoring and assessing the pupils), professional development (wide range of professional development activities and networks of school for knowledge sharing), and more learning resources (more attention to the skills in regular teaching materials and a broader access to additional open learning materials). It is also recommended that all twenty-first century skills become more visible in the nationwide curriculum framework, including CT.

Even though computational thinking does not have an explicit position in the Dutch curriculum, the Academy's report in 2013 and the subsequent studies in 2014 helped shape the discussion on this subject. Recently a national curriculum debate on the rationale, aims, objectives, and content of education has started in the Netherlands. The main question during this debate is: "What knowledge is most important to learn in basic education, and why/for what purpose?" It is believed that computational thinking and digital literacy will be an important part of this discussion.

49.2.3 USA: Fostering CT in K-12

In the United States, the CS principles course curriculum framework for high school students has been developed to expose K-12 students' computational thinking competencies. The framework proposed six computational thinking practices to "help students coordinate and make sense of knowledge to accomplish a goal or task" (College Board 2014, p. 2). The six computational thinking practices include:

- *Connecting Computing*: This CT practice relates to the influence of computing and its implications on individuals and society. Students are expected to:
  - "Identify impacts of computing;
  - Describe connections between people and computing; and
  - Explain connections between computing concepts" (College Board, p. 4).
- *Creating Computational Artifacts*: Given the creative nature of computing, this practice allows students to engaging in computing by designing and developing computational artifacts. Students are expected to:
  - "Create an artifact with a practical, personal, or societal intent;
  - Select appropriate techniques to develop a computational artifact; and
  - Use appropriate algorithmic and information-management principles" (College Board, p. 4).

- *Abstracting*: The third CT practice of the course focuses on students' understanding and applying abstraction "to develop models and simulations of natural and artificial phenomena, use them to make predictions about the world, and analyze their efficacy and validity" (College Board, p. 4). Students are expected to:
  - "Explain how data, information, or knowledge is represented for computational use;
  - Explain how abstractions are used in computation or modeling;
  - Identify abstractions; and
  - Describe modeling in a computational context" (College Board, p. 4).
- *Analyzing Problems and Artifacts*: Computational thinking also involves developing solutions, models, and artifacts for problems as well as evaluating the appropriateness of the proposed solutions and artifacts. Students are expected to:
  - "Evaluate a proposed solution to a problem;
  - Locate and correct errors;
  - Explain how an artifact functions; and
  - Justify appropriateness and correctness" (College Board, p. 5).
- *Communicating*: The CS principles course proposed communication (both written and oral) as an important CT practice that allows students to describe the influence of technology and computation supported by data visualizations and computational analysis. Students are expected to:
  - "Explain the meaning of a result in context;
  - Describe computation with accurate and precise language, notations, or visualizations; and
  - Summarize the purpose of a computational artifact" (College Board, p. 5).
- *Collaborating*: Finally, collaboration is a key CT practice, where peers learn to work together effectively to solve ill-structured problems that use computation. Students are expected to:
  - "Collaborate with another student in solving a computational problem;
  - Collaborate with another student in producing an artifact;
  - Share the workload by providing individual contributions to overall collaborative effort;
  - Foster a constructive collaborative climate by resolving conflicts and facilitating the contributions of a partner or team member;
  - Exchange knowledge and feedback with a partner or team member; and
  - Review and revise their work as needed to create a high-quality artifact" (College Board, p. 5).

As can be seen from these three cases, there is no consensus definition of computational thinking, and what comprises computational thinking competencies is not agreed upon. For example, both the national curriculum in England and the CS principles course curriculum framework have overlapping CT concepts (such as abstraction, algorithms, and data representation) but also include different aspects

of what makes up as core computational thinking practices (such as communication and collaboration). While some of these practices are specific for the work of a computer scientist, others are general competencies relevant in many professional domains. Given the differences within the field on mechanisms to incorporate computational thinking in the classroom, the competencies model (See Koeppen et al. 2008) may prove useful in examining what we hope students learn. Specifically, Koeppen and colleagues suggested that competencies are context-specific cognitive dispositions that play a key role in addressing problems and tasks in specific domains. Computational thinking competencies could be developed in specific subject areas in primary and secondary education.

49.3 Computational Thinking Competencies

Recently there has been an increased focus on developing competencies for students that focus on real-world problem-solving abilities and go beyond measuring more general cognitive abilities (Koeppen et al. 2008). However, even though the concept of development competencies has enjoyed an increased currency in education, defining it has proved to be challenging as there are a number of fundamentally different concepts of competence (Klieme et al. 2008). Researchers have suggested that Chomsky (1965) introduced the idea of competence to describe the cognitive system underlying the linguistic abilities necessary to produce speech (Klieme et al. 2008). Weinert (1999) suggested that there are nine different ways of defining competence, including "(a) general cognitive ability; (b) specialized cognitive skills: (c) competence-performance model: (d) modified competence-performance model: (e) motivated action tendencies: (f) objective and subjective self-concepts: (g) action competence: (h) key competencies: (i) metacompetencies" (p. 6). From these definitions of competencies, Klieme, Hartig, and Rauch suggested a working definition of competencies as "context-specific cognitive dispositions that are acquired by learning and needed to successfully cope with certain situations or tasks in specific domains" (p. 9). Competence, thus, refers to both action and development, implying that every person is able to act in a specific situation, but that the nature of action is determined by the actor's (professional) biography and can be developed. This definition of competence is appropriate not only within the school context but also in the context of the future workplace.

Within this framework of competencies (as cognitive dispositions) which are necessary to solve ill-structured problems, computational thinking fits in nicely. We concur with others that computational thinking consists of problem-solving that utilizes approaches on which computer scientists heavily rely (Grover and Pea 2013; Yadav et al. 2014). We argue that computational thinking requires students to develop both domain-specific and general problem-solving skills (see also Neubert et al. in this volume for a more detailed elaboration of domain-specific and general problem-solving skills). A core issue in this respect is how to separate the cognitive activity of computational thinking from the action of merely working on a computer

or another digital device. For instance, word processing and/or creating webpages do use digital technologies; however, these do not necessarily involve the kinds of conceptualization unique to computational thinking. The EDUsummIT 2013 working group on computational thinking (Mishra et al. 2013) argued that computational thinking needed to be more than learning to program and that the application of computational thinking needs to be seen in diverse fields. An important aspect of CT was seen as the ability to augment human capabilities by learning to "manipulate" the abilities of digital technologies and beyond that to identify the appropriate technical and physical tools and understanding in how to apply multiple tools in appropriate ways to solve problems and/or develop solutions in a person/tool partnership. So how do we develop such computational thinking competencies in children and youth?

49.4 Developing CT Competence in Education

In the USA, the Computer Science Teachers Association (CSTA) and International Society for Technology in Education (ISTE) established a steering committee focused on developing not only a definition of computational thinking suited for K-12 but also how CT would manifest in the K-12 curriculum (Barr and Stephenson 2011). The committee proposed a framework for incorporating computational thinking using the nine concepts and capabilities across the disciplines from computer science to language arts in compulsory primary and secondary education settings. For example, the core computational thinking concepts and capabilities could be practiced in language arts classrooms by identifying patterns for different types of sentences (data analysis), writing an outline (problem decomposition), using similes and metaphors (abstraction), and writing instructions (algorithms). Similarly, the authors also described how the nine CT core ideas could be highlighted in secondary science classrooms, including collecting, analyzing, and summarizing data from experiments (data collection, data analysis, and data representation), building models of physics entity (abstraction), and simultaneously running experiments with different parameters (parallelization). See Barr and Stephenson for specific ideas on how to embed core CT concepts in math, science, social studies, and language arts that do not necessarily focus on programming. The key to successfully embedding these computational thinking capabilities (or competencies) requires that both students and teachers utilize CT vocabulary when describing problems and solutions (Barr and Stephenson 2011). Members of the committee also discussed a number of predispositions and dispositions as being essential dimensions of CT. Some of these dimensions included confidence in dealing with complexity, persistence in working with difficult problems, having a tolerance for ambiguity in dealing with open-ended problems, and the ability to work in collaborative groups toward a common goal. While we agree that these are key aspects of computational thinking, there is a risk of diluting the very essence of computational thinking,

making it too vague and broad and indistinguishable from other twenty-first century skills.

While the CSTA and ISTE frameworks provide a broad basis for implementing CT in the classroom, Grover and Pea (2013) argued that programming is a key tool to support cognitive tasks related to computational thinking competence. They proposed the following computational thinking elements that could form the basis for curriculum: "(a) Abstractions and pattern generalizations (including models and simulations); (b) Systematic processing of information; (c) Symbol systems and representations; (d) Algorithmic notions of flow of control; (e) Structured problem decomposition (modularizing); (f) Iterative, recursive, and parallel thinking; (g) Conditional logic; (h) Efficiency and performance constraints; (i) Debugging and systematic error detection" (p. 39–40). Similarly, the National Research Council (NRC 2010) report on the scope and nature of computational thinking suggested that computational thinking is just another language (in addition to written and spoken language, science, and mathematics) that can be used to talk about the complex processes within the universe. Mitch Resnick, a member of the NRC committee, argued that "…computational thinking is more than programming, but only in the same way that language literacy is more than writing. They are both very important. Yes, it's more, but don't minimize programming just because it's more… programming, like writing, is a means of expression and an entry point for developing new ways of thinking" (NRC 2010 p. 13).

The idea of programming to support computational thinking goes beyond just coding and syntax and includes problem decomposition and algorithmic thinking. Another NRC (1999) report defined programming as "the construction of a specification (sequence of instructions or program) for solving a problem by an agent other than the programmer… [Programming] entails decomposing the problem into a sequence of steps and specifying them sufficiently precisely, unambiguously, and primitively that the interpreting agent, usually a computer, can effectively realize the intended solution" (p. 42). We agree with this view of programming, which can include giving few commands (such as giving directions to get from Point A to Point B, making a peanut butter and jelly sandwich, etc.) to writing complex programs that can be executed by an agent (i.e., computer – mechanical or human).

A number of programming and computational tools can help support the development of these computational thinking competencies across the K-12 schools. Grover and Pea (2013) have argued that such environments need to have a "low floor, high ceiling" that allows beginners to dive into the programming environment without frustration, but also should provide enough depth to meet the needs of advanced students. NRC (1999) stated that two conditions are critical to programming – "precisely and primitively."

- "Precise" specifications are essential to provide assurance that the agent can determine which actions are to be performed and in what order, so that the intended result is achieved. Avoiding ambiguity is obviously crucial, but even seemingly unambiguous commands can fail. For example, "turn right" fails if the soccer players can approach the intersection from either the east or the west, so

"turn north" is preferred. Similarly, "beat" and "fold in" are not synonyms for "stir" when combining ingredients, so successful recipes use precise terminology selected with great care. An important non-technology advantage of programming knowledge is that the need for precision can promote precision in everyday communication.

- "Primitive" specifications are essential to provide assurance that the steps to be performed are within the operational repertoire of the executing agent. The programmer may understand the task as "pi times R squared," but if the executing agent doesn't know what "squared" means or how to accomplish it, then the programmer must express the task in more primitive terms, perhaps revising it to "pi times R times R." For many taxpayers, the word "qualifying" in the instruction phrase "subtract qualifying contributions" would likely fail the test for primitiveness, because they would not readily understand what the term means (p. 42–43).

A number of tools make programming accessible to students from elementary school (e.g., Scratch Jr, Hopscotch, Kodable, and Alice) to high school (e.g., App Inventor and Python). These tools promote continual use of abstract and algorithmic thinking, which are valuable skills for approaching problem-solving in one's discipline (NRC 1999).

However, as pointed out previously computational thinking is not just limited to using programming tools as the Computer Science Teachers Association (CSTA) CT teacher resource highlights. The CSTA teacher resource provides specific CT concepts and their operational definition as well as exemplars that teachers could adapt from across elementary, middle, and high school grades in multiple content areas. The teacher resource documents also include prototype learning experiences called Computational Thinking Learning Experiences (CTLEs) that resemble a lesson/unit plan. The CSTA computational thinking documents can be accessed at http://csta.acm.org/Curriculum/sub/CompThinking.html. Yadav et al. (2016) also provided some specific examples of computational thinking that could be embedded in secondary schools and that span across multiple content areas including math, science, social studies, and language arts.

49.5 Implications for Vocational Education

"Every child should have the opportunity to learn concepts and principles from Computing (including both Information Technology and Computer Science) from primary school age onwards, and by age 14 should be able to choose to study towards a recognised qualification in these areas" (Royal Society 2012, p. 31). This argument from the Royal Society is a major reason that current policies and practices around the world, not only in England, emphasize to focus on CT in secondary education; however, that attention for CT in vocational education is mostly lacking. The examples of how CT is perceived as a curriculum issue in England, the

Table 49.1 Levels distinguished in the problem-solving in technology-rich environments test used in PIAAC (OECD 2013)

Level 1:
Adults can complete tasks in which the goal is explicitly stated and for which the necessary operations are performed in a single and familiar environment. They can solve problems in the context of technology-rich environments whose solutions involve a relatively small number of steps, the use of a restricted range of operators, and a limited amount of monitoring across a large number of actions
Level 2:
Adults can complete problems that have explicit criteria for success, a small number of applications, and several steps and operators. They can monitor progress toward a solution and handle unexpected outcomes or impasses
Level 3:
Adults can complete tasks involving multiple applications, a large number of steps, impasses, and the discovery and use of ad hoc commands in a novel environment. They can establish a plan to arrive at a solution and monitor its implementation as they deal with unexpected outcomes and impasses

Netherlands, and the United States, described earlier in this chapter, also show that CT is often seen as a part of the broader domain of digital literacy. In these three examples, digital literacy is recognized as an important twenty-first century competence domain, as has also been emphasized by the European Union (2007) and the OECD (2005). Digital literacy not only incorporates CT but also information literacy and computer literacy skills (Thijs et al. 2014; Voogt and Pareja Roblin 2012). Thus it is useful to explore whether it is needed to pay attention to digital literacy in vocational education and training (VET).

In the frame of the Programme for the International Assessment of Adult Competencies (PIAAC), the OECD assessed the problem-solving skills in technology-rich environments of 16–65-year-olds in 24 countries (OECD 2013). Problem-solving in technology-rich environments was defined as "the ability to use digital technology, communication tools and networks to acquire and evaluate information, communicate with others and perform practical tasks" (OECD 2013, p. 86). The results of this study exhibited that most adults scored at the lower levels of the scale with 29.4 % scoring on level 1, 28.2 % on level 2, and only 5.8 % of the adults score on level 3 (Table 49.1 provides a description of these three levels). Although the 16–2-4year-olds scored a bit higher on level 3 (9 %), it was still low compared to levels 1 and 2 (32.4 % and 41.7 %, respectively). Quite a large number of adults could not be classified because they were not able to take the test or their basic qualifications for taking the test were too low.

Hämäläinen et al. (2014) further analyzed the PIAAC data for 11 European countries particularly focusing on adults with a VET qualification. Their findings showed that within and between the countries large differences existed in the ability of VET adults to solve problems in technology-rich environments. Although the countries differed in their VET systems, either school-based or workplace-based, no systematic differences in scores related to these two VET systems could be observed. In all countries, VET trained adults scored significantly lower on the test than adults

with at least an upper secondary education degree. In five of the eleven countries studied, adults with a VET qualification scored lower than adults with at least a lower secondary education degree. The authors concluded that additional studies are required to better understand how learning VET experiences can prepare people for the needs of twenty-first century workplaces. As Hämäläinen et al. (2015) argued "we need new ways to enhance the quality of problem-solving in technology-rich environments to respond to the needs of working life and to empower VET adults' professional development" (p. 46).

These results suggested that attention for digital literacy is not only an issue for discussion related to the secondary education curriculum, but that digital literacy should also be discussed in the realm of VET. The PIAAC study demonstrated that many adult learners are at risk when they need to solve problems in technology-rich environments and adults with a VET qualification even more. We argue that computational thinking as an inseparable part of a digital literacy should also be an important competence domain within VET. Given the role of computing in the (working) life of today's citizens, the competence to solve problems in technology-rich environments is of paramount importance.

49.6 Conclusions

In summary, we agree with the views of the NRC (2010) and the Royal Society (2012) that computational thinking is a broadly applicable competence domain, which is important for individuals to be successful in today's technological society, to increase interest in information technology, and to support inquiry in other disciplines. Furthermore, attention for CT in compulsory education also leverages the opportunity for students to become interested in information technology or computer science as a potential area of interest for further studies or a future career. Given that computational thinking has been highlighted as an ubiquitous twenty-first century skill and the emphasis placed on the need to embed CT in primary and secondary schooling, we need to focus on better understanding how computational thinking tools support learners. The three cases of computational thinking in England, the Netherlands, and the United States provide a good starting point to incorporate; however, we need to carefully examine the role of computational thinking across a number of fields that go beyond the use of programming. The first step in this direction might be to agree upon a definition of computational thinking or at the very least to establish a common understanding of essential CT constructs that are relevant across disciplines. Such an effort would allow educators and researchers to focus on how different CT constructs relate to their disciplines and embed them in their own work. Barr and Stephenson (2011) provide a good framework on how computational thinking applies to different subject areas, but this needs to be expanded to include examples that educators can easily adapt in their classrooms. Additionally, there is a need to pay attention to CT as part of the broader concept of digital literacy in vocational education and training, as otherwise adults with

only a VET qualification may not be well prepared for the working life in the twenty-first century.

Furthermore, there is a need to focus on teacher preparation and professional development, both at the preservice and in service levels, to prepare educators to see the relevance of computational thinking to their own teachings. At the in service teacher level, we need professional development workshops that engage teachers in incorporating computational thinking in their specific subject areas rather than generic workshops that only provide a general view of computational thinking. At the preservice teacher level, one possibility is to infuse computational thinking modules into teaching methods courses to allow teacher education students to see the applicability of CT concepts to their brand of study. Yadav and colleagues (2014) found that embedding a one-week computational thinking module in a teacher education program has the potential to significantly improve preservice teachers' CT competence in general.

Finally, implementing computational thinking needs to go hand in hand with a research agenda that examines whether CT activities have the desired influence on student outcomes. However, this is challenging work as there are few valid and reliable measures that assess computational thinking without the use of programming tools. Currently, there is also limited research on how computational thinking is manifested in fields other than computer science (e.g., Sengupta et al. 2013; Wolz et al. 2011). Hence, rigorous measures need to be developed that are based upon specific computational thinking constructs, which could be used to assess CT in the context of specific subject areas in all education, including vocational and professional education and training.

References

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. doi:10.1145/1929887.1929905.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge: MIT Press.

College Board. (2014). *AP computer science principles draft curriculum framework*. Retrieved from http://media.collegeboard.com/digitalServices/pdf/ap/comp-sci-principles-draft-cf-final.pdf

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52*(6), 28–30. doi:10.1145/1516046.1516054.

Department for Education. (2013). *National curriculum in England: Computing programmes of study*. Statutory Guidelines, 11 September 2013. Downloaded from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study on December 19, 2014.

European Union. (2007). *Key competences for lifelong learning. A European Reference framework*. Retrieved November 11, 2009 at http://ec.europa.eu/dgs/education_culture/publ/pdf/ll-learning/keycomp_en.pdf

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. doi:10.3102/0013189X12463051.

Hämäläinen, R., Cincinnato, S., Malin, A., & De Wever, B. (2014). VET workers' problem-solving skills in technology-rich environments: European approach. *International Journal for Research in Vocational Education and Training, 1*(1), 57–80. 10.13152/IJRVET.1.1.4.

Hämäläinen, R., De Wever, B., Malin, A., & Cincinnato, S. (2015). Education and working life: VET adults' problem-solving skills in technology-rich environments. *Computers & Education, 88*, 28–47. doi:10.1016/j.compedu.2015.04.013.

Klieme, E., Hartig, J., & Rauch, D. (2008). The concept of competence in educational contexts. In J. Hartig, E. Klieme, & D. Leutner (Eds.), *Assessment of competencies in educational contexts* (pp. 3–22). Göttingen: Hogrefe & Huber Publishers.

Koeppen, K., Hartig, J., Klieme, E., & Leutner, E. (2008). Current issues in competence modeling and assessment. *Zeitschrift für Psychologie/Journal of Psychology, 216*(2), 60–72. doi:10.1027/0044-3409.216.2.61.

Lenstra, J. K., Barthel, P., Brock, E. O. de., Jong, F. M. G. de., Lagendijk, R. L., & Oortmerssen, G. v., et al. (2012). *Digitale geletterdheid in het voortgezet onderwijs; Vaardigheden en attitudes voor de 21ste eeuw* (Digital literacy in secondary education; Skills and attitudes for the 21st century). Amsterdam: KNAW.

Mishra, P., Voogt, J., Fisser, P., & Dede, C. (2013, October 1–2). *Advancing computational thinking in 21st century learning.* Summary report from the EDUsummIT 2013 Thematic Working group on Computational Thinking, Washington DC.

Naace., & CAS. (2014). *Joint guidance 2014. National curriculum for computing*. Downloaded from http://www.naace.co.uk/curriculum/guidance/jointnaacecasguidance on December 19, 2014.

National Research Council. (1999). *Being fluent with information technology*. Washington, DC: National Academy Press. Downloaded from http://www.nap.edu/download.php?record_id=6482 on December 12, 2014.

National Research Council. (2010). Report of a workshop on the scope and nature of computational thinking. Washington, DC: National Academy Press. Downloaded from http://www.nap.edu/openbook.php?record_id=12840 on December 12, 2014.

Organisation for Economic Co-operation and Development [OECD]. (2005) *The definition and selection of key competencies [Executive Summary].* Retrieved November 11, 2009 at http://www.oecd.org/dataoecd/47/61/35070367.pdf

Organisation for Economic Co-operation and Development [OECD]. *OECD skills outlook 2013. First results from the survey of adult skills.* Retrieved June 26, 2015 at http://www.oecd.org/site/piaac/publications.htm

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 1–11). Norwood: Ablex.

Polya, G. (1945). *How to solve it; A new aspect of mathematical method*. Princeton: Princeton University Press.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*, 351–380. doi:10.1007/s10639-012-9240-x.

The Royal Society. (2012). *Shut down or restart? The way forward for computing in UK schools*. London: The Royal Society.

Thijs, A., Fisser, P., & Hoeven, M. van der. (2014). *21e eeuwse vaardigheden in het curriculum van het funderend onderwijs* [21st century skills in the curriculum of Dutch primary and lower secondary education]. Enschede: SLO.

Tolboom, J., Krüger, J., & Grgurina, N. (2014). *Informatica in de bovenbouw havo/vwo. Naar aantrekkelijk en actueel onderwijs in informatica* [Informatics in upper secondary education. Towards attractive and current informatics education]. Enschede: SLO.

Voogt, J., & Pareja Roblin, N. (2012). Teaching and learning in the 21st century. A comparative analysis of international frameworks. *Journal of Curriculum Studies, 44*(3), 299–321. doi:10.1080/00220272.2012.668938.

Weinert, F. E. (1999). *Definition and selection of competencies: Concepts of competence*. Munich: Max Planck Institute for Psychological Research.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–36. doi:10.1145/1118178.1118215.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725. doi:10.1098/rsta.2008.0118.

Wolz, U., Stone, M., Pearson, K., Pulimood, S., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education, 11*, 2, article 9. doi:10.1145/1993069.1993073.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16. doi:10.1145/2576872.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends*. doi:10.1007/s11528-016-0087-7.