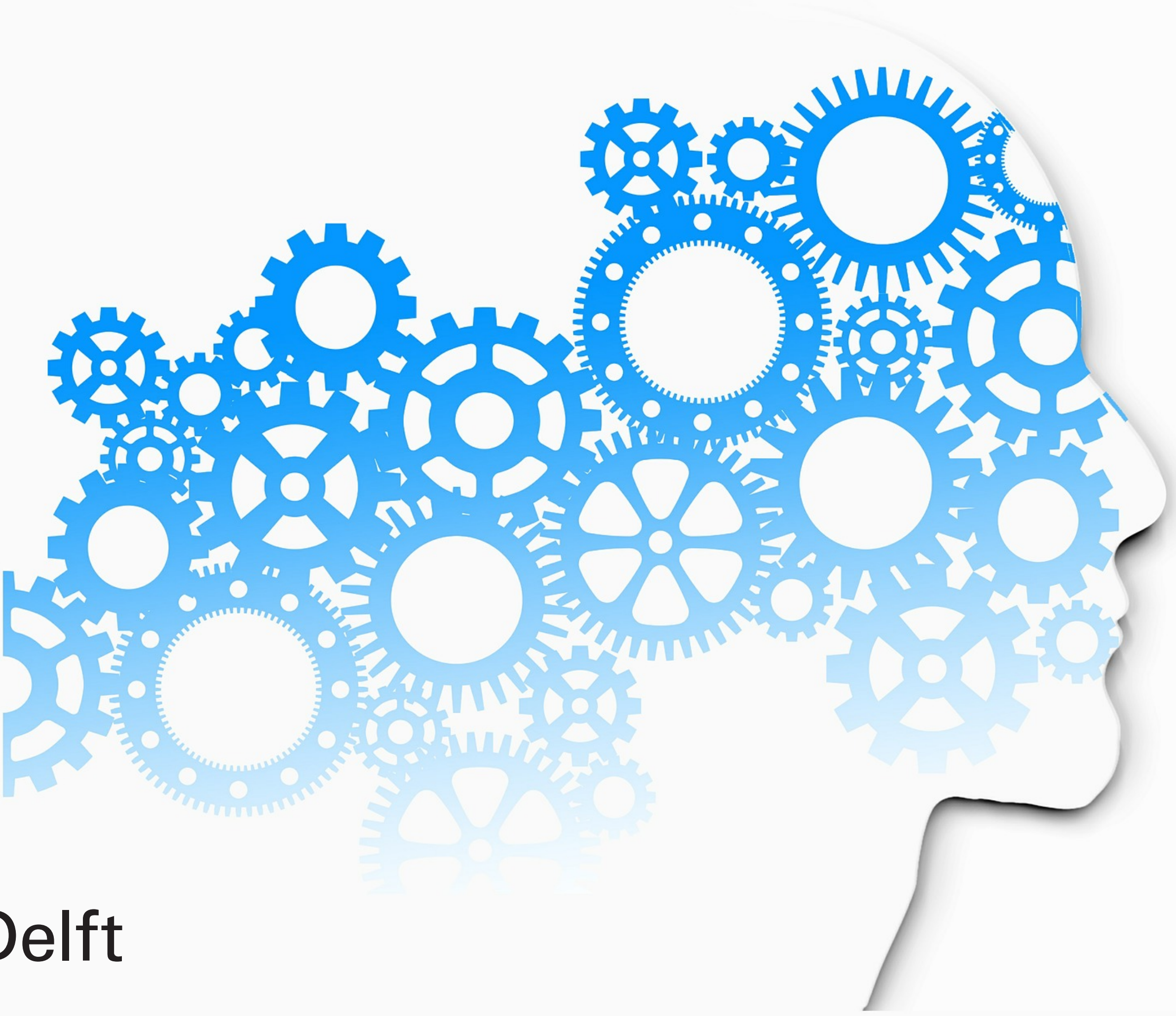
Teaching Computational Thinking: what do our educators need?

Julian de Groot



TU Delft

Teaching Computational Thinking: what do our educators need?

by

Julian de Groot

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 28, 2018 at 14:00 CET.

Student number:	4616316	
Project duration:	June 1, 2017 – June 28, 2018	
Thesis committee:	drs. M. Bruggink,	TU Delft, supervisor
	dr. ir. F. Hermans,	TU Delft, supervisor
	Prof. dr. M. de Vries,	TU Delft, supervisor

TU Delft

Preface

With pride I am able to present this thesis. It is the result of a research that I have conducted between February 2017 and June 2018 to meet the graduation requirements of the master Science Education and Communication for the subject Informatica. The research focused on teachers' needs, more specifically: the support they need teaching computational thinking concepts. The completion of this work would not have been possible without the guidance, support and help I received from many people. First and foremost I want to thank both supervisors Martin Bruggink and Felienne Hermans who, despite the often vastly different schedules, have guided me throughout each stage of the research and provided me with great advice and useful information. Operating at the frontier of science in the field of education and programming has been a great experience with you. I also want to thank co-supervisor Marc de Vries for his time and effort concerning the ethical part of the research and Johannes Hofmeister and Jeroen Spandau for helping me out with the quantitative part of my research. Furthermore I want to thank all course participants, respondents and colleagues for the input they have provided. Ever since I set the goal to become a 1st degree computer science teacher in 2009, I have studied each year next to my work to get where I am now today. There were difficult times and it was not always easy to stay motivated. All this time my family has been there for me to fall back on and provide me with the necessary support. Finally with great love I want to thank my wife for her patience and support to take the final steps in this journey I set myself on.

Julian de Groot
Middelharnis, June 2018

Summary

The educational landscape is changing to a more digitally oriented one. In this landscape digital literacy has become an important aspect for future generations. In the Netherlands a task-force, supported by the institute of curriculum development, Stichting Leerplan Ontwikkeling (SLO), has been assembled to define core objectives and learning goals for digital literacy that are to be finalized by 2019. The SLO defined digital literacy with the following four elements: ICT basic skills, computational thinking, information skills and media awareness. Computational thinking, frequently taught through programming lessons, is often considered to be the most challenging component of digital literacy. Despite all the work around planning and curriculum building, it is important that teachers' perspective, their involvement and professional development is thoroughly investigated and prepared to let them teach the concepts of computational thinking (CT) with confidence. With a limited amount of research focusing on teachers, this study aimed at identifying the needs for teachers who are getting started teaching the concepts of CT. In the first part of the study examination of teachers' TPACK took place via semi-structured interviews. Assessment of learning was identified as a major challenge/need. Teachers expressed not being able to measure how much and what a pupil has learned about CT-concepts and primarily used computer-based practical assignments as an assessment tool in a formative way. In the second part of the study a lead-up survey was created to gain more insight about the used CT assessment tools and teachers' attitude towards CT assessment. To test whether attitude towards assessment and assessment tool suitability was different for programming lessons, respondents were asked to rate statements and assessment tool suitability within three different contexts (i.e. education in general, their own subject and programming lessons). Despite a noticeable decrease in summative score and an increase in formative score for programming lessons, no statistically significant difference was found across the three contexts when measuring latent variables for formative and summative assessment. However, several statistically significant differences were found measuring the suitability of assessment tools. In terms of programming lessons, teachers rated not practical and not computer-based assessment tools significantly lower in suitability while a set of small practical computer-based assignments were rated significantly higher in suitability. Future research can focus on examining teachers' arguments with regard to the suitability scores that they have given and measuring attitude towards assessment for programming in general.

Contents

1 Introduction

1.1. Background

In a computerized and data-driven world using, understanding and creating systems that solve the problems of our future becomes especially relevant. The importance of teaching future generations the techniques to accomplish this were already clear for Papert in the 80s [60] but were later on brought back to life by Wing [80] who redefined problem-solving in a new age as computational thinking (CT).
Wing's work has been steadily gaining attention over the years from researchers who are trying to find a definition for CT, but also from policy makers. Computational thinking, often associated with programming, coding and algorithmic thinking, has been put on the agenda of the European Union as a 21st century skill but also as part of 'digital literacy' [23]. This prominent position of CT resulted in initiatives and research aimed at making computational thinking part of compulsory education by teaching children how to program [75]. Besides helping to solve future predicted shortages on the global job market, teaching every child to program and think 'computationally' also paves the way to digital citizenship and gives children a new set of tools to express themselves and use in many contexts. Guided by this awareness and importance, bringing computational thinking and its concepts into school curricula has now become a focus for many countries[9].

The foundation of curriculum development (SLO) and public organization Kennisnet, which help and contribute to changes of school curricula in the Netherlands, are helping a team[1] build a digital literacy curriculum containing core objectives and learning goals, which will be finalized by 2019. Digital literacy, according to SLO, contains four elements: ICT basic skills, computational thinking, information skills and media awareness. Research shows how schools experience difficulty bringing CT into their curricula[43]. Unfortunately, the amount of research that focuses specifically on teachers and the challenges they are facing teaching CT-concepts is still limited [10, 33, 69]. Preparing teachers to face these challenges and understand the concepts of computational thinking (i.e. learn how to link them to their own and other domains) can increase self-efficacy. Without knowing how concepts such as abstraction, generalization, decomposition, algorithmic thinking and debugging apply in the world around us, it cannot be expected of teachers to teach these things effectively and confidently. With discovered links between teacher self-efficacy and student achievement [14] it is logical to focus on the teachers, supporting them with pedagogical strategies and ways to determine a learner's development in computational thinking.

The TU Delft, as a partner of Bètasteunpunt Zuid-Holland, has started professional development projects for digital literacy that specifically focus on helping teachers teach computational thinking by letting children work on programming. Through this project teachers learn about several CT concepts and pedagogical strategies to teach CT more confidently. The study took place within the context of this professional development project.

[1]`https://curriculum.nu/ontwikkelteam/digitale-geletterdheid/`

1.2. Study goals

With an increasing shortage of teachers that are properly trained to teach CT-concepts curriculum innovation runs the risk of failing as Webb et al. state [77]. Considering the current focus on curriculum change in the Netherlands, it becomes evident proper training for teachers is required and more insight in teachers' needs and the challenges they face is necessary. This study aims at identifying needs for teachers who are getting started applying the concepts of computational thinking in their lessons. As such they can be more effectively supported by policy makers, trainers and other educators. The recommendations and conclusions of this study can be used to speed up and improve the education of CT by focusing on the teachers and their professional development.

1.3. Research Question

To reach the goals of this study the following research question was a main guideline:
What do our teachers need in order to teach computational thinking concepts?

In order to answer this question the following subquestions have to be explored:

1. What is the biggest challenge teachers are facing teaching computational thinking concepts?
2. In what way can this challenge be overcome in terms of teacher support and professional development?

2 Theoretical Framework

This chapter introduces digital literacy, its history and origin with Computational Thinking (CT) as one of its components and main focus of this theoretical framework.

2.1. A brief history of digital literacy

The term ‘digital literacy’ was first used and simplified by Gilster in his book in 1997 in which he describes it as an ‘essential life skill’ [29]. Since Gilster’s publication several researchers have been conducting extensive review on the origin of the term, trying to understand its scope and importance for education. It has since then become clear that whilst encompassing multiple literacies over time, it has also become a more ambiguous term. [8, 38]
Digital literacy as we know it today has since its first use by Gilster become a collection of several literacies: information-, computer-, library-, media- and network literacy. Analysis of the literature by Bawden [6] shows us how computer-, media- and library literacy shaped into ‘information literacy’ and later combined with computer- and network literacy would form the concept of what we know today as ‘digital literacy’.
Driven by the focus on digital literacy from the European Commission, Fraillon et al. [26] created a framework that encompasses all aspects of computer- and information literacy (CIL) with four distinct levels and descriptions. Despite its growing ambiguity researchers concluded the educational sector had to focus more on this ‘digital literacy’ to truly prepare the children of today for the more software-driven world of tomorrow [13].

2.2. Digital literacy in the Netherlands

In the Netherlands the Dutch Ministry of Education, Culture & Science (Ministerie van OCW) is tasked with the transformation of current curricula.
In 2016 the ministry created the platform ‘Onderwijs2032’ that aims to redesign current school curricula so that they become more future-proof. One of the major contributors to these curriculum changes is the institute of curriculum development, Stichting Leerplan Ontwikkeling (SLO).
The platform released an advisory report with a strong focus on digital literacy:

> *”Digital literacy should be at the core of future-oriented education. This domain has four components: information and communication technology (ICT) basic skills, information skills, media awareness and computational thinking, which helps to understand how technology works.” [58, p. 33]*

The report describes four components similar to those in Fraillon et al.’s framework [26]:

- **ICT basic skills** are required to take advantage of, understand and see the limits of the way current technology, networks and other digital systems work.
- **Information skills** are required to effectively select, find, analyze, process and use specific information from (online) resources combined with assessing its quality, usability and reliability.

- **Media awareness** is required to form a certain mindset with which a person can understand the role of the media in our society in such a way that they can be an active and critical consumer and they can act according to ethical norms involving privacy and security.
- **Computational thinking** is required to be able to (re)formulate problems in such a way that it becomes possible to solve it with computer technology. Generally but not necessarily achieved by the use of ICT-tools (hard- and software) it is combined with thinking skills that involve logical reasoning, (data) analysis, representation and pattern recognition.

The SLO has released a draft curriculum[1] based on these four components in which it describes learning goals for several stages in primary education. The draft provides structure to schools that seek help implementing digital literacy. A definitive version for both primary and secondary education has not been released yet.

Computational thinking left behind
By the time SLO's draft was released, several components of digital literacy had already been known to schools as part of the 21st century skills. However, one component in particular, computational thinking, received a lot of attention from the Royal Netherlands Academy of Arts and Sciences (KNAW). The KNAW wrote a report [45] in which it pleads for immediate integration of 'computational thinking' into current school curricula and describes it as follows:

> *"Computational thinking is a prerequisite for understanding and controlling the consequences, chances and risks of digitizing information and communication." [45, p. 18]*

Digital literacy as described by SLO is still relatively new to most Dutch primary- and secondary schools even though many of its learning goals have already been implemented into lesson programs of several subjects over the last years. The major component that many schools are struggling to implement is computational thinking [43].

2.3. Computational Thinking

Since Wing [80] first used the term 'Computational Thinking' there have been several updates by herself [79, 81] and other researchers explicating it further. In her first work Wing defines it as:

> *"Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science." [80, p. 33]*

Once her work caught more attention internationally in all types of sectors (education, politics, industry etc.) the topic was extensively explored but also interpreted in various ways. As such there came a need to make the definition of CT more precise. Wing then redefined CT as follows:

> *"Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." [81, p. 1]*

From her new definition it can be deducted that the thought processes in CT are not dependent on technology and that there is a clear focus on problem-solving skills where the actual steps towards the solution can be carried out by a person, a computer or both. Inspired by Wing's work, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) created a CT framework [18] for teachers with 'Learning Examples' for direct use in the classroom. These 'Learning Examples' are based on their developed operational definition of CT in which they describe its practices as follows:

[1] `http://downloads.slo.nl/Repository/Computational%20thinking%20PO-VO%20(versie%203-2-2017).docx`

Computational Thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics [18, p. 13]:

- *Formulating problems in a way that enables us to use a computer and other tools to help solve them.*
- *Logically organizing and analyzing data.*
- *Representing data through abstractions such as models and simulations.*
- *Automating solutions through algorithmic thinking (a series of ordered steps).*
- *Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.*
- *Generalizing and transferring this problem-solving process to a wide variety of problems.*

Further finetuning on the definition of CT took place in 2016 when the CSTA published the K-12 Computer Science Standards [19] in which they describe it as:

"A problem-solving methodology that expands the realm of computer science into all disciplines, providing a distinct means of analysing and developing solutions to problems that can be solved computationally. With its focus on abstraction, automation, and analysis, CT is a core element of the broader discipline of computer science." [19, p. 6]

Over the years, Wing's definition and viewpoints on CT served as inspiration for the CSTA and ISTE, but also for Computing at School (CAS) in the UK [39].
The frameworks of the CSTA, ISTE and CAS provide examples and pedagogical approaches for ways in which CT can be embedded across all subjects for both primary and secondary education. The importance of this implementation is further explained by Stephen Wolfram [72] as 'computational trends'. His overview of the ways in which CT-concepts play an important role all around us, was also emphasized by The Royal Society [74] defining the essence of CT as *"applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes." [74, p. 29]*

The CSTA and ISTE built their framework on nine unique CT-concepts that have been widely discussed and studied by numerous researchers over time.

2.3.1. Computational Thinking concepts and dispositions

After Wing's publication in 2006 and her redefinition of CT in 2011, a lot of research has been performed exploring the concepts and skills of CT.
One of the key works in this quest for a clear definition and CT-framework is that of Barr and Stephenson [4]. Their work is known for its model showing nine core CT-concepts & capabilities where each relationship between a CT-concept and CT-capability is explained through an example that can be used in the classroom. The concepts and capabilities are shown in Table 2.1.
Barr and Stephenson's nine core CT-concepts were used by the CSTA/ISTE working group later that year for the development of their own framework [18].
In this framework the following list of dispositions is mentioned and described as CT 'attitude' [18, p. 7]:

- *Confidence in dealing with complexity*
- *Persistence in working with difficult problems*
- *The ability to handle ambiguity*
- *The ability to deal with open-ended problems*
- *Setting aside differences to work with others to achieve a common goal or solution*
- *Knowing one's strengths and weaknesses when working with others*

Since Barr and Stephenson's publication (2011), several highly cited works with a wide scope have been released:

- Lee et al. [47] with their focus on creativity and design, introducing a use-modify-create oriented CT-framework with K-12 classroom examples on modeling & simulation, robotics and game design.

- Brennan and Resnick [11] with their three dimensional classification of CT that uses a subset of Barr and Stephenson's concepts but expands their original dispositions and attitudes with coding practices and perspectives to let learners understand others and the world around them (see Table 2.2).
- Grover and Pea [33] and Selby [66] with their extensive literature review on Computational Thinking, that tries to find consensus by structuring the concepts and dispositions from Barr and Stephenson and Brennan and Resnick.

In line with the views of The Royal Society [74], Yadav et al. [84] argue that the body of abilities, knowledge and skills for Computational Thinking can be seen as a competence. They stress its importance for individuals to thrive in today's society, its place as 21st century skill and the need to implement it in primary and secondary school curricula.

Table 2.1: Barr and Stephenson's nine core CT-concepts and capabilities

Concepts		Capabilities
Data collection	Algorithms & procedures	Computer Science (CS)
Data analysis	Automation	Math
Data representation	Parallelization	Science
Problem decomposition	Simulation	Social studies
Abstraction		Language Arts

Table 2.2: Brennan and Resnick's practices and perspectives

Practices	Perspectives
Being incremental and iterative	Expressing
Testing and debugging	Connecting
Reusing and remixing	Questioning
Abstracting and modularizing	

2.3.2. Computational Thinking as digital competence

Voogt et al. attempted to speed up implementation of CT in compulsory education by providing an outline of what should be taught and how CT should be positioned in the curriculum [75]. They argue that by adding lists of dispositions and attitude for CT, it runs the risk of ”diluting the idea of CT, blurring and making it indistinct from other 21st century skills (e.g., media information literacy).”
One year later in 2016 Gretter and Yadav present two approaches in which UNESCO's competencies for media & information literacy are merged with CT. In their work Gretter and Yadav show how CT as a competency can fit in with 21st century skills and its competencies without risking dilution as Voogt et al. mentioned.
Another year later in 2017 Yadav et al. show their way of exploring CT as an emerging competence domain [84] and state that CT and its framework of cognitive dispositions ”requires students to develop both domain-specific and general problem-solving skills”. In their work Yadav et al. also bring up the difficulty of separating CT cognitive processes from the actions involved merely working with digital devices. They explain that letting someone build webpages involves the use of digital technology but does not necessarily foster the knowledge needed in order to understand the concepts behind CT and argue, among others [25], that working on CT-competencies should be more than programming.

2.3.3. Computational Thinking and programming

Wing [80] was very specific about the characteristics of Computational Thinking:

> *”Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction.” [80, p. 34]*

However, it is by programming many of the concepts behind CT are introduced as Lye and Koh point out in their extensive review of 27 intervention studies [50]. Their general outline of the CT-concepts and dispositions show that there is a focus on problem-solving skills [4, 47] by using specific programming tools that foster CT [33, 82]. This brings us to the distinction between CT and programming. Brennan and Resnick state that CT does not necessarily require programming, but programming does provide a practical way to foster CT-concepts [11]. Despite the clarification in literature about CT being more than programming [54], the idea that they are the same persists [25]. Mitch Resnick stated:

> *"computational thinking is more than programming, but only in the same way that language literacy is more than writing." ... "programming, like writing, is a means of expression and an entry point for developing new ways of thinking." [54, p. 13]*

Programming, often being used interchangeably with the term 'coding', is the activity of solving a problem towards the design of a solution. The last 'implementation' step of writing down the instructions for the computer to execute is referred to as coding. Along with Brennan and Resnick's framework [11] Duncan et al. reason that the skills required implementing the solution (e.g. debugging and testing) is beyond coding and is part of CT, giving programming a more prominent position [22].
This higher process and skill is often referred to as "algorithmic thinking", which is needed to formulate the solution of a problem into an algorithm so that it can be implemented as a computer program [12]. Lee et al. point out how CT overlaps with elements from other 'types of thinking' (e.g. engineering thinking, design thinking, and mathematical thinking) and has a strong connection with creativity [47].
Due to research such as Lee et al.'s, programming has been included in all types of European policy documents [24, 76] and found its way to classrooms all around the world [9]. Along with this surge in interest and necessity for curriculum refinement implications for K-12 education arose [50].

2.4. Teaching computational thinking

As described in the previous sections, a surge of interest in computational thinking took place. Inspired by Wing's work [80] the National Research Council held workshops in 2010 and 2011 in which several experts explored the nature and scope of computational thinking [54] and its pedagogical aspects [55].
During the first workshop several participants suggested and validated computational thinking's close relationship with Seymour Papert's notions and work on 'procedural thinking' [60]. The different views on how Papert's notions could be built upon and how to best teach it beyond the use of computers and programming was the topic of the second workshop.

2.4.1. Educational activities

One of the starting points during this second workshop was the description of activities associated with computational thinking. The following category of educationally relevant activities, addressing 'problems' and possible solutions, were mentioned:

- Breaking big problems into smaller problems in the context of scientific models and visualizations in which students can interact with and explore visualizations and datasets similar to the way scientists explore the natural world (e.g. chemical reactions, the human genome, plate tectonics, simulations on the laws of Newton).
- Cooperatively use computational media to build, create and invent solutions to problems along with the ability to express your ideas in computational terms. (e.g. build an educational game as context for modeling of abstractions, determine the most environment friendly energy solution by constructing, evaluating and communicating about possible designs)
- Being able to apply 'systems thinking' as a way to understand and see the world as a set of interacting social and physical environmental systems in which complex problems have to be addressed. (e.g. applied along with geography and earth science using scientific visualizations of the climate to let learners reason how their behaviour affects the planet as a whole)

These categories of activities underline the application of computer science principles as a way to increase our understanding of the world around us.

2.4.2. Contexts

The National Research Council participants argued that teachers are supported best by providing plentiful domain-specific content and contexts. To promote contextualization of learning CT the following examples were given [54, 55]:

CT in an IT context:

- Troubleshooting and general problem solving with electronic devices from everyday life such as smartphones, computers and cameras that require the learner to understand the concept of system states and boundary conditions.
- Game playing and game development to gain domain-specific knowledge by practicing the modeling of abstractions.
- Working with simple models of physical phenomena to understand complex systems and basic principles of science through simulations that allow learners to experiment and tinker as a way of iterative refinement.

CT in a non-IT context, which is often referred to as unplugged[2]:

- Replay, structure and analyze processes from nature to gain a better understanding and uncover the concepts of sequencing and algorithms behind them (e.g. model the behaviour of bees in a beehive where scouts have to communicate with others to efficiently gain access to nectar).
- Discussing, analyzing and understanding user interfaces as a critical part of CT because of their specific role within a system where user input can affect program behavior while also allowing users to see program output.
- Practicing sorting algorithms, principles of cryptography through kinesthetic activities to let learners understand concepts that can affect us on a daily matter [21].
- Writing a story by iteratively writing and rewriting it, researching, collecting data, fact-check information, debug elements and loop this process collaboratively similar to that of the publishing of stories in a journalism.

Ursula Wolz is very clear on the CT concepts that can be found in journalism:

> *The similarities stem from the reliance of both fields on language. Languages can be natural as found in journalism or formal as found in computer science. Both formal and informal languages involve access to information, aggregation of data, and synthesis of information. Concepts of reliability, privacy, accuracy, and logical consistency are essential to both formal and informal languages. Both involve knowledge representation (e.g., determining the appropriate granularity for reporting a story or taking data) and abstraction from cases. [55, p. 7]*

Wolz's description on the connection with language stresses the point of using plentiful domain-specific content and teachers' context awareness to show how CT concepts apply for various domains.

2.4.3. Collaborative strategies

Both IT and non-IT context examples have in common that they can be strengthened by applying collaborative strategies referred to as 'computational participation' [40, 41]:

- pair programming, as an industry technique in the classroom to improve learners' abilities communicating concepts, reviewing and designing code.
- peer mentoring, which is often described as having a secondary teacher or class tutor. This strategy makes it possible for the teacher to share responsibility and regulate work being done through the use of peer mentors.
- physical and virtual collaboration, in which learners can effectively execute team work through digital learning environments, gather information and work on projects to form new ways of online collaboration.

[2] `http://www.csunplugged.nl/`

Collaborative strategies such as pair programming have proven to work better for learners than those that went through the learning process without it [50].

2.4.4. Tracing

Next to the set of earlier mentioned collaborative strategies, it is also important within IT contexts where writing code is a common practice, to focus on code tracing to help learners understand code. Research has made it clear that being good at programming requires the necessary tracing and explaining skills [67]. An effective way to work on these skills is by letting learners draw trace tables, storyboards, write pseudo-code and sketching out code traces that visualize a program's state to read and understand code better [20, 39].
Another option of visualizing processes and interpreting code, is by letting students create state transition diagrams and flow charts as a way of pedagogical link-making [65]. The focus on tracing, by drawing and working on 'paper' exercises to understand the concepts, is backed by Lister in his epistemology of computer programming where he shows example exercises and a waves-model describing the stages of programming reflecting the developmental stages of Jean Piaget [48]. Lister describes how learners can progress beyond a preoperational stage of programming by being guided through and shown many examples of code online. He argues that additional steps would still be required to progress towards a concrete operational and formal operational stage of programming. In this last stage a learner would be able to reason logically, consistently and systematically about any written code.
Similarly, Nelson et al. [56] show the importance of tracing skills through a novel pedagogy that focuses on teaching program semantics to explain computer behavior before writing code. Letting learners trace code and train concepts 'unplugged' before letting them write code has proven to be beneficial to the learning process [37].
Logically, a successful pedagogical strategy is not fully focused on one specific approach, but combines elements, can follow multiple pathways and can provide learners with proper accessibility and enough challenge.

2.4.5. Assessment

Collins and Halverson describe how education has entered the 'lifelong learning' era after progressing through the apprenticeship and schooling eras [15]. In the 'lifelong learning' era, the pedagogy has evolved towards a "pedagogy of reliance on interaction" where rich technological environments often serve as tutor.
Collins and Halverson argue that despite having entered this era, the means of assessment have not progressed but remained stuck in the 'apprenticeship' era. They argue that assessment mostly takes place inside rich technological often computer-based environments where learners get feedback and support to carry out tasks. Learning CT this way means that the pedagogical strategies that teachers apply shift from a teacher-centered pedagogy towards a more student-centered pedagogy [10]. Despite the wealth of potential in applying new pedagogical strategies, it has often been combined with a shift away from the traditional 'direct instruction' approach which, based on recent research [73] through an extensive analysis of 328 studies, still proves to be superior compared to other educational methods.
Logically, for any pedagogical strategy to be successful there have to be proper means of assessment. During the workshops of the National Research Council [54, 55] the participants touch on this topic and stress the importance of rich learning environments that provide necessary guidance and challenge for learners as they explore complex topics, study scientific visualizations and collaborate with others. The council promotes the use of embedded assessment in these technological environments in such a way that it provides immediate feedback for learners but also allows teachers to monitor progress real-time for targeted tutoring.
Understanding the connections between several topics is mentioned as a key element for CT and could be evaluated properly through a case study approach: learners would have to design a solution to a problem, discuss its positives and negatives, provide evidence, evaluate and debug possible prototypes towards optimization. The council thinks a lab-centric instruction would be ideal where a lot of hands-on time gives room for the necessary assessment.

Brennan and Resnick discuss the council's design-oriented approach in their framework [11] but also provide an overview of assessment strengths and limitations for other approaches. Gouws et al. provide an additional framework [31] that describes the CT skill sets and the specific levels at which they can be practiced (e.g. recognize, understand, apply and assimilate). Repenning et al. take this further by providing an embedded assessment framework through their CTP Analysis (CTPA) tool that allows measurement of performance, CT skill progression and transfer [64]. The organization Computing at School (CAS) then also released several documents discussing ways to collect evidence of learning [39], formative assessment [42] and an overview of indicators in their 'Computing Progression Pathways' that can be used as an assessment framework [17].
CAS also announced the work on 'Project Quantum'[3] that serves as an online open assessment system to help educators. In the Netherlands the assessemnt framework of Eduscrum is gaining attention. The framwork combines the team-based principles of SCRUM into an educational setting [4].
Another tool that can help teachers evaluate students work is Dr. Scratch[5], which can provide specific feedback and assign CT scores to a given Scratch project [52].
Grover et al. and The Computer Science Education Research (CSER) Group in Australia give an overview of assessment approaches and strategies, showing how various assessment tools can be used to assess student learning in a formative and summative way [16, 34]. Any educator using these approaches has a starting point for assessment. However, without agreement on strategies for teaching and analysis of methods, assessment of CT remains in an early stage.

2.4.6. Tools and CT rich environments

In the previous sections several educational activities and focus points for pedagogical environments were discussed, from which a set of criteria for CT tools and environments emerge similar to that of Repenning et al. [63]:

- easy for beginners but challenging enough for advanced learners, often referred to as: "low floor, high ceiling".
- scaffolding the process in which tasks are solved incrementally with an increasing complexity; examples are extending an existing program or debugging it.
- enables transfer in such a way that the introduced topics and connected concepts can be applied or used to explain elements from other subjects and real-life examples to support the contextualization of learning.
- systematic and sustainable in such a way that it can be used by all teachers for all students, possibly with an existing curriculum.
- The use of interactive visualizations and simulations for modeling and promoting the understanding of domain-specific contexts.
- The modeling and troubleshooting of data sets in order to gain understanding on its limitations and possibilities by fitting it to models.
- Incorporates learner knowledge to tackle common misconceptions.[6]
- Provides proper means of assessment to validate student achievement and CT development.
- Searching for patterns in large data sets seeking out its dynamics and possible emergent behavior.

For several of the items described above, researchers have investigated ways of bringing them to classrooms. Lee et al. show how interactive visualizations can shape into an educationally relevant activity [47]. With their project GUTS they propose a three-stage progression model called 'use-modify-create' engaging learners in CT.
Weintrop et al. show how the principles of CT concerning interpreting and the analysis of datasets in computational modeling can be effectively combined with learning principles of

[3]http://community.computingatschool.org.uk/resources/4382/single/
[4]http://eduscrum.nl/
[5]http://www.drscratch.org/
[6]http://www.pd4cs.org/mc-index/

science, technology, engineering and math (STEM) [78].
Repenning et al. advised to use the principles of programming and game design as starting point towards successful implementation of modeling and simulation through 'computational thinking patterns' (CTP) [64]. Basawapatna et al. reasoned that these design patterns can help with the transfer of knowledge and skills when creating STEM simulations [5].
Repenning et al. were not the first to mention this transfer of skills and knowledge. Papert's notions in 1980 on 'procedural thinking' also include the bold claim that his pupils, by programming, were able to improve their learning and become better thinkers. It is exactly this claim that inspired many researchers to investigate and measure if and how this transfer takes place to other domains. Small signs of evidence are showing up but results are unfortunately still limited [34].
There are several CT-tools that meet most of the previously mentioned criteria in various degrees: Scratch, Snap!, Alice, Kodu, Greenfoot, Agentsheets, Agentcubes, App Inventor, www.code.org and other environments that involve more tangible tools such as Lego Mindstorms, Arduino, BBC Micro:bit and the Raspberry Pi. For the younger children (e.g. K-6) working with tangible objects by programming movements of robots such as the Bee-Bot can be particularly useful to train CT-concepts. Bee-Bot allows children to physically work together on tasks or if preferred share things and collaborate virtually through an app. This possibility of approaching a task both virtually and physically in a programming environment was first used by Seymour Papert in 1967 when he introduced his programming environment Logo that used a robot turtle. Many of the environment's features are still present in modern programming learning environments that use drag and drop click-able blocks.
These block-based programming environments let users carry out activities and solve tasks with varying complexity. By using visual programming languages novice programmers can focus on the concepts, creating and experimenting rather than dealing with specific syntax rules of higher level programming languages such as Python and Java that increase cognitive load. [22, 50]

Gender bias and disabilities
Beyond the earlier mentioned criteria for educational activities and pedagogical environments, it is relevant to make sure any pedagogical approach is gender-inclusive and supports learners with disabilities. In terms of gender bias we see how, due to societal and cultural factors, girls do not get exposed equally to CT concepts and approaches [68]. Despite the teachers' challenges of understanding CT and connected pedagogy, Snodgrass et al. presented a framework to help them guide learners with disabilities [71].

2.5. Teaching CT: the challenges and unresolved issues

Despite the efforts of many researchers in establishing CT-frameworks to increase our understanding of its scope and necessity, there are challenges that need to be addressed before it can be brought into schools. In some cases these challenges involve curriculum building:

- Who a CT curriculum is for (e.g. all students or only certain educational levels [77].)
- At what age teaching elements of CT starts and in which order concepts are introduced and taught (i.e. a separate subject or as part of existing subjects [33].)
- How the curriculum provides balance in terms of teaching concepts and practices, assessing achievement and preventing gender bias [33, 78].

For any school to get started several political and legislative challenges have to be overcome that need to be addressed by national curriculum development organizations, school-boards and management. This entire process of curriculum-building, implementing and teacher involvement requires complex change and a significant amount of resources [4]. An example is England, where they focused on developing a strong curriculum first [7].

[7]https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/

2.5.1. Professional development: back to the teachers

With a curriculum as foundation to bring CT into schools, the next steps involve teachers and more specifically: their professional development. Despite the relatively large amount of research focused on definitions, frameworks and CT-related tools, the amount of research that focuses on the perspective of teachers concerning integration of CT is still scarce [10, 69]. This scarcity is problematic since a teacher's perception of CT influences the way they teach it [51]. Studies show what preconceptions pre-service and in-service teachers have concerning CT and how quickly they can be positively changed, which gives professional development an even more prominent position [7, 83]. Over the last years researchers have discovered the following problems teachers are facing teaching CT:

- providing feedback and assessing progress for many learners at once [2].
- pedagogical challenges of teaching CT concepts at certain ages [25, 48].
- inadequacies in existing digital learning environments to teach CT [82].
- a lack of systematic assessment procedures [47].
- challenges of differentiation, lack of learner knowledge and lack of resources [43, 69]
- limited ways to get certified for in-service and pre-service teachers [19, 28, 53].
- ways to collaborate with other educators as a form of community building [61].

The challenges described above, combined with the earlier mentioned preconceptions, can lead to a lack of self-efficacy to teach CT. [30, 36] show a connection between the behavior of teachers and their efficacy on teaching the subject, which makes self-efficacy crucial to improve teachers' classroom practices and student learning outcomes. With so many challenges to be met it's not strange that even England, despite having a CT curriculum, is still experiencing difficulties developing pedagogical strategies for teachers that serve learners on any level [77].

Many of the earlier mentioned problems that teachers face in England are being overcome by setting up special teacher training sessions through Computing at School (CAS). To increase the sense of community and collaboration, CAS has introduced a system of schooling through regional 'Master Teachers' that host training sessions in specific areas, supported and endorsed by companies such as Microsoft, Google, British Telecom and the Raspberry Pi foundation. Similar initiatives exist in other countries:

- Project Inria in France [8]
- www.code.org in USA [9]

Furthermore, handbooks are being released to help teachers take the first steps [10], but the amount of research evaluating these practices, methods and the influence they have on the quality of teaching CT, are still limited.

[8]https://project.inria.fr/classcode/

[9]https://code.org

[10]https://edoc.coe.int/en/internet/7515-internet-literacy-handbook.html

3 Research Method

The digital age that we are in today requires pupils to possess skills and knowledge, often referred to as computational thinking (CT), to successfully navigate and participate in society (See Chapter 2). England leads by example with putting CT into curricula of primary schools and upwards. However, the Netherlands, like many other countries, is still in the process of preparation and planning with regard to curriculum change [9]. Besides offering an optional subject 'Computer Science' in upper-secondary school there is no similar equivalent offering CT in lower-secondary school and primary education. To change this outlook the Dutch government is stimulating and supporting schools to reform curricula in several ways:

- The Kennisnet Foundation released a book with a lot of practical information concerning the introduction of digital literacy (including CT) into schools.[1]
- Curriculum.nu is working on a digital literacy curriculum with core objectives for CT that will be finalized in 2019.[2]

3.1. Relevance and objectives

Many schools in the Netherlands do not want to wait until the core objectives for digital literacy are set and autonomously have been working on ways to let CT be part of their curricula despite the challenges that are faced[43]. These perspectives have resulted in a large demand for professional development and a need to find out what teachers exactly have to learn and know in order to effectively be able to teach CT concepts [3]. To uncover these needs this study focuses on answering the following research question:
What do our teachers need in order to teach computational thinking concepts?

The research objective is to make recommendations for curriculum makers, university course builders and educators in order to develop a better study program and provide means for teachers to teach CT-concepts more successfully. The TU Delft Science Education & Communication (SEC) department and the regional Beta Steunpunt Zuid can directly benefit from the results of this research.

3.2. Research design

Following a sequential mixed methods design, qualitative data was first gathered and interpreted before quantitative data collection took place (See Chapter 4 and Chapter 5 respectively). A general purposive sampling strategy was used: teachers with a various amount of experience teaching CT-concepts were approached. For the qualitative part of the study the initial sample focus was aimed at teachers with a limited amount of experience teaching CT-concepts to get a broad view of the challenges that they face during the first experiences of

[1]`http://bit.ly/2CTOMEV`

[2]`https://curriculum.nu/wp-content/uploads/2018/01/2594.00.029_Curriculum.nl_Roadmap-A3-formaat_geactualiseerd_15012018.pdf`

teaching CT. To improve response rate, for the quantitative part of the study the sample focus was increased to include teachers with any amount of experience teaching CT-concepts.

3.3. Ethics

To ensure the integrity and quality of the research, a proactive approach was used following the ethics principles[3]. Meetings with supervisors were frequently set up (e.g. weekly or bi-weekly) in which each step was evaluated and challenges were discussed.
For each stage of the research the concepts of valid consent[4] were applied in the following ways:

- Approval to approach participants was asked through supervisors.
- Participants were fully informed before participating.
- Participants could stop participating in the research at any point preventing any form of data collection.
- All interviews were recorded and transcribed verbatim.
- All collected data was anonymized, stored on encrypted online volumes and protected by two-factor authentication.

As final part of an ethics instructional module, research integrity, possible weaknesses and moral dilemmas that were faced during this study will be discussed in Chapter 6.

[3]http://www.ethicsguidebook.ac.uk/Key-ethics-principles-15
[4]http://www.ethicsguidebook.ac.uk/Consent-72

4 Qualitative methods and data

The goal of this research was to find out what teachers need in order to teach CT concepts effectively. To reach this, teachers were questioned via semi-structured interviews. To know what questions to ask, exploratory interviews were conducted. The input from the exploratory interviews lead to the design of a professional development course. Information from the interviews and observations of the course lead to the creation of an interview format. This format was used to question four participants that enrolled in the course.

4.1. Exploratory interviews

The four teachers that were approached had experience teaching CT-concepts and/or were responsible in their school for the design of a curriculum that encompasses CT-concepts. Two teachers already used a curriculum or a set of learning goals and described the challenges they are facing. The other two expressed being occupied with curriculum design and explained the challenges they are facing designing it (i.e. in accordance with to the challenges in section 2.5. In terms of professional development all teachers expressed that any professional development aiming to teach CT-concepts effectively should focus on the following elements:

- Providing plentiful content knowledge explaining what CT is and explicating its relevance.
- Providing learning goals for schools to use for pedagogical strategies and curriculum design.
- The use of small steps, plenty of examples and simple tools for maximum accessibility and opportunities so teachers can implement what they have learned immediately.

These elements were used as guidelines for the design of a professional development course.

4.2. Professional development course

Highlighted elements from the exploratory interviews were used to design a professional development course. Twenty-two teachers enrolled in the the course consisting of four training sessions. It was held in the period between September and November 2017. Each session focused on didactics and pedagogical strategies, giving participants tools and practice activities to start teaching CT concepts right away (e.g. code tracing, debugging, remixing & reusing, pair programming, unplugged and live coding). Every participant started out with the accelerated introductory Computer Science course from the website www.code.org[1] with the option to switch to a higher level by executing tasks with the visual programming language Snap! from the 'Beauty and Joy of Computing' (BJC) curriculum created by the University of Berkeley[2]. Beyond this two-tier approach all participants worked with CT-concepts in a more creative way by following parts of the 'Creative Computing' guide developed by the Harvard

[1]https://studio.code.org/s/20-hour

[2]https://bjc.edc.org/

University ScratchEd research team.[3] Each session included homework and participants could stay in touch and communicate through an online Google group to discuss progress and ask questions. At the end of each session participants received exit cards to answer several questions. The answers were used to get a better understanding of the teachers' needs in comparison to existing research results from Kennisnet[4].
Beyond the similarities with the research of Kennisnet the following teacher needs emerged:

- Improve knowledge on CT-concepts and in what order to introduce them to learners.
- Knowing how to teach CT-concepts to a large group with mixed abilities.
- Having more and better teaching tools with proper means of assessment and easy ways to monitor progress of CT-learning.

In general participants of the professional development course expressed difficulty to describe their needs due to their limited amount of knowledge and experience teaching CT-concepts.

4.3. CT TPACK coding scheme and interview format

To prevent negative effects of the paradox of socratic ignorance (e.g. a participant not knowing what they do not know) for the semi-structured interviews a TPACK framework was used to uncover participants' knowledge gaps (e.g. implicit needs). The combination of a CT TPACK framework [3] and the needs from the professional development course were used to create an *a priori* coding scheme and an interview format with questions that let participants describe their needs and TPACK for CT. CT TPACK is defined as an extension of the PCK model by Angeli et al. [3]:

> *"TPCK is conceptualized as a unique body of knowledge that is formed by the contribution of five distinct knowledge bases, namely, content knowledge (CK), pedagogical knowledge (PK), knowledge of learners (LK), knowledge of the educational context (CX), and technology knowledge (TK). This body of knowledge grows when teachers are engaged systematically in useful educational practices, either in their own classrooms or teacher professional development programs." [3, p. 53]*

Findings from each interview were used to refine the coding scheme and the interview format. To validate the coding scheme, Krippendorff's alpha method was used for intercoder reliability assessment, which took place between the first and the second interview. The minimum acceptable level of reliability was set at 0.667 due to the exploratory nature of the research [49]. A lower alpha value was found (0.6050) due to the nature of the data (e.g. ambiguity in terms of unit start & end and overlap in used codes). The detailed calculation output can be found in Appendix B. Further reliability of coding was assessed through peer review between each interview. The changes made in the coding scheme as a result of the peer review can be found in Appendix C. The final version of the interview format and coding scheme can be found in Appendix A and Appendix D respectively. For analysis of the interview transcripts the final coding scheme was used.

4.4. Semi-structured interview results

Introductory questions showed how all participants:

- were teaching CT-concepts in secondary schools.
- had a STEM-background: three participants had a teaching degree for a STEM-subject, one participant had no teaching degree but worked as a technical assistant.
- had limited experience teaching CT-concepts (e.g. one year or less than one year).
- represented a total of three schools: in one school CT-lessons were offered to all children while the other two schools offered it to higher-level classes in the first and second year only.
- expressed how the professional development course had an immediate positive impact on the type of material offered during their lessons and the confidence with which they teach CT-concepts.

[3]http://scratched.gse.harvard.edu/guide/
[4]https://www.kennisnet.nl/fileadmin/kennisnet/digitale_vaardigheden/programmeren_maken/bijlagen/Programmeren_nog_een_moeilijk_verhaal_voor_scholen.pdf

4.4.1. Participants TPACK

The interviews in which the participants described their CT TPACK showed the following results for each of the five knowledge bases:

Content knowledge (CK)

- CT was described close to its generic description: problem-solving.
- In terms of algorithmic thinking, abstraction and decomposition were mentioned but decomposition was emphasized.
- Skills, attitude and connected pedagogical strategies for debugging were described in detail (e.g. stimulate learners to approach a problem in various ways through trial and error).

Learner knowledge (LK)

- Difficulties with abstraction and decomposition were described (e.g. the pupils' ability to split a problem into smaller pieces and see how previously written code can be used in another context).
- Difficulties with debugging were described (e.g. pupils struggling to interpret an error message and determine where it goes in the flow of a program).

Pedagogical knowledge (PK)

- Three participants favored a primarily student-centered approach based on a more constructivist 'whole language' vision [48] whereas one participant expressed applying both a teacher- and student-centered approach due to the professional development course.
- Pedagogical strategies not necessarily connected to CT were emphasized (e.g. asking questions to promote understanding)
- Didactics focused around a computer-based tool or platform that make pupils go through a series of small practical tasks that teach CT-concepts in each lesson.
- Participants taught several unplugged lessons[5] due to the professional development course.
- Two participants used the professional development course to improve and adjust the learning goals in both the curriculum and learning activities. The other two participants used the course to get started.

Technical knowledge (TK)

- Participants expressed how they use online platforms and computer based (technical) tools to foster learning CT-concepts.
- The following platforms or tools were used: Scratch, Code.org, RoboMind, GameMaker, Human Resource Machine and Arduino.
- A proactive attitude in terms of learning and adapting to new technologies was expressed.

Context knowledge (CX)

- Three participants expressed having started to teach CT-related lessons due to the perceived positive impact of CT on a societal level.
- Three participants described how the schools' vision as a 'Technasium' had a large influence on the popularity of CT on an organizational level.
- Ways of connecting CT-concepts with other subjects were described by three participants. From these three participants one mentioned connections with physics. The other two mentioned involving companies to enrich the curriculum and relevance of CT within their school.
- All participants described how they believe CT is necessary to understand and participate fully in a technology driven world.

[5]`http://www.csunplugged.nl/`

4.4.2. TPACK gaps

The interviews in which the participants described their CT TPACK uncovered the following gaps:

- **[CK]** Everyone expressed a strong desire for more subject knowledge.
- **[LK]** None of the participants were able to express their pupils' specific pre- or misconceptions to which they adapted their teaching methods and lesson material.[6]
- **[PK]**
  - Two participants expressed a desire for more lesson material, used the course to get started and expressed greatly depending on a CT-curriculum before being able to teach CT-concepts effectively.
  - All participants expressed difficulties dealing with the differences in skills and knowledge among the pupils.
  - One participant expressed the desire to know more about different educational learning activities for CT.
  - One participant expressed the desire to have more lesson material showing the relevance of CT through real life examples.
  - One participant wondered what pupils learned at the end of the lesson series.
  - Beyond checking the progress through computer-based tools all participants described not measuring and verifying for each learner if the CT-concepts were understood (e.g. assessment of learning) and expressed their inability to do so due to either a lack of time, a lack of a CT-curriculum - or both.
- **[CX]**
  - Two participants considered environments with drag and drop programming not to be real programming.
  - Three participants expressed needing more time and resources similar to the findings by Kennisnet research[7].
  - Three participants described how their pupils have a severe lack of basic ICT skills (i.e. the ability to create an online account and formatting a text document).
  - One participant expressed the desire to stay in touch with other teachers who face the same challenges teaching CT-concepts.

The following four trends emerged for all participants:

1. A lack of subject knowledge. **[CK]**
2. A lack of learner knowledge about pre- or misconceptions. **[LK]**
3. A lack of general pedagogical knowledge to deal with differences in pupils' abilities. **[PK]**
4. No assessment of learning to verify if the CT-concepts are understood. **[PK]**

From these trends numbers 1 and 3 were explicitly mentioned by participants as a need. Numbers 2 and 4 were not, only after specifically asking about it. The assessment of learning was identified as the biggest challenge or need due to its absence.

4.5. Conclusion

In every interview participants referred to the professional development course but none of them were able to point out how or in what way their pedagogical strategies contribute to assessing which CT-concepts their pupils had actually learned. Despite the large number of assessment tools that can be used in a formative and summative way [16], all participants described primarily using computer based practical assignments in a formative way. Despite not being able to establish what had truly been learned by their pupils, two participants mentioned grading the progress of the computer-based practical assignments anyway. The gaps in content knowledge and pedagogical knowledge could have influenced participants' beliefs about CT assessment and the suitability of CT assessment tools. A lead-up survey was designed to gain more insight into both the CT assessment tools and teachers' attitude towards CT assessment for learning (formative) and CT assessment of learning (summative).

[6]http://www.pd4cs.org/mc-index/

[7]https://www.kennisnet.nl/fileadmin/kennisnet/digitale_vaardigheden/programmeren_maken/bijlagen/Programmeren_nog_een_moeilijk_verhaal_voor_scholen.pdf

5 Quantitative methods and data

The goal of this research was to find out what teachers need in order to teach CT concepts effectively. To reach this, teachers were questioned via semi-structured interviews from which the assessment of learning was identified as the biggest challenge or need, due to its complete absence. Furthermore, interview participants described predominantly using computer-based assessment tools in a formative way. A survey was designed to generalize findings and gain insight into CT assessment tools and attitude towards formative and summative assessment for CT. Throughout the survey the term ’programming lessons’ was used instead of CT. This decision, the question options and layout are discussed in Chapter 6. The survey can be found in Appendix E.

5.1. Respondents

Teachers were asked to participate through the supervisors’ social networks, computer science sites and news bulletins. To capture attitude towards assessment for CT from a larger audience, teachers were invited from both primary and secondary education. These teachers indicated that they were either about to start or already teaching programming. The total number of respondents was 54. Table 5.1 and Figure 5.1 show the distribution of the respondents per age category and the gender distribution per age category.

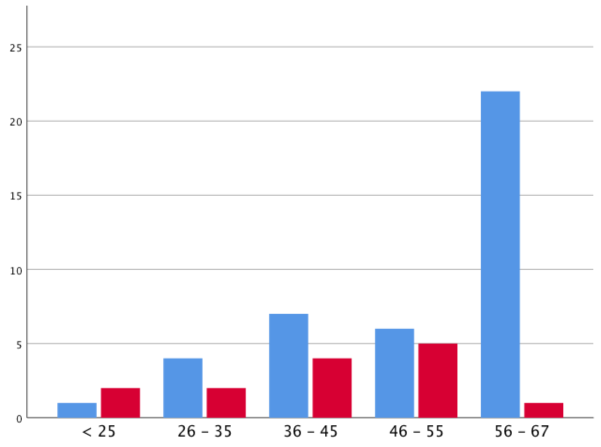


Figure 5.1: Age distribution by gender
(male in blue and female in red)

	Frequency	Percent
<25 years	3	13.65
26 - 35 years	6	11.1
36 - 45 years	11	20.4
46 - 55 years	11	20.4
56 - 67 years	23	42.6
Total	54	100.0

Table 5.1: Age distribution

5.2. Survey design

In the first part of the survey teachers were asked to provide some background information such as gender, year of birth, teacher degree, subject and years of experience teaching programming lessons. In the second part of the survey they were asked to what extent they agreed with statements regarding assessment attitude and rate assessment tool suitability. The five-point Likert scales that were used ranged from strongly disagree to strongly agree and very unsuitable to very suitable, respectively. To be able to capture assessment attitude, a series of statements related to formative and summative assessment were created inspired by comparisons made from existing research [35, 46, 70] and peer review. Respondents were asked to rate these statements and assessment tool suitability within three different contexts as part of a 'repeated measures' design to test whether assessment attitude and assessment tool suitability is different within the context of programming lessons. The three different contexts implemented in the survey were:

- Education in general
- The subject they feel most comfortable teaching
- Programming lessons

In the third part of the survey teachers were asked how frequently they used each type of assessment tool for their programming lessons on a five-point Likert scale (e.g. from never to often). In the fourth and final part of the survey they could leave notes, sign up to receive research results and were asked to finish a sentence stating what they think would help assessment for programming lessons the most.

5.3. Survey results

The data from the survey was analyzed through IBM's statistical software SPSS. Friedman tests were executed to determine if there were any statistically significant differences between different contexts. The first step of the analysis involved a check whether the data passed all assumptions (i.e. ordinally scaled dependent variables, a random sample and multiple measures for the same group). Differences between 5% trim mean and mean values were found to be marginal preventing affection of outliers for further analysis [59].
Each context contained the same five statements related to formative assessment, five statements for summative assessment and six examples of assessment tools: a total of sixteen questions per context.
Table 5.2 shows per context, by measuring Cronbach's alpha, whether the five formative and five summative questions reliably measure the same latent variable (e.g. attitude towards formative and summative assessment). Except for the 'questionable' formative latent variable within the educational context, all values are considered to range from acceptable to good [44].

Table 5.2: Latent variable Cronbach alpha values

Context	**Latent variable with alpha value**
Education in general	Formative: 0.654 Summative: 0.792
Own subject	Formative: 0.790 Summative: 0.810
Programming lessons	Formative: 0.726 Summative: 0.858

The Friedman test, as non-parametric variant of a repeated measures ANOVA, compared medians for the statements/questions for each of the three contexts. A visual overview of the data showing the differences, for every latent variable per context, can be seen in Figure 5.2. A Friedman test was carried out to compare the aggregated latent variable scores for the different contexts. Despite programming lessons' highest mean score of the formative latent variable and lowest mean score of the summative latent variable, there was no significant difference between the different contexts.The remaining six questions concerning suitability of assessment tools were not measuring a latent variable, but were seen as a family of hypotheses for which Bonferonni correction [1, 27] had to be applied to prevent false positives (e.g. an adjusted sigma: 0.05/6 = 0.0028). Wilcoxon Signed Rank post-hoc tests were performed and Kendall's coefficient (W) of concordance was calculated to assess agreement among the respondents. For each of the assessment tools a separate Friedman test was carried out. The results of these tests can be seen in Table 5.3, Table 5.4 and Table 5.5.

Figure 5.2: Latent variable mean scores per context

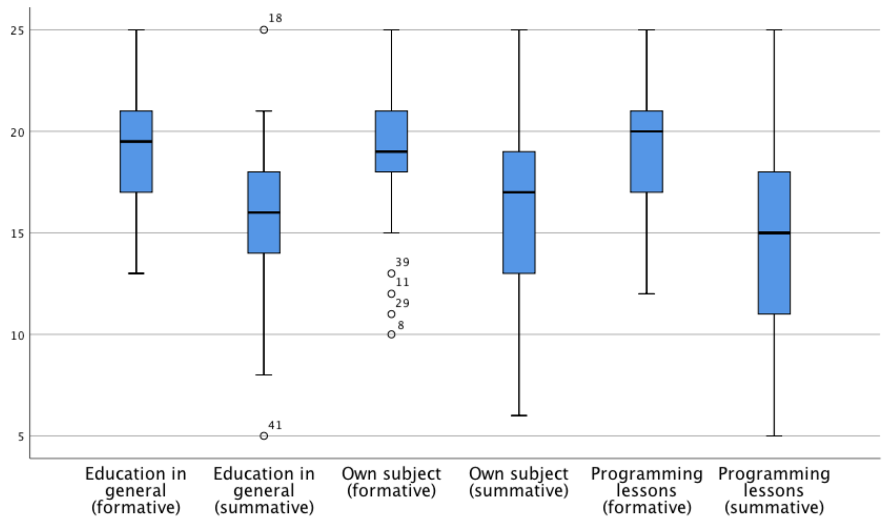


Table 5.3: Friedman test results for assessment tools without significant differences ($p > 0.0028$)

Assessment tool	**Test statistics**	**Mean ranks for education in general, own subject and programming lessons**
A digital test with open and/or closed questions	χ^2 (2) = 10.377, p = 0.006, W = 0.096	2.15, 2.12 and 1.73
A large practical assignment (project)	χ^2 (2) = 2.986, p = 0.225, W = 0.028	1.94, 1.95 and 2.11

Table 5.4: Friedman test results for assessment tools with significant differences ($p < 0.0028$)

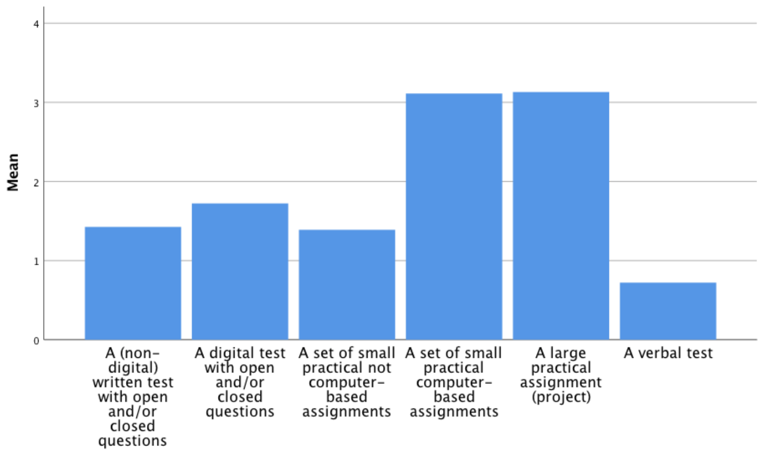
Assessment tool	**Test statistics**	**Mean ranks for education in general, own subject and programming lessons**
A (non-digital) written test with open and/or closed questions	χ^2 (2) = 41.277, p = 0.000, W = 0.382	2.35, 2.19 and 1.45
A set of small practical not computer-based assignments	χ^2 (2) = 12.687, p = 0.002, W = 0.117	2.29, 1.95 and 1.76
A set of small practical computer-based assignments	χ^2 (2) = 13.462, p = 0.001, W = 0.125	1.81, 1.91 and 2.28
A verbal test	χ^2 (2) = 33.042, p = 0.000, W = 0.306	2.49, 1.91 and 1.60

A visual overview showing how often each assessment tool was used (e.g. the mean score of each Likert-scale) can be seen in Figure 5.3. An overview of the responses on the final open-ended question can be found in Appendix F.

Table 5.5: Friedman pairwise post hoc results for assessment tools with significant differences ($p < 0.0028$)

Assessment tool	**No significant difference**	**Significant differences**
A (non-digital) written test with open and/or closed questions	Between own subject and education in general	Reduction in ratings between: - education in general and programming lessons ($p = 0.000$) - own subject and programming lessons ($p = 0.000$)
A set of small practical not computer-based assignments	Between own subject and programming lessons	Reduction in ratings between: - education in general and programming lessons ($p = 0.000$) - own subject and programming lessons ($p = 0.002$)
A set of small practical computer-based assignments	Between: - own subject and education in general - own subject and programming lessons	Increase in ratings between: - education in general and programming lessons ($p = 0.001$)
A verbal test	Between own subject and programming lessons	Reduction in ratings between: - education in general and programming lessons ($p = 0.000$) - education in general and own subject ($p = 0.000$)

Figure 5.3: Mean frequency score of each assessment tool (on Likert scale 1-5)



5.4. Conclusion

No statistically significant differences were found when measuring the latent variables for formative and summative assessment across the three contexts (i.e. education in general, own subject and programming lessons). Despite a lack of significant differences between the contexts, the programming lessons context had the highest mean score for formative assessment and the lowest mean score for summative assessment. Various statistically significant differences were found for programming lessons with regard to the suitability of assessment tools:

- A (non-digital) written test with open and/or closed questions was rated lower
- A set of small practical not computer-based assignments was rated lower
- A set of small practical computer-based assignments was rated higher
- A verbal test was rated lower

The other assessment tools (i.e. a digital test with open and/or closed questions and a large practical assignment) showed no statistically significant differences. Frequencies of the used assessment tools show an emphasis on (practical) computer-based assignments. This primary use of practical computer-based assessment tools and its perceived suitability is underpinned by results from the final part of the survey in which respondents emphasize the use of computer-based practical assignments, formative assessment and a general desire for more assessment material. Despite the various suitable tools that support both formative and summative assessment for programming [16], respondents showed a limited use and limited perceived suitability for assessment tools other than (practical) computer-based assignments.

6 Research integrity

A researcher needs to be committed adhering to the standards and ethical norms that come with responsible research conduct by applying a certain set of practices and rules [57].
Despite the extensive amount of literature and legislation based around research integrity, it remains a fact that each research comes with challenges that have to be faced (e.g. moral dilemmas) in order to keep making progress. As part of a course requirement this chapter discusses the research integrity challenges for each stage of the research from a third-person view to provide more objectivity.
The author feels that the following elements of research integrity held true for the entire research:

- Being honest and fair in the way findings are reported and the research is performed.
- Accountability and openness in sharing research methods, findings/results in an open and professional way.

6.1. Research proposal

In this stage the author was confronted with the following challenge in terms of research integrity:

- The disclosure of conflicts and competing of interest.

The author's wish to integrate several university courses into the thesis was granted but required the creation of an educational product in the form of a professional development course for Bètasteunpunt Zuid. This competing of interest with regard to the author's wishes resulted in a research strategy and objectives as described in Chapter 3. A moral issue arose when the author had to chose between the following options offered by the supervisors:

1. A research aimed at measuring and increasing teachers' self-efficacy concerning the teaching of CT-concepts.
2. A research aimed at measuring used CT-tools, knowledge and attitude regarding CT-concepts.

To stay aligned with the interests of both supervisors, a combination of both options was chosen by focusing on teachers' needs. The chosen methods and steps taken as described in Chapter 3, Chapter 4 and Chapter 5 show how teachers' self-efficacy was indirectly measured and their knowledge and attitude was measured by analyzing their TPACK. Further analysis of used tools and attitude took place in a lead-up survey as discussed in Chapter 6.

6.2. Theoretical Framework

During the creation of the theoretical framework the author was confronted with the following challenges in terms of research integrity:

- Respect and authorship so that credit is properly given to those that contributed to the research.
- Responsible publication, fairness and proficiency in terms of authorship and peer review.

The author had to combine pieces of literature fast enough to present them correctly during the professional development course. To preserve quality and validity during this process extensive peer review was performed in which pieces of the theoretical framework were discussed. To prevent plagiarism the author used APA style guidelines for all used literature and citations.

6.3. Exploratory interviews

In this stage the author was confronted with the following challenges in terms of research integrity:

- Ethical/responsible research when animals or humans are involved by complying with the rules and law.
- Respect and authorship so that credit is properly given to those that contributed to the research.
- Responsible publication, fairness and proficiency in terms of authorship and peer review.
- Accuracy in the treatment of data so that there are clear records of all data and material and confidential materials are curated properly.
- Objectivity in data examination and analysis preventing bias and the disclosure of conflicts and competing of interest.

Several interviews were conducted with participants chosen by the supervisors for their experiences with CT curriculum design or the teaching of CT-concepts. For these interviews the principles of valid consent were followed as described in section 3.3. Despite the descriptive nature of the data collected from these teachers, their input was used for the design of a professional development course.
A more open method involving the random approach of teachers (e.g. not known by the supervisors) could not be chosen due to the severe time constraints it would have created. In terms of respect and authorship a conflict of interest arose for participants who preferred to be mentioned in the thesis, but also wanted to remain anonymous. The author decided to opt for full anonimization in the final work but gave credit to those due in the preface.
Between each exploratory interview their findings were peer reviewed. After all exploratory interviews took place a summary of findings was reviewed once more (e.g. recorded parts of the interviews were played back) to verify the findings before using them as guidelines for a professional development course.

6.4. Professional development course design

In this stage the author was confronted with the following challenges in terms of research integrity:

- Ethical/responsible research when animals or humans are involved by complying with the rules and law.
- Accuracy in the treatment of data so that there are clear records of all data and material and confidential materials are curated properly.

The results of the exploratory interviews as described in Chapter 4, served as an input and starting point for the design of the professional development course. In the exploratory interviews participants were asked what they think would be important for any professional development course to be successful. Based on the feedback from them, the educational activities, contexts and strategies from the theoretical framework (See Chapter 2) new lesson material would have had to be created that would not only let learners experience programming but would also provide a balance between the practical and theoretical side (e.g. practicing skills and learning concepts). Due to time constraints this could not be fully realized before the start of the professional development course: selections from existing lesson material were combined with newly created material to maintain balance between theory and practice. After registration for the course closed, it became clear that video recording on-site would be possible. However, to prevent last-minute dropouts with regard to a new form of consent the recording was canceled. Instead, for each session several notes were taken in the form of memo's.

6.5. Course observations

As described in Chapter 1 the study took place within the context of a professional development project from Bètasteunpunt Zuid-Holland. The author was confronted with the following challenges in terms of research integrity:

- Accuracy in the treatment of data so that there are clear records of all data and material and confidential materials are curated properly.
- Objectivity in data examination and analysis preventing bias and the disclosure of conflicts and competing of interest.

During this stage of the research a conflict of interest was to observe and write memo's as an 'outsider' while also co-presenting several of the professional development course sessions. It is possible that during the co-presenting and answering of teachers' questions some elements were not observed that would have been put into the memo's otherwise. However, one can argue by deciding to take the role as co-presenter the author also had the opportunity to witness situations and receive certain information that otherwise would have not have taken place as observer only. For continuity of the course and to ensure objectivity, the author took both roles and peer reviewed each session's memo. For these memo's any data leading to the potential identification of individual course participants was removed or anonymized.

6.6. Semi-structured interviews and survey design

In these stages the author was confronted with the following challenges in terms of research integrity:

- The disclosure of conflicts and competing of interest.
- Objectivity in data examination and analysis preventing bias and the disclosure of conflicts and competing of interest.

Analysis of the semi-structured interview transcripts uncovered several teachers' TPACK gaps (see Chapter 4). Ideally, similar to the format of the semi-structured interviews, all TPACK elements would have been analyzed through a survey (i.e. CK, PK, LK, TK and CX) to see if the same 'gaps' are found for a larger group. A survey analyzing all five knowledge areas would have had too many questions which would have lead to dropouts and would have also imposed time constraints building it. To counter this and increase the chance of measuring an effect, further investigation of one particular TPACK gap was performed (e.g. assessment of learning).
An important part of the survey was to consider how participants are informed and how certain topics are framed. With so many people having different ideas about what computational thinking is, there was a risk of priming subjects with a definition of computational thinking so generic (e.g. problem solving) that it would skew the results.
An in-depth discussion on this topic can be found in Appendix G. The discussion itself went beyond the scope of the research but had strong implications for the way the survey participants were primed (e.g. given context before questions were asked). After extensive peer review the author decided to prime subjects with the context of 'programming lessons' rather than working with the definition of CT.

7 Conclusion

This study aimed at improving the integration of computational thinking (CT) into compulsory education by focusing on teachers' needs. To realize this, information from the semi-structured interviews was used which lead to the identification of explicit and implicit needs, from which assessment attitude and tools were investigated further with a lead-up survey. Based on the results of this study several recommendations were formulated. The research question that served as main guideline of this study was:

"What do our teachers need in order to teach computational thinking concepts?"

7.1. Answering the research question

To answer the main research question the following subquestions are discussed below:

- What is the biggest challenge teachers are facing teaching computational thinking concepts?
- In what way can this challenge be overcome in terms of teacher support and professional development?

7.1.1. Qualitative results: identifying the main challenge

In this study teachers who had just started teaching computational thinking related lessons reported several challenges:

1. A lack of resources, similar to findings from existing research by Kennisnet [43], ranging from technical issues in the classroom to being in need of a curriculum.
2. Differentiation (e.g. how to teach a group with differences in knowledge and skill).
3. A (perceived) lack of subject knowledge resulting in less confidence to teach.
4. Teachers' pedagogical strategies (e.g. using various educational activities and knowing when and how to introduce which concepts).
5. Pupils' lack of basic IT skills (e.g. abilities to format a document or creating accounts).
6. The assessment of learning to verify if the CT-concepts are understood.

The analysis of the participants' TPACK uncovered assessment as the biggest challenge or need for teachers who have recently started CT-related lessons. The participants expressed a desire to be able to check what has been learned by their pupils in terms of CT concepts, but showed no pedagogical strategies supporting it. Instead of using various assessment tools in a formative and summative way, the teachers primarily used a set of computer-based practical assignments as assessment tool in a formative way. In some cases the progress of the assignments was graded without really knowing what concepts the pupils had mastered. To investigate whether this approach was driven by a different attitude towards CT assessment (tools), a survey was created.

7.1.2. Quantitative results: generalizing findings

The first part of the study showed how teachers who had recently started teaching CT used a specific set of assessment tools primarily in a formative way. To investigate this further a survey was designed to determine whether they have a different attitude towards summative and formative assessment for CT related lessons in comparison to their own subject and education in general. Teachers were also asked to rate the suitability of various assessment tools for each of the three contexts (i.e. education in general, own subject and programming lessons). In the survey CT-related lessons were described as 'programming lessons' due to the multi-interpretable nature of CT. No statistically significant differences were found when measuring the latent variables for formative and summative assessment across the three contexts. However, statistically significant differences were found with regard to assessment tools for programming:

- reductions in suitability for written (non-digital) tests, verbal tests and a set of small practical not computer-based assignments, and
- an increase in suitability for a set of small practical computer-based assignments.

Respondents were also asked to finish a sentence stating what they think would help assessment for programming lessons the most, which showed similar results to those that were found in the first part of the study. Furthermore, most teachers emphasized formative assessment and the use of practical computer-based assignments to teach CT-concepts in programming lessons.

7.1.3. Final conclusion

Results from this study show that the biggest challenge teachers are facing is assessing what has been learned by their pupils in terms of CT concepts. The majority of them took a student-centered pedagogical approach with a focus on formative assessment and practical computer-based assessment tools. Existing CT-frameworks emphasize a student-centered pedagogy with a design-based constructionist approach originating from Papert's work [60]. The favoring of this pedagogy over a teacher-centered one [10] was visible in the way teachers described their lessons. Moreover, they explained how they let pupils train CT concepts in a way that Lister describes as 'the bottom-up' approach [48]. Due to a lack of subject knowledge participants expressed difficulty taking a teacher-centered 'top down' approach. Besides the primarily student-centered methods this study showed a limited use of assessment tools and a perceived limited view of their suitability (e.g. a focus on formative assessment and practical computer-based assessment tools). Next to the explicitly mentioned desire for more subject knowledge it is necessary that any future professional development initiatives focus on showing teachers' pedagogical strategies that offer a balance between a bottom-up and top-down approach. Such pedagogical strategies need to be guided by learning goals connected to CT-concepts with various assessment tools (i.e. not only computer-based) that can be used in a formative and a summative way.

7.2. Limitations and future work

In this research there were several limitations that could have influenced the findings. For the first part of the study the sample of teachers was limited to teachers that had just started teaching CT. To increase reliability and validity, data could be collected from more teachers with various amounts of experience, possibly through triangulation.

During the course observations and the interviews an existing coding scheme was refined that could be used to measure and assess teachers' TPACK related to CT. Using the refined coding scheme lead to theory building and assessment as a key element for teachers' needs. Future work could use the coding scheme as a starting point and focus on multiple elements to obtain a more complete result.

One of the main elements of the survey focused on testing whether teachers would express a different attitude concerning formative and summative assessment for which, despite an increase in mean scores between both latent variables for programming lessons, no statistically significant results were found. Future research could focus on measuring attitude towards assessment for programming by exploring this increase in further detail.

Bibliography

[1] Mikel Aickin and Helen Gensler. Adjusting for multiple testing when reporting research results: The Bonferroni vs Holm methods. *American Journal of Public Health*, 86(5): 726–728, 1996. ISSN 00900036. doi: 10.2105/AJPH.86.5.726.

[2] Charoula Angeli and Nicos Valanides. Epistemological and methodological issues for the conceptualization, development, and assessment of ICT-TPCK: Advances in technological pedagogical content knowledge (TPCK). *Computers and Education*, 52(1): 154–168, 2009. ISSN 03601315. doi: 10.1016/j.compedu.2008.07.006. URL http://dx.doi.org/10.1016/j.compedu.2008.07.006.

[3] Charoula Angeli, Joke Voogt, Andrew Fluck, Mary Webb, Margaret Cox, Joyce Malyn-Smith, and Jason Zagami. A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. 19(3):47–57, 2016.

[4] Valerie Barr and Chris Stephenson. Bringing computational thinking to K-12. *ACM Inroads*, 2(1):48, feb 2011. ISSN 21532184. doi: 10.1145/1929887.1929905. URL http://dl.acm.org/citation.cfm?doid=1929887.1929905.

[5] Ashok Basawapatna, Kyu Han Koh, Alexander Repenning, David C. Webb, and Krista Sekeres Marshall. Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11*, page 245, New York, New York, USA, 2011. ACM Press. ISBN 9781450305006. doi: 10.1145/1953163.1953241. URL http://portal.acm.org/citation.cfm?doid=1953163.1953241.

[6] David Bawden. Information and digital literacies: a review of concepts. *Journal of Documentation*, 57(2):218–259, 2001. ISSN 0022-0418. doi: 10.1108/EUM0000000007083. URL http://www.emeraldinsight.com/doi/10.1108/EUM0000000007083.

[7] Nathan Bean, Joshua Weese, Russell Feldhausen, and R. Scott Bell. Starting from scratch: Developing a pre-service teacher training program in computational thinking. *Proceedings - Frontiers in Education Conference, FIE*, 2014, 2015. ISSN 15394565. doi: 10.1109/FIE.2015.7344237.

[8] Daj Belshaw. What is digital literacy? A Pragmatic investigation. *EdD Thesis, Durham: University of Durham, ...*, page 274, 2011. URL http://neverendingthesis.com/doug-belshaw-edd-thesis-final.doc.

[9] Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis, and Yves Punie. *Developing Computational Thinking : Approaches and Orientations in K-12 Education.* Number June. 2016. ISBN 978-1-939797-24-7. doi: 10.2791/792158.

[10] Matt Bower, Leigh Wood, Jennifer Lai, Cathie Howe, Raymond Lister, Raina Mason, Kate Highfield, and Jennifer Veal. Improving the Computational Thinking Pedagogical Capabilities of School Teachers. *Australian Journal of Teacher Education*, 42(3):53–72, mar 2017. ISSN 03135373. doi: 10.14221/ajte.2017v42n3.4. URL http://ro.ecu.edu.au/ajte/vol42/iss3/4/.

[11] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. *annual American Educational Research Association meeting, Vancouver, BC, Canada*, pages 1–25, 2012. doi: 10.1.1.296.6602. URL http://web.media.mit.edu/{~}kbrennan/files/Brennan{_}Resnick{_}AERA2012{_}CT.pdf.

[12] Peter Brusilovsky, Eduardo Calabrese, Jozef Hvorecky, Anatoly Kouchnirenko, and Philip Miller. Mini languages: A Way to Learn Programming Principles. *Education and Information Technologies*, 2(1):65–83, 1997. ISSN 1360-2357. doi: http://dx.doi.org/10.1023/A:1018636507883.

[13] C. Burnett. Technology and literacy in early childhood educational settings: A review of research. *Journal of Early Childhood Literacy*, 10(3):247–270, 2010. ISSN 1468-7984. doi: 10.1177/1468798410372154. URL `http://ecl.sagepub.com/cgi/doi/10.1177/1468798410372154`.

[14] Gian Vittorio Caprara, Claudio Barbaranelli, Patrizia Steca, and Patrick S. Malone. Teachers' self-efficacy beliefs as determinants of job satisfaction and students' academic achievement: A study at the school level. *Journal of School Psychology*, 44(6):473–490, 2006. ISSN 00224405. doi: 10.1016/j.jsp.2006.09.001.

[15] Allan Collins and Richard Halverson. The second educational revolution: rethinking education in the age of technology. *Journal of Computer Assisted Learning*, 26(1):18–27, jan 2010. ISSN 02664909. doi: 10.1111/j.1365-2729.2009.00339.x. URL `http://ocw.mit.edu/courses/media-arts-and-sciences/mas-714j-technologies-for-creative-learning-fall-2009/readings/MITMAS{_}714JF09{_}read03{_}coll.pdfhttp://doi.wiley.com/10.1111/j.1365-2729.2009.00339.x`.

[16] CSER. Supporting teachers to assess F – 10 Digital Technologies Literature review. (June):0–16, 2017.

[17] Andrew Csizmadia, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby, and John Woollard. Computational thinking A guide for teachers. pages 20–24, 2015. URL `http://computingatschool.org.uk/computationalthinking`.

[18] CSTA. Computational Thinking in K–12 Education - Teacher Resources 2nd edition, 2011. URL `https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/472.11CTTeacherResources{_}2ed.pdf`.

[19] CSTA. *[INTERIM] CSTA K–12 COMPUTER SCIENCE STANDARDS*. 2016. ISBN 9781450347624.

[20] Kathryn Cunningham, Cherry Street, Sarah Blanchard, Barbara Ericson, and Mark Guzdial. Using Tracing and Sketching to Solve Programming Problems. pages 164–172, 2017.

[21] Paul Curzon, Peter W. McOwan, Nicola Plant, and Laura R. Meagher. Introducing teachers to computational thinking using unplugged storytelling. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE '14*, pages 89–92, 2014. doi: 10.1145/2670757.2670767. URL `http://dl.acm.org/citation.cfm?doid=2670757.2670767`.

[22] Caitlin Duncan, Tim Bell, and Steve Tanimoto. Should your 8-year-old learn coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE '14*, pages 60–69, 2014. doi: 10.1145/2670757.2670774. URL `http://dl.acm.org/citation.cfm?doid=2670757.2670774`.

[23] European Commission. Communication From the Commission To the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions - a New Skills Agenda for Europe. pages 1–18, 2016. URL `https://ec.europa.eu/transparency/regdoc/rep/1/2016/EN/1-2016-381-EN-F1-1.PDF`.

[24] Anusca Ferrari. *DIGCOMP : A Framework for Developing and Understanding Digital Competence in Europe .* 2013. ISBN 9789279314650. doi: 10.2788/52966.

[25] George H. L. Fletcher and James J. Lu. EducationHuman computing skills. *Communications of the ACM*, 52(2):23, 2009. ISSN 00010782. doi: 10.1145/1461928.1461938. URL http://portal.acm.org/citation.cfm?doid=1461928.1461938.

[26] Julian Fraillon, John Ainley, Wolfram Schulz, Tim Friedman, and Eveline Gebhardt. *Preparing for Life in a Digital Age*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-14221-0. doi: 10.1007/978-3-319-14222-7. URL http://link.springer.com/10.1007/978-3-319-14222-7.

[27] Andrew V Frane. Are Per-Family Type I Error Rates Relevant in Social and Behavioral Science? *Journal of Modern Applied Statistical Methods*, 14(1):12–23, 2015. ISSN 1538-9472. doi: 10.22237/jmasm/1430453040. URL http://digitalcommons.wayne.edu/jmasm/vol14/iss1/5.

[28] F García-Peñalvo, D Reimann, M Tuul, A Rees, and I Jormanainen. TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers KA2 project " TACCLE 3 – Coding " (2015-1-BE02-KA201-012307). page 72, 2016. doi: 10.5281/zenodo.165123. URL http://repositorio.grial.eu/bitstream/grial/688/1/TACCLE3O5Literaturereview-final.pdf.

[29] Paul Gilster. *Digital Literacy*, volume 273. 1997. ISBN 0471249521. URL http://www.nwlg.org/digitalliteracy/.

[30] R. D. Goddard, W. K. Hoy, and A. W. Hoy. Collective Teacher Efficacy: Its Meaning, Measure, and Impact on Student Achievement. *American Educational Research Journal*, 37(2):479–507, 2000. ISSN 0002-8312. doi: 10.3102/00028312037002479. URL http://aer.sagepub.com/cgi/doi/10.3102/00028312037002479.

[31] Lindsey Ann Gouws, Karen Bradshaw, and Peter Wentworth. Computational thinking in educational activities. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13*, page 10, 2013. doi: 10.1145/2462476.2466518. URL http://dl.acm.org/citation.cfm?id=2462476.2466518{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2462476.2466518.

[32] Sarah Gretter and Aman Yadav. Computational Thinking and Media & Information Literacy : An Integrated Approach to Teaching Twenty-First Century Skills. *TechTrends*, pages 510–516, 2016. ISSN 8756-3894. doi: 10.1007/s11528-016-0098-4. URL http://dx.doi.org/10.1007/s11528-016-0098-4.

[33] S. Grover and R. Pea. Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, 42(1):38–43, jan 2013. ISSN 0013-189X. doi: 10.3102/0013189X12463051. URL http://edr.sagepub.com/cgi/doi/10.3102/0013189X12463051.

[34] Shuchi Grover, Roy Pea, and Stephen Cooper. Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2):199–237, apr 2015. ISSN 0899-3408. doi: 10.1080/08993408.2015.1033142. URL http://dx.doi.org/10.1080/08993408.2015.1033142http://www.tandfonline.com/doi/full/10.1080/08993408.2015.1033142.

[35] Judith T.M. Gulikers, Harm J.A. Biemans, Renate Wesselink, and Marjan van der Wel. Aligning formative and summative assessments: A collaborative action research challenging teacher conceptions. *Studies in Educational Evaluation*, 39(2):116–124, 2013. ISSN 0191491X. doi: 10.1016/j.stueduc.2013.03.001.

[36] Ying Guo, Shayne B. Piasta, Laura M. Justice, and Joan N. Kaderavek. Relations among preschool teachers' self-efficacy, classroom quality, and children's language and literacy gains. *Teaching and Teacher Education*, 26(4):1094–1103, 2010. ISSN 0742051X. doi: 10.1016/j.tate.2009.11.005.

[37] Felienne Hermans and Efthimia Aivaloglou. To Scratch or not to Scratch ?: A controlled experiment comparing plugged first and unplugged first programming lessons. 2017.

[38] N. Hockly. Digital literacies. *ELT Journal*, 66(1):108–112, jan 2012. ISSN 0951-0893. doi: 10.1093/elt/ccr077. URL `http://content.ebscohost.com/ContentServer.asp?T=P{&}P=AN{&}K=39774959{&}S=R{&}D=lih{&}EbscoContent=dGJyMNLr40SeqLI4v+bwOLCmr0qep65Ssau4SrOWxWXS{&}ContentCustomer=dGJyMPGutk6uqbFQuePfgeyx44Dt6fIA{%}5Cnhttp://search.ebscohost.com/login.aspx?direct=true{&}db=lih{&}AN=3977495`.

[39] Simon Peyton Jones. *COMPUTING AT SCHOOL Computing A CPD toolkit for primary teachers*. 2014. ISBN 9781783395217. URL `http://primary.quickstartcomputing.org/resources/pdf/qs{_}handbook.pdf`.

[40] Yasmin B. Kafai. From computational thinking to computational participation in K–12 education. *Communications of the ACM*, 59(8):26–27, jul 2016. ISSN 00010782. doi: 10.1145/2955114. URL `http://dl.acm.org/citation.cfm?doid=2975594.2955114`.

[41] Yasmin B. Kafai and Quinn Burke. *Connected Code : Why Children Need to Learn Programminged Teaching*. 2014. ISBN 9780262027755.

[42] P Kemp. *Computing in the national curriculum A guide for secondary teachers Computing in the*. 2014. ISBN 9781783393763. URL `http://www.computingatschool.org.uk/data/uploads/cas{_}secondary.pdf`.

[43] Kennisnet. Computational thinking in het Nederlandse onderwijs. page 48, 2016. URL `https://www.kennisnet.nl/fileadmin/kennisnet/publicatie/Computational{_}Thinking{_}in{_}het{_}Nederlandse{_}onderwijs.pdf`.

[44] Paul Kline. Other types of Psychological Tests. In *The Handbook of Psychological Testing*, pages 325–346. 2000. ISBN 0415211581.

[45] KNAW. *Digitale geletterdheid in het voortgezet onderwijs*. 2012. ISBN 9789069846606. URL `http://www.knaw.nl/Content/Internet{_}KNAW/publicaties/pdf/20121027.pdf`.

[46] Alice Man Sze Lau. 'Formative good, summative bad?' – A review of the dichotomy in assessment literature. *Journal of Further and Higher Education*, 40(4):509–525, 2016. ISSN 14699486. doi: 10.1080/0309877X.2014.984600.

[47] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. Computational thinking for youth in practice. *ACM Inroads*, 2(1):32, 2011. ISSN 21532184. doi: 10.1145/1929887.1929902. URL `http://dl.acm.org/citation.cfm?doid=1929887.1929902`.

[48] Raymond Lister. Toward a Developmental Epistemology of Computer Programming. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '16*, pages 5–16, New York, New York, USA, 2016. ACM Press. ISBN 9781450342230. doi: 10.1145/2978249.2978251. URL `http://dl.acm.org/citation.cfm?doid=2978249.2978251`.

[49] Matthew Lombard, Jennifer Snyder-Duch, and Cheryl Campanella Bracken. Content Analysis in Mass Communication: Assessment and Reporting of Intercoder Reliability, 2002. ISSN 03603989.

[50] Sze Yee Lye and Joyce Hwee Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41:51–61, dec 2014. ISSN 07475632. doi: 10.1016/j.chb.2014.09.012. URL `http://dx.doi.org/10.1016/j.chb.2014.09.012http://linkinghub.elsevier.com/retrieve/pii/S0747563214004634`.

[51] Marion Milton, Mary Rohl, and Helen House. Secondary Beginning Teachers ' Preparedness to Teach Literacy and Numeracy: A Survey. *Australian Journal of Teacher Education*, 32(2):37–56, 2007. URL htt://ro.ecu.edu.au/ajte/vol32/iss2/4.

[52] Jesús Moreno-león, Gregorio Robles, Universidad Rey, and Juan Carlos. Analyze your Scratch projects with Dr . Scratch and assess your Computational Thinking skills. *Scratch Conference 2015*, pages 1–7, 2015.

[53] Chrystalla Mouza, Lori Pollock, Kathleen Pusecker, Kevin Guidry, Ching-Yi Yeh, James Atlas, and Terry Harvey. Implementation and Outcomes of a Three-Pronged Approach to Professional Development for CS Principles. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*, pages 66–71, New York, New York, USA, 2016. ACM Press. ISBN 9781450336857. doi: 10.1145/2839509.2844585. URL http://dl.acm.org/citation.cfm?doid=2839509.2844585.

[54] National Research Council. *Report of a Workshop on The Scope and Nature of Computational Thinking*. 2010. ISBN 978-0-309-14957-0. doi: 10.17226/12840. URL http://www.nap.edu/catalog/12840.

[55] National Research Council. *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. 2011. ISBN 978-0-309-21474-2. doi: 10.17226/13170. URL http://www.nap.edu/catalog/13170.

[56] Greg L Nelson, Benjamin Xie, and Andrew J Ko. Comprehension First : Evaluating a Novel Pedagogy and Tutoring System for Program Tracing in CS1. pages 42–51, 2017.

[57] National academy of engineering of medicine National academy of sciences and Institute. *On Being a Scientist*. 2009. ISBN 9780309119702. doi: 10.17226/12192.

[58] Platform Onderwijs2032. *Advisory Report Ons onderwijs2032 Advisory Report*. Number January. 2016. ISBN 9789082492804.

[59] J Pallant. SPSS survival manual. *Journal of Advanced Nursing*, 36(3):478–478, 2007. ISSN 03092402. doi: 10.1046/j.1365-2648.2001.2027c.x. URL http://mcgraw-hill.co.uk/openup/chapters/0335208908.pdf.

[60] Seymour. Papert. Mindstorms: Computers, children, and powerful ideas. *NY: Basic Books*, page 1980, 1980. URL http://dl.acm.org/citation.cfm?id=1095592.

[61] Lori Pollock, Crystalla Mouza, Amanda Czik, Alexis Little, Debra Coffey, and Joan Buttram. From Professional Development to the Classroom: Findings from CS K-12 Teachers. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 477–482, 2017. doi: 10.1145/3017680.3017739. URL http://doi.acm.org/10.1145/3017680.3017739.

[62] Laura Helena Porras-Hernández and Bertha Salinas-Amescua. Strengthening Tpack: A Broader Notion of Context and the Use of Teacher's Narratives to Reveal Knowledge Construction. *Journal of Educational Computing Research*, 48(2):223–244, 2013. ISSN 0735-6331. doi: 10.2190/EC.48.2.f. URL http://journals.sagepub.com/doi/10.2190/EC.48.2.f.

[63] Alexander Repenning, David Webb, and Andri Ioannidou. Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, page 265, New York, New York, USA, 2010. ACM Press. ISBN 9781450300063. doi: 10.1145/1734263.1734357. URL http://portal.acm.org/citation.cfm?doid=1734263.1734357.

[64] Alexander Repenning, Ryan Grover, Kris Gutierrez, Nadia Repenning, David C Webb, Kyu Han Koh, Hilarie Nickerson, Susan B Miller, Catharine Brand, Ian Her Many Horses, Ashok Basawapatna, and Fred Gluck. Scalable Game Design. *ACM Transactions on*

Computing Education, 15(2):1–31, apr 2015. ISSN 19466226. doi: 10.1145/2700517. URL http://dl.acm.org/citation.cfm?doid=2767124.2700517.

[65] Phil Scott, Eduardo Mortimer, and Jaume Ametller. Pedagogical link-making: A fundamental aspect of teaching and learning scientific conceptual knowledge. *Studies in Science Education*, 47(1):3–36, 2011. ISSN 03057267. doi: 10.1080/03057267.2011.549619.

[66] Cynthia Selby. Computational Thinking : The Developing Definition. *ITiCSE Conference 2013*, pages 5–8, 2013.

[67] Cynthia C. Selby. Relationships: computational thinking, pedagogy ofprogramming, and Bloom's Taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*, pages 80–87, New York, New York, USA, 2015. ACM Press. ISBN 9781450337533. doi: 10.1145/2818314.2818315. URL http://doi.acm.org/10.1145/2818314.2818315{%}5Cnhttp://dl.acm.org/citation.cfm?id=2818315http://dl.acm.org/citation.cfm?doid=2818314.2818315.

[68] Oshani Seneviratne. *Emerging Research, Practice, and Policy on Computational Thinking*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-52690-4. doi: 10.1007/978-3-319-52691-1. URL http://link.springer.com/10.1007/978-3-319-52691-1.

[69] Sue Sentance and Andrew Csizmadia. Teachers ' perspectives on successful strategies for teaching Computing in school. *Ifip Tc3*, (June):1–11, 2015. URL http://community.computingatschool.org.uk/files/6303/original.pdf.

[70] Valerie J. Shute and Yoon Jeon Kim. Formative and stealth assessment. In *Handbook of Research on Educational Communications and Technology: Fourth Edition*, pages 311–321. 2014. ISBN 9781461431855. doi: 10.1007/978-1-4614-3185-5_3.

[71] Melinda R. Snodgrass, Maya Israel, and George C. Reese. Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers and Education*, 100:1–17, 2016. ISSN 03601315. doi: 10.1016/j.compedu.2016.04.011. URL http://dx.doi.org/10.1016/j.compedu.2016.04.011.

[72] Stephen Wolfram. How to teach Computational Thinking, 2016. URL http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/.

[73] Jean Stockard, Timothy W. Wood, Cristy Coughlin, and Caitlin Rasplica Khoury. The Effectiveness of Direct Instruction Curricula: A Meta-Analysis of a Half Century of Research. *Review of Educational Research*, XX(X):003465431775191, 2018. ISSN 0034-6543. doi: 10.3102/0034654317751919. URL http://journals.sagepub.com/doi/10.3102/0034654317751919.

[74] The Royal Society. Shut down or restart? The way forward for computing in UK schools. *Technology*, (January):1–122, 2012. doi: 10.1088/2058-7058/25/07/21.

[75] Joke Voogt, Petra Fisser, Jon Good, Punya Mishra, and Aman Yadav. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4):715–728, 2015. ISSN 15737608. doi: 10.1007/s10639-015-9412-6.

[76] Riina Vuorikari, Yves Punie, Stephanie Carretero, and Lieve Van Den Brande. *DigComp 2.0: The Digital Competence Framework for Citizens*. 2016. ISBN 978-92-79-58876-1. doi: 10.2791/11517. URL http://publications.jrc.ec.europa.eu/repository/bitstream/JRC101254/jrc101254{_}digcomp2.0thedigitalcompetenceframeworkforcitizens.updatephase1.pdf.

[77] Mary Webb, Niki Davis, Tim Bell, Y. Katz, Nicholas Reynolds, Dianne P. Chambers, and Maciej M. Sysło. Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, 22(2):445–468, 2017. ISSN 15737608. doi: 10.1007/s10639-016-9493-x.

[78] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, 25(1):127–147, feb 2016. ISSN 1059-0145. doi: 10.1007/s10956-015-9581-5. URL `http://link.springer.com/10.1007/s10956-015-9581-5`.

[79] J. M Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366 (1881):3717–3725, oct 2008. ISSN 1364-503X. doi: 10.1098/rsta.2008.0118. URL `http://rsta.royalsocietypublishing.org/cgi/doi/10.1098/rsta.2008.0118`.

[80] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33, mar 2006. ISSN 00010782. doi: 10.1145/1118178.1118215. URL `http://portal.acm.org/citation.cfm?doid=1118178.1118215`.

[81] Jeannette M Wing. Computational Thinking: What and Why? *the-link - The Magaizne of the Varnegie Mellon University School of Computer Science*, (March 2006):1–6, 2010. URL `http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why`.

[82] Ursula Wolz, Meredith Stone, Kim Pearson, Sarah Monisha Pulimood, and Mary Switzer. Computational Thinking and Expository Writing in the Middle School. *ACM Transactions on Computing Education*, 11(2):1–22, jul 2011. ISSN 19466226. doi: 10.1145/1993069.1993073. URL `http://dl.acm.org/citation.cfm?doid=1993069.1993073`.

[83] Aman Yadav, Ninger Zhou, Chris Mayfield, Susanne Hambrusch, and John T. Korb. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11*, number 2, page 465, New York, New York, USA, 2011. ACM Press. ISBN 9781450305006. doi: 10.1145/1953163.1953297. URL `http://portal.acm.org/citation.cfm?doid=1953163.1953297`.

[84] Aman Yadav, Jon Good, Joke Voogt, and Petra Fisser. Competence-based Vocational and Professional Education. 23:1051–1067, 2017. ISSN 1871-3041. doi: 10.1007/978-3-319-41713-4. URL `http://link.springer.com/10.1007/978-3-319-41713-4`.

Appendices

A

Semistructured Interview Format

Acties van mijzelf:

- ❏ Uitleg opname, verwerking van data anoniem (informed consent / ethics) en structuur interview
- ❏ Redundant opnemen (telefoon+macbook)

Inleidende vragen:

- ❏ naam / school / leeftijd / hoe lang voor de klas / achtergrond/vak
- ❏ hoe lang ervaring CT-gerelateerde lessen

U heeft nu een (basis) curus digitale geletterdheid achter de rug waarbij de focus lag op CT, waar in de praktijk vorm aan gegeven kan worden via programmeren. Met deze vorm van professionalisering achter de rug ..

- ❏ **Open: Wat zijn uw behoeftes op dit moment om CT te onderwijzen?**
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Middelen
  - ❏ Op het gebied van professionalisering

Aan het einde komen we nog een keer bij deze vraag terug.

- ❏ **Open: Hoe zou u CT omschrijven aan iemand?**
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Wat komt er bij u op als u denkt aan ‘Computational Thinking’?

- ❏ **Open: Welke pedagogische strategie / didactiek zet u in bij het onderwijzen van CT ?**
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Welke activiteiten wilt u dat leerlingen doorlopen en in welke volgorde?
    - ❏ Waarom bied je deze activiteiten in deze volgorde aan?
    - ❏ Waarom heb je voor dit lesmateriaal gekozen?
    - ❏ Wat heeft je de overtuiging gegeven om deze keuzes te maken?
  - ❏ Wat wilt u de leerlingen aanleren om uw doelen te behalen?
    - ❏ Wat heeft je de overtuiging gegeven om deze keuzes te maken?
  - ❏ Hoe geeft u hen les om deze doelen te behalen?
  - ❏ Wat is uw rol als docent hierbij?
  - ❏ Hoe controleert u of leerlingen uw doelen behalen?
  - ❏ Wat doet u helemaal buiten de computer en/of andere ICT om mbt het lesgeven in CT?

  Indien nodig draag ik voorbeelden aan zoals omschreven bij het stuk over pedagogiek ter verduidelijking om te vragen in hoeverre de deelnemer dit toepast/kent.

- ❏ **Open: Welke leerproblemen zijn voor u bekend bij het onderwijzen van CT?**
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Bij complexe situaties waarbij, buiten de inzet van een programmeertaal, zaken vereenvoudigt moeten worden om tot de kern van een probleem te komen.
    Voor verduidelijking toelichting abstractie
  - ❏ Bij het formuleren van een oplossing in generieke termen zodanig dat het ook toegepast kan worden bij een ander probleem
    Voor verduidelijking toelichting generalisatie
  - ❏ Bij het opdelen van een probleem of taak in kleinere deelproblemen/deeltaken om de complexiteit te kunnen hanteren
    Voor verduidelijking toelichting decompositie
  - ❏ Bij het verklaren van algoritmes aan de hand van een serie geordende instructies/regels die stap voor stap uitgevoerd worden om een probleem op te lossen of een doel te bereiken.
    Voor verduidelijking algoritmisch denken
  - ❏ Bij het opsporen en verhelpen van fouten door logisch te redeneren. (debugging)

- ❏ **Open: Welke kennis/vaardigheden past u toe aangaande integratie van ICT in uw lessen CT?**
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Hoe zet u ICT in om leerlingen CT aan te leren?
  - ❏ Welke vormen van digitale technologie ziet u als geschikt om CT aan te leren?
  - ❏ Wat beschouwt u als ICT-basisvaardigheden voor een docent om CT aan te kunnen leren?
  - ❏ Hoe maakt u zich nieuwe vormen van technologie eigen? (software & hardware)
  - ❏ In hoeverre gaat u mee met nieuwe technologische ontwikkelingen?

  Indien nodig draag ik voorbeelden aan zoals omschreven bij het stuk over technologie ter verduidelijking om te vragen in hoeverre de deelnemer dit toepast/kent.

- ❏ **Open:** Wat kunt u vertellen over de condities op sociaal-, technologisch-, organisatorisch en economisch vlak die ertoe hebben geleid dat CT in uw school een (grotere) rol is gaan spelen?
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Van wie is het initiatief gekomen (leerlingen/directie/collega's e.d.)
  - ❏ Wat voor noodzaak is er volgens u aanwezig (geweest)?

- ❏ **Open:** Wat kunt u vertellen over de voorwaarden die nodig zijn om binnen de klas aan CT te werken?
  Vervolg/verdiepende vragen indien nodig:
  - ❏ Welke behoeften zijn er op het gebied van middelen/tijd/geld?
  - ❏ Wat is uw mening/opvatting en die van uw leerlingen aangaande het nut van CT?

Na deze verkenning van CT in en rondom uw lessen keren we nu weer terug naar de eerste vraag aangaande de behoeftes.

- ❏ **Wilt u hier nog iets aan toevoegen?**

- ❏ Heeft u verder nog op- of aanmerkingen aangaande het interview?

Abstractie

Omschrijving: Complexe situaties eenvoudiger weergeven door het belangrijkste te benadrukken en details buiten beschouwing te laten om zo te komen tot de kern van een probleem.

Zoals:

- Herkennen van belangrijke elementen in een proces, verhaal of foto.
- Identificeren van verschillen in vergelijkbare situaties en deze in versimpelde termen benoemen.
- Beschrijven hoe apparaten en digitale instrumenten werken door de hoofdlijnen en belangrijkste componenten aan te geven.
- Weergeven van de werkelijkheid in een conceptueel model.
- Herkennen en gebruiken van verschillende geabstraheerde verschijningsvormen (zoals een plattegrond/kaart of begrip/concept).

Voorbeeld activiteiten:
http://www.drscratch.org/learn/Abstraction/
https://scratch.mit.edu/projects/120178758/#editor
Laat een leerling bijvoorbeeld een conceptueel model bedenken van een tafel; ‘wanneer is iets een tafel’ ?

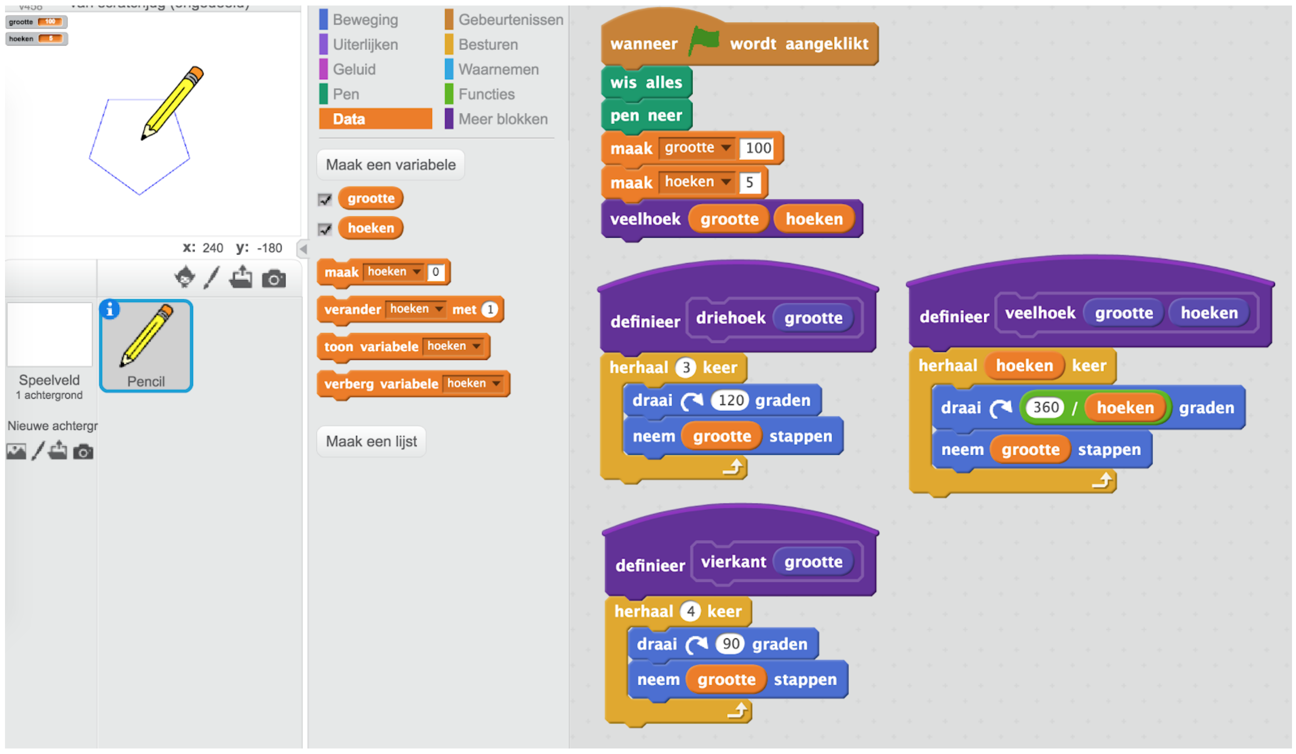
Generalisatie (patronen)

Omschrijving: Formuleren van een oplossing in generieke termen zodanig dat het ook toegepast kan worden bij een ander probleem.

Zoals:

- Herkennen van patronen, overeenkomsten en verbanden om die vervolgens toe te passen.
- Een manier om nieuwe problemen (snel) op te lossen op basis van oplossingen en voorgaande ervaring van eerdere problemen.
- Je afvragen in hoeverre iets lijkt op problemen die je al eerder hebt opgelost en op welke manier ze anders zijn.
- Een algoritme dat de oplossing is voor een bepaald probleem kan op een zodanige manier worden aangepast dat het toepasbaar is op een groot aantal overeenkomstige problemen. Wanneer in zo'n geval een zelfde type probleem moet worden opgelost kan een universele oplossing worden toegepast.

Voorbeeld activiteiten:
Een leerling maakt in Scratch een tekening om een driehoek en een vierkant te tekenen. De leerling maakt er uiteindelijk door middel van abstractie een 'eigen blok' van met de naam driehoek en een blok met de naam vierkant. Vervolgens wil de leerling een vijfhoek tekenen. Gebaseerd op de code van het vierkant en de driehoek komt de leerling tot de realisatie dat er een relatie is tussen het aantal zijden en de bijbehorende hoeken. Vervolgens schrijft de leerling het algoritme uit die deze relatie tot uitdrukking laat komen om vervolgens elke veelhoek te kunnen tekenen.



```
wanneer [vlag] wordt aangeklikt
wis alles
pen neer
maak grootte 100
maak hoeken 5
veelhoek (grootte) (hoeken)

definieer driehoek (grootte)
herhaal 3 keer
    draai ↻ 120 graden
    neem (grootte) stappen

definieer veelhoek (grootte) (hoeken)
herhaal (hoeken) keer
    draai ↻ (360 / (hoeken)) graden
    neem (grootte) stappen

definieer vierkant (grootte)
herhaal 4 keer
    draai ↻ 90 graden
    neem (grootte) stappen
```

Decompositie

Omschrijving: Probleem of taak opdelen in kleinere deelproblemen of deeltaken om de complexiteit te kunnen hanteren.

Zoals:

- Opdelen van een grotere en meer complexe taak in een aantal deeltaken (bijv. bij een zaakvak, werkstuk, regie van film of lesstof)
- Omzetten van een concreet probleem in een passende visuele weergave (bijv. stappenschema)
- Uitwerken van deelopdrachten en de opbrengsten samenstellen tot een eindproduct
- Checken of geen belangrijk deel gemist of vergeten wordt bij het uitvoeren van deelopdrachten

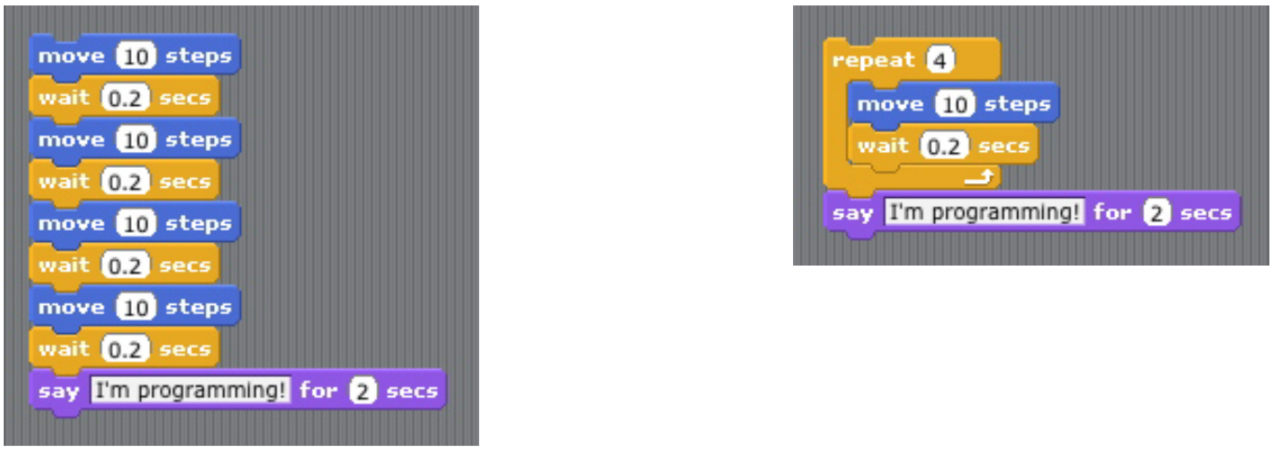
Voorbeeld activiteiten:
https://scratch.mit.edu/projects/2647059/

Algoritmisch denken

Omschrijving: De werking van algoritmes verklaren als een serie geordende instructies of regels die stap voor stap worden uitgevoerd om een probleem op te lossen of een doel te bereiken. Veelgebruikte concepten daarbij zijn sequences, loops, parallelism, events, conditionals, operators, en dataflow.

Verdere verduidelijking

Sequences (sequentie) & Loops / Voorbeeld:

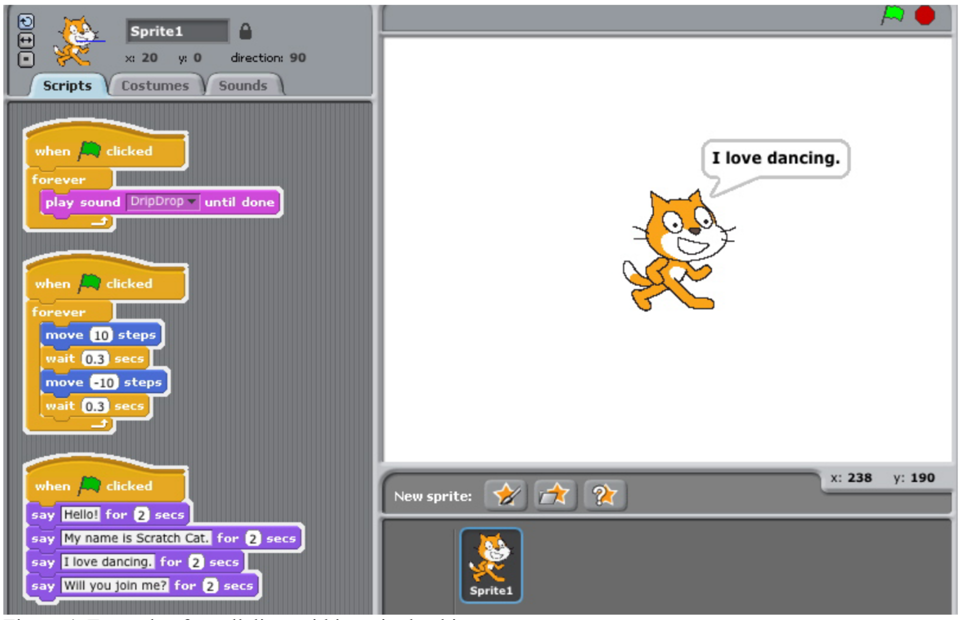


```
move 10 steps
wait 0.2 secs
move 10 steps
wait 0.2 secs
move 10 steps
wait 0.2 secs
move 10 steps
wait 0.2 secs
say I'm programming! for 2 secs
```

```
repeat 4
    move 10 steps
    wait 0.2 secs
say I'm programming! for 2 secs
```

Figure 2. A sequence of repeated instructions expressed as a loop.

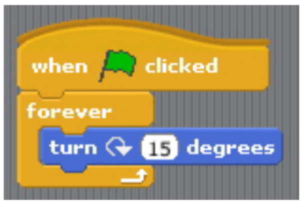
Parallelism (parallellisatie) / Voorbeeld:

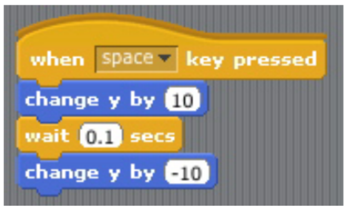


```
when clicked
forever
    play sound DripDrop until done

when clicked
forever
    move 10 steps
    wait 0.3 secs
    move -10 steps
    wait 0.3 secs

when clicked
say Hello! for 2 secs
say My name is Scratch Cat. for 2 secs
say I love dancing. for 2 secs
say Will you join me? for 2 secs
```

Figure 4. Example of parallelism within a single object.

Events (gebeurtenissen) / Voorbeeld:





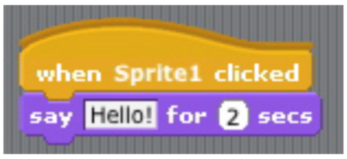


Figure 3. Examples of events producing actions.

Conditionals (condities) / Voorbeeld:

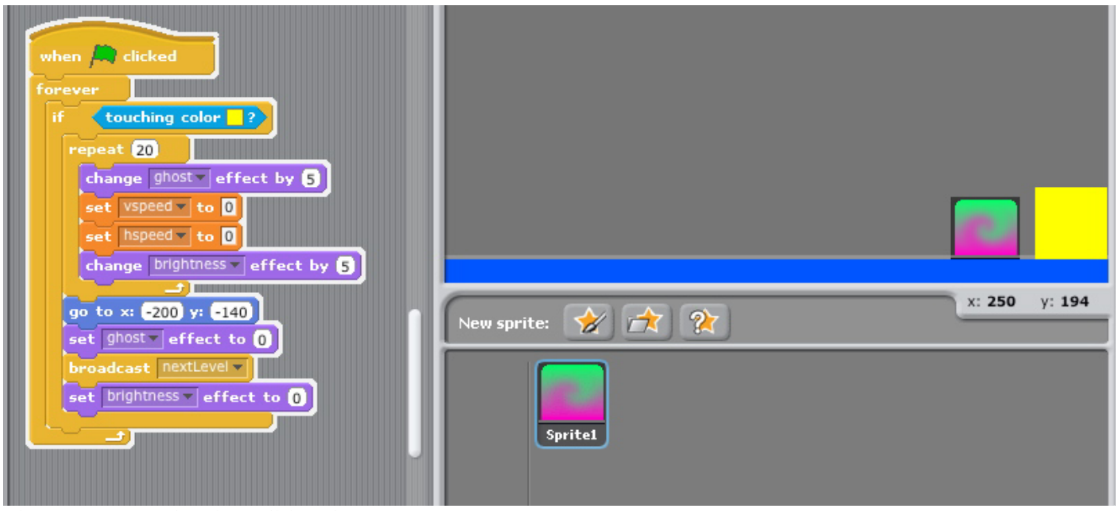


Figure 5. Example of conditionals.

Operators (operatoren) / Voorbeeld:

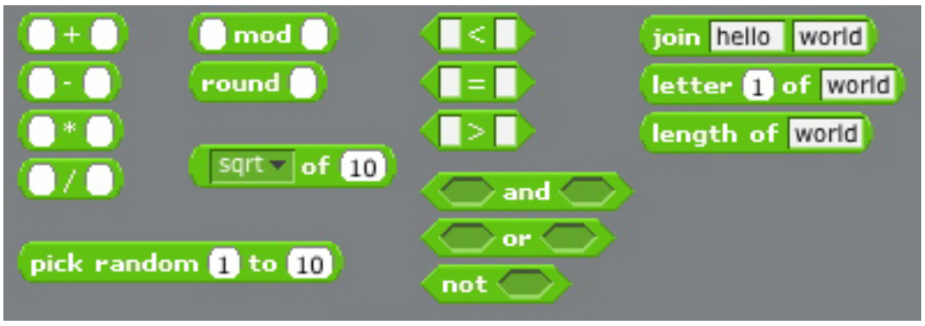
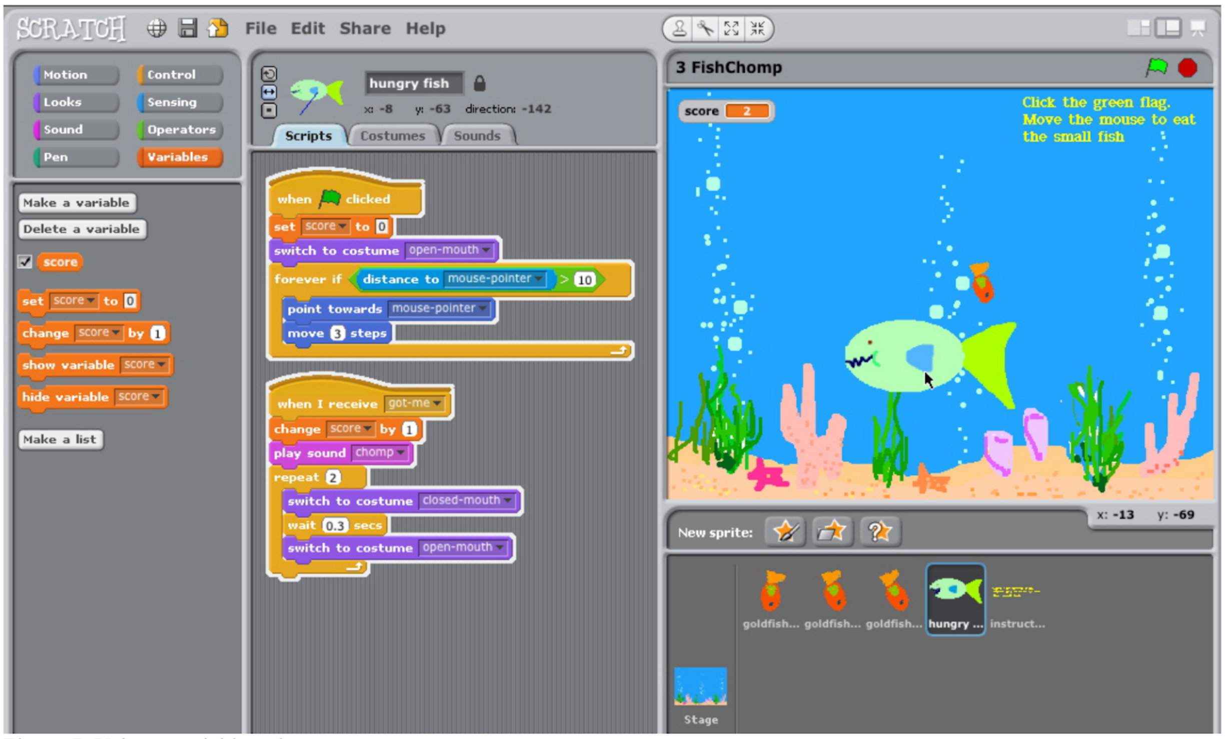


Figure 6. Operator blocks.

Data (gegevens) / Voorbeeld:



```
when [flag] clicked
set score to 0
switch to costume open-mouth
forever if distance to mouse-pointer > 10
    point towards mouse-pointer
    move 3 steps

when I receive got-me
change score by 1
play sound chomp
repeat 2
    switch to costume closed-mouth
    wait 0.3 secs
    switch to costume open-mouth
```

Figure 7. Using a variable to keep score.

Dataflow / Omschrijving:

Het kunnen verklaren en ordenen van gebeurtenissen in een logische volgorde op basis van het aangeleverde probleem.

Pedagogiek

- Het gebruik van vragen voor meer begrip (onderwijs leergesprek)
- Het gebruik van voorbeelden (uitleg en demonstraties)
- Werken aan het (iteratief & incrementeel) opbouwen van modellen van een probleem
- Het presenteren of uitleggen van een probleem aan de hand van een serie stappen
- Het modelleren van beslissingen nemen aan de hand van bepaalde condities
- Het uitbouwen en herbruiken van bestaande tools/oplossingen van jezelf en anderen (reuse/remix)
- Het opdelen van een complex probleem in kleinere (simpelere) stappen waarbij incrementeel naar de oplossing toegewerkt wordt.
- Het uitwerken van een model (voor een bepaald probleem) voordat er gebruik word gemaakt van de computer om het op te lossen
- Het laten uitproberen, constant evalueren en aanpassen

Technologie

- Het vinden en gebruiken van online animaties/filmpjes om zaken te demonstreren/verduidelijken
- Het internet gebruiken om misconcpten te ontdekken gerelateerd aan CT
- Het inzetten van digitale technologie geschikt voor het aanleren van CT
- Het helpen van leerlingen bij de inzet van digitale technologie om gegevens te verzamelen
- Het helpen van leerlingen bij de inzet van digitale technologie om gegevens te ordenen en er patronen in te ontdekken
- Het helpen van leerlingen bij de inzet van digitale technologie om modellen/simulaties te maken en/of aan te passen.

- Het inzetten van digitale technologie om mijn productiviteit als docent te vergroten
- Het inzetten van digitale technologie om mijn communicatie met leerlingen te verbeteren
- Het inzetten van digitale technologie om leerlingen te motiveren
- Het inzetten van digitale technologie om de presentatie van gegevens te verbeteren (spreadsheets e.d.)
- Het inzetten van digitale technologie om te helpen bij het toetsen van de leerdoelen

- Het inzetten van digitale technologie die wetenschappers gebruiken
    - om gegevens inzichtelijk te maken die anderzijds moeilijk waar te nemen zijn
    - om de weergave van gebeurtenissen te verlangzamen/ versnellen
    - om modellen/simulaties te maken van wetenschappelijke fenomenen
    - om gegevens te verzamelen die anderzijds moeilijk te verzamelen is
    - om gegevens zodanig te organiseren/ordenen zodat patronen ondekt kunnen worden die normaliter moeilijk te achterhalen zijn.

- ICT basisvaardigheden zoals
    - het opslaan van afbeeldingen
    - opzoeken van specifieke informatie aangaande een bepaald onderwerp
    - een email opstellen met bijlagen
    - een presentatie- en tekstdocument maken met bijbehorende software
    - een nieuw programma jezelf eigen maken
    - een programma zelf installeren
    - het aanpassen/maken van foto/video
    - het gebruiken van ‘nieuwere media’ zoals blogs, social media, podcasts, eigen website

B
Intercoder reliability SPSS output

```
Run MATRIX procedure:

Krippendorff’s Alpha Reliability Estimate


             Alpha    LL95%CI    UL95%CI      Units   Observrs      Pairs
Nominal      .6050      .3582      .8025    24.0000     2.0000    24.0000

Probability (q) of failure to achieve an alpha of at least alphamin:
   alphamin          q
      .9000      .9964
      .8000      .9434
      .7000      .7382
      .6700      .7382
      .6000      .4106
      .5000      .1414

Number of bootstrap samples:
  5000

Judges used in these computations:
 julian   felienne

====================================================

Observed Coincidence Matrix
     .00     .00    1.00     .00     .00     .00    2.00    1.00     .00    2.00
     .00    4.00     .00    1.00     .00     .00     .00     .00     .00     .00
    1.00     .00     .00     .00     .00     .00     .00     .00     .00     .00
     .00    1.00     .00    4.00     .00     .00     .00     .00     .00     .00
     .00     .00     .00     .00   16.00     .00     .00     .00     .00     .00
     .00     .00     .00     .00     .00    2.00     .00     .00     .00     .00
    2.00     .00     .00     .00     .00     .00     .00     .00     .00     .00
    1.00     .00     .00     .00     .00     .00     .00     .00    1.00     .00
     .00     .00     .00     .00     .00     .00     .00    1.00    2.00     .00
    2.00     .00     .00     .00     .00     .00     .00     .00     .00    4.00
```

```
Expected Coincidence Matrix
     .64     .64     .13     .64    2.04     .26     .26     .26     .38     .77
     .64     .43     .11     .53    1.70     .21     .21     .21     .32     .64
     .13     .11     .00     .11     .34     .04     .04     .04     .06     .13
     .64     .53     .11     .43    1.70     .21     .21     .21     .32     .64
    2.04    1.70     .34    1.70    5.11     .68     .68     .68    1.02    2.04
     .26     .21     .04     .21     .68     .04     .09     .09     .13     .26
     .26     .21     .04     .21     .68     .09     .04     .09     .13     .26
     .26     .21     .04     .21     .68     .09     .09     .04     .13     .26
     .38     .32     .06     .32    1.02     .13     .13     .13     .13     .38
     .77     .64     .13     .64    2.04     .26     .26     .26     .38     .64

Delta Matrix
     .00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00
    1.00     .00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00
    1.00    1.00     .00    1.00    1.00    1.00    1.00    1.00    1.00    1.00
    1.00    1.00    1.00     .00    1.00    1.00    1.00    1.00    1.00    1.00
    1.00    1.00    1.00    1.00     .00    1.00    1.00    1.00    1.00    1.00
    1.00    1.00    1.00    1.00    1.00     .00    1.00    1.00    1.00    1.00
    1.00    1.00    1.00    1.00    1.00    1.00     .00    1.00    1.00    1.00
    1.00    1.00    1.00    1.00    1.00    1.00    1.00     .00    1.00    1.00
    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00     .00    1.00
    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00    1.00     .00

Rows and columns correspond to following unit values
    1.00    6.00    7.00   18.00   19.00   20.00   21.00   24.00   25.00   28.00

Examine output for SPSS errors and do not interpret if any are found

------ END MATRIX -----
```

C Coding scheme updates

Between the first and the second interview the following changes/updates were made:

- For the content knowledge (CK) and learner knowledge (LK) the sublabels for abstraction, decomposition and generalization were merged into a new label called 'algorithmic thinking' after establishing granularity is too high.
- An implication was established for learner knowledge (LK), i.e. a teacher that is capable of explicating LK for a certain LK subcategory also means this teacher has content knowledge (CK) for this particular subcategory. Please note that the reverse is not necessary true!

Between the second and the third interview the following changes/updates were made:

- A new subcategory for pedagogical knowledge (PK) concerning the general transformation of content knowledge (CK) to pedagogical knowledge (PK) was added.
- Learning programming languages (tools) now belongs under the category technological knowledge (TK) instead of CK
- When trying to differentiate between PK and CK related the following agreements were made: if a participant describes elements of CT it belongs in the CK category, but if it is being described as how to teach it than it belongs in the PK category.

Between the third and the fourth interview the following changes/updates were made:

- To reduce ambiguity within the 'CT related PK labels' a clear line between CT-related PK and non-CT related PK was made. As such 4 originally split up labels are now merged into 1 label.
- To improve granularity for PK connected to CT, considering the transformation of CK about algorithmic thinking to PK, we now make a difference between a transformation that is executed through a programming language or other technological tool versus without the use of any programming language or technological tool. As such the PK sublabel teaching algorithmic thinking is now split up in a TECH and non-TECH variant.
- PK subcategory A, B, E and F are now placed under algorithmic thinking (their umbrella term)
- PK subcategory C is merged with subcategory F with an updated description
- PK subcategory E is renamed to 'PK_CT_REUSEMIX'.
- PK subcategory G is now placed under 'PK_CT_DEBUG' with an updated description, i.e. more examples.
- The subcategory of learner knowledge for algorithmic thinking was updated to be more universal applicable.
- An implication was established for pedagogical knowledge (PK), i.e. a teacher that is capable of explicating PK for a certain CT subcategory it also means this teacher has the content knowledge (CK) for this particular subcategory. Please note that the reverse is not necessary true!

D
Final coding scheme

This coding scheme includes all categories, subcategories and labels associated with the original TPACK model. Please note the table captions.

Table D.1: Content Knowledge (CK) for computational thinking and its subcategories includes knowledge and understanding about..

#	LABEL/CODE	DESCRIPTION
1.	CK_CT_ALGO	the skills/knowledge of abstraction (including modeling), generalization, decomposition and several computational thinking concepts, such as, data, processing, information, sequencing, loops, parallel processing, events, conditions, operators, variables, and dataflow of control.
2.	CK_CT_DEBUG	the skills of and attitude necessary for debugging (i.e. understanding that there is not one best solution when it comes to creating working code, understanding that it is about trial and error)

Table D.2: Learner Knowledge (LK) for computational thinking and its subcategories includes knowledge about learners' characteristics, preconceptions and difficulties in..

#	LABEL/CODE	DESCRIPTION
3.	LK_CT_GEN	aspects connected to programming languages that impede CT learning i.e. the linguistic aspect.
4.	LK_CT_ALGO	developing abstractions (that are beyond of any particular programming language or tool), generalizing from one solution to another by identifying common patterns, decomposing complex problems to simpler ones, thinking algorithmically to solve a problem (including difficulties in understanding relevant concepts, such as sequencing, loops, flow of control, conditions, etc.)
5.	LK_CT_DEBUG	skills and attitude necessary for debugging

Table D.3: General pedagogical knowledge (PK) and PK for computational thinking (PK_CT) and its subcategories

#	LABEL/CODE	DESCRIPTION
6.	PK_NON_CT	didactics not necessarily connected to CT such as effective management of student learning i.e. differentiated instruction,a student-centered / teacher-centered approach, project-based learning, group work, collaboration,the use of questions to promote understanding,use of examples, use of explanations and use of demonstrations
7.	PK_CT_ACTIVITY	unplugged, pair programming, live coding etc.
8.	PK_CT_ALGO_TECH	by the use of a programming language or technological tool teach the skills/knowledge about A. abstraction by modeling how to problem solve or think about a problem in iterative and incremental ways, B. generalizing from one solution to another by showing how to identify common patterns i.e. through the use of functions and parameters, C. complex problems to simpler ones and show how to develop a solution in increments, D. showing how to design a model and decision making with conditions (e.g. through the use of flowcharts) before writing a program for solving the problem and E. showing how to solve a problem by presenting or explaining the solution to a problem in terms of a series of steps (by using relevant concepts, such as sequencing, loops, flow of control, conditions, etc).
9.	PK_CT_ALGO	outside the use of a programming language or technological tool teach the skills/knowledge about A. abstraction by showing how complex situations can be simplified or broken down into the most important elements as a way to get to the core of a problem, B. generalizing from one solution to another by showing how to identify common patterns (e.g. by showing how certain smaller problems apply to a larger scale in everyday life around you), C. decomposing a problem or task into smaller problems or subtasks to handle complexity and maintain a clear overview by visualizing it for students and taking them through each step, D. showing how to design a model and decision making with conditions (e.g. through the use of flowcharts) before writing a program for solving the problem and E. showing how to solve a problem by presenting or explaining the solution to a problem in terms of a series of steps by using relevant concepts, such as sequencing, loops, flow of control, conditions, etc.
10	PK_CT_DEBUG	teaching the skills and attitude necessary for debugging by building confidence and working towards correct code i.e. try things out as you go and make revisions based on what happens
11	PK_CT_GOALS	working/planning based on a collection of CT learning goals or an established CT-curriculum and thinking about the order of introducing each aspect of CT
12	PK_CT_ASSESS	sufficient means of CT-assessment
13	PK_CT_REUSEMIX	show how to do something based on (and expanding) what others or you have done (e.g. improve code by making it more modular and/or well structured)

Table D.4: Technology knowledge (TK) for computational thinking and its subcategories includes the teachers knowledge and skills about how to ...

#	LABEL/CODE	DESCRIPTION
14.	TK_CT_OPERATE	operate/use a variety of technologies
15.	TK_CT_CREATE	invent/create new technologies/tools
16.	TK_CT_SOLVE	solve a task using technical processes, methods, tools and recognize when information technology can assist or impede the achievement of a goal
17.	TK_CT_ADAPT	learn and adapt to new technologies.

Table D.5: Context knowledge (CX) for computational thinking is defined from the point of view explicated by Porras-Hernández and Salinas-Amescua [62] who proposed to regard context knowledge along two dimensions, i.e. scope (macro, mezzo, and micro level context) and actor (students' and teachers' inner and external context). The following subcategories can be identified:

#	LABEL/CODE	DESCRIPTION
18.	CX_CT_SCOPE_MACRO	defined by social, political, technological, and economic conditions at a global level that influence the value and worth of adding computer science / CT to the school curriculum
19.	CX_CT_SCOPE_MEZZO	defined by the social, cultural, political, organizational, and economic conditions settled in the local community and the educational institution about the value of computational thinking in children's lives.
20.	CX_CT_SCOPE_MICRO	the level that deals with in-class conditions for learning (e.g., available resources for computational thinking, available technologies, norms and policies, beliefs, expectations, teachers' and students' goals about computational thinking).
21.	CX_CT_TEACH_INT	teachers needs, preferences, misconceptions, learning difficulties, prior knowledge, self-efficacy and pedagogical beliefs.
22.	CX_CT_TEACH_EXT	teacher ethnicity, culture, community, and socioeconomic background
23.	CX_CT_PUPIL_INT	students needs, preferences, misconceptions, learning difficulties beyond programming and prior knowledge on CT
24.	CX_CT_PUPIL_EXT	students ethnicity, culture, community, and socioeconomic background

E

Survey

Toetsing & programmeren

Welkom bij deze enquête!
Mijn naam is Julian de Groot en ik ben master student aan de TU Delft en VO-docent informatica.

Door de toename in populariteit van digitale geletterdheid in het onderwijs zijn veel scholen begonnen aan curriculumvernieuwing onder andere met onderdelen van computational thinking.
Computational thinking richt zich op het zodanig formuleren van problemen en het uitdrukken van bijbehorende oplossingen dat een computer het uit kan voeren. Dergelijke zaken komen bijvoorbeeld aan bod bij programmeren.
Deze enquête richt zich op alle docenten die dergelijke lessen geven, dat binnenkort van plan zijn te gaan doen of er in geïnteresseerd zijn.
Als u bijvoorbeeld nu programmeerlessen geeft, van plan bent te gaan geven of er in geïnteresseerd bent dan is deze enquête iets voor u!

Het invullen van de enquête duurt ongeveer 15 tot 20 minuten. Alle gegevens worden geanonimiseerd als onderdeel van een wetenschappelijke studie en zullen niet worden gedeeld met derde partijen.
Niet volledig ingevulde enquêtes worden niet opgeslagen en kunnen niet worden gebruikt voor het onderzoek.

*Vereist

Over uzelf

Om een idee te krijgen wat de achtergrond is van onze deelnemers stellen wij u een aantal vragen over uzelf.

1. **Wat is uw geslacht?** *

   Markeer slechts één ovaal.

   - Man
   - Vrouw
   - Anders
   - Wil ik niet zeggen

2. **Wat is uw geboortejaar?** *

Docent bevoegdheid en ervaring

3. **Wat is uw hoogste docentbevoegdheid?** *

   Markeer slechts één ovaal.

   - Basisonderwijs (Pabo)
   - 1e Graads
   - 2e Graads
   - Geen

4. **Voor welk(e) vak/vakken heeft u deze hoogste docentbevoegdheid of bent u nog voor in opleiding?** *

5. **Hoeveel jaar ervaring heeft u met het geven van lessen programmeren?** *
Markeer slechts één ovaal per rij.

	geen	minder dan 1 jaar	1	2	3	4	5	meer dan 5 jaar
PO	◯	◯	◯	◯	◯	◯	◯	◯
VO onderbouw	◯	◯	◯	◯	◯	◯	◯	◯
VO bovenbouw	◯	◯	◯	◯	◯	◯	◯	◯

Algemene toelichting

In de volgende delen van de enquête wordt u een aantal vragen gesteld aangaande uw attitude omtrent toetsing.
U krijgt drie keer dezelfde vragen te zien binnen de volgende contexten: algemeen/uw vak en programmeer lessen.

Wij zijn daarbij geïnteresseerd naar uw attitude met betrekking tot toetsing. Onder toetsing verstaan we alle manieren die u kunt gebruiken om de voortgang van het leerproces te meten.

Wij laten u een lijst met stellingen zien. Elke vraag kan worden beantwoord door op een schaal van 1 tot 5 aan te geven in welke mate u het eens of oneens bent met deze stelling.
Probeer daarbij alle vragen zo snel en nauwkeurig mogelijk te beantwoorden.

6. **Voorbeeldvraag**
Markeer slechts één ovaal.

	1	2	3	4	5	
Zeer oneens	◯	◯	◯	◯	◯	Zeer eens

HET ONDERWIJS IN HET ALGEMEEN

Denk voor de volgende delen van de enquête aan het onderwijs in het algemeen. Hierbij moet u voorbij het vak denken dat u lesgeeft. Denk hierbij aan alle vakken op school.

HET ONDERWIJS IN HET ALGEMEEN

7. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor HET ONDERWIJS IN HET ALGEMEEN.** *
Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van de voortgang	◯	◯	◯	◯	◯
Toetsing hoort plaats te vinden gedurende een serie leeractiviteiten	◯	◯	◯	◯	◯
Toetsing hoort continu plaats te vinden om voortgang te evalueren	◯	◯	◯	◯	◯
Toetsing hoort informatie te bieden om het leren en onderwijzen aan te passen	◯	◯	◯	◯	◯
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te volgen en te sturen door middel van feedback	◯	◯	◯	◯	◯

HET ONDERWIJS IN HET ALGEMEEN

8. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor HET ONDERWIJS IN HET ALGEMEEN.** *
Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van cijfers	◯	◯	◯	◯	◯
Toetsing hoort plaats te vinden aan het einde van een serie leeractiviteiten	◯	◯	◯	◯	◯
Toetsing hoort op een uitgekozen moment plaats te vinden om te bepalen wat een leerling dan weet	◯	◯	◯	◯	◯
Toetsing hoort informatie te bieden om leerlingen in een andere stroom of op een ander niveau te plaatsen	◯	◯	◯	◯	◯
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te evalueren door te vergelijken met een benchmark of norm	◯	◯	◯	◯	◯

HET ONDERWIJS IN HET ALGEMEEN

9. **Geef voor ieder van de volgende toetsvormen aan in hoeverre u deze geschikt vindt voor HET ONDERWIJS IN HET ALGEMEEN.** *
Markeer slechts één ovaal per rij.

	1 (Zeer ongeschikt)	2	3	4	5 (Zeer geschikt)
Een (niet-digitale) schriftelijke toets met open en / of gesloten vragen	◯	◯	◯	◯	◯
Een digitale toets met open en / of gesloten vragen	◯	◯	◯	◯	◯
Een reeks kleine, niet op de computer uit te voeren, praktische opdrachten	◯	◯	◯	◯	◯
Een reeks kleine, op de computer uit te voeren, praktische opdrachten	◯	◯	◯	◯	◯
Een grote praktische opdracht (project)	◯	◯	◯	◯	◯
Een mondeling	◯	◯	◯	◯	◯

UW VAK

Denk voor de volgende delen aan uw vak of vakken die u het prettigst vindt om te onderwijzen.

UW VAK

10. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor UW VAK.** *
Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van de voortgang	◯	◯	◯	◯	◯
Toetsing hoort plaats te vinden gedurende een serie leeractiviteiten	◯	◯	◯	◯	◯
Toetsing hoort continu plaats te vinden om voortgang te evalueren	◯	◯	◯	◯	◯
Toetsing hoort informatie te bieden om het leren en onderwijzen aan te passen	◯	◯	◯	◯	◯
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te volgen en te sturen door middel van feedback	◯	◯	◯	◯	◯

UW VAK

11. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor UW VAK.** *
Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van cijfers	◯	◯	◯	◯	◯
Toetsing hoort plaats te vinden aan het einde van een serie leeractiviteiten	◯	◯	◯	◯	◯
Toetsing hoort op een uitgekozen moment plaats te vinden om te bepalen wat een leerling dan weet	◯	◯	◯	◯	◯
Toetsing hoort informatie te bieden om leerlingen in een andere stroom of op een ander niveau te plaatsen	◯	◯	◯	◯	◯
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te evalueren door te vergelijken met een benchmark of norm	◯	◯	◯	◯	◯

UW VAK

12. **Geef voor ieder van de volgende toetsvormen aan in hoeverre u deze geschikt vindt voor UW VAK.** *
Markeer slechts één ovaal per rij.

	1 (Zeer ongeschikt)	2	3	4	5 (Zeer geschikt)
Een (niet-digitale) schriftelijke toets met open en / of gesloten vragen	◯	◯	◯	◯	◯
Een digitale toets met open en / of gesloten vragen	◯	◯	◯	◯	◯
Een reeks kleine, niet op de computer uit te voeren, praktische opdrachten	◯	◯	◯	◯	◯
Een reeks kleine, op de computer uit te voeren, praktische opdrachten	◯	◯	◯	◯	◯
Een grote praktische opdracht (project)	◯	◯	◯	◯	◯
Een mondeling	◯	◯	◯	◯	◯

LESSEN PROGRAMMEREN

Denk voor de volgende delen aan lessen waarbij u leerlingen leert problemen zodanig te formuleren en de bijbehorende oplossingen uit te drukken zodat een computer het uit kan voeren. Dergelijke zaken komen bijvoorbeeld aan bod bij programmeren.

LESSEN PROGRAMMEREN

13. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor LESSEN PROGRAMMEREN.** *

Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van de voortgang	⬭	⬭	⬭	⬭	⬭
Toetsing hoort plaats te vinden gedurende een serie leeractiviteiten	⬭	⬭	⬭	⬭	⬭
Toetsing hoort continu plaats te vinden om voortgang te evalueren	⬭	⬭	⬭	⬭	⬭
Toetsing hoort informatie te bieden om het leren en onderwijzen aan te passen	⬭	⬭	⬭	⬭	⬭
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te volgen en te sturen door middel van feedback	⬭	⬭	⬭	⬭	⬭

LESSEN PROGRAMMEREN

14. **Geef voor ieder van de volgende stellingen aan in hoeverre u het eens bent voor LESSEN PROGRAMMEREN.** *

Markeer slechts één ovaal per rij.

	1 (Zeer oneens)	2	3	4	5 (Zeer eens)
De prestaties van leerlingen horen te worden gemeten aan de hand van cijfers	⬭	⬭	⬭	⬭	⬭
Toetsing hoort plaats te vinden aan het einde van een serie leeractiviteiten	⬭	⬭	⬭	⬭	⬭
Toetsing hoort op een uitgekozen moment plaats te vinden om te bepalen wat een leerling dan weet	⬭	⬭	⬭	⬭	⬭
Toetsing hoort informatie te bieden om leerlingen in een andere stroom of op een ander niveau te plaatsen	⬭	⬭	⬭	⬭	⬭
Toetsing hoort gebruikt te worden om het leerproces van leerlingen te evalueren door te vergelijken met een benchmark of norm	⬭	⬭	⬭	⬭	⬭

LESSEN PROGRAMMEREN

15. **Geef voor ieder van de volgende toetsvormen aan in hoeverre u deze geschikt vindt voor LESSEN PROGRAMMEREN.** *
Markeer slechts één ovaal per rij.

	1 (Zeer ongeschikt)	2	3	4	5 (Zeer geschikt)
Een (niet-digitale) schriftelijke toets met open en / of gesloten vragen	⬭	⬭	⬭	⬭	⬭
Een digitale toets met open en / of gesloten vragen	⬭	⬭	⬭	⬭	⬭
Een reeks kleine, niet op de computer uit te voeren, praktische opdrachten	⬭	⬭	⬭	⬭	⬭
Een reeks kleine, op de computer uit te voeren, praktische opdrachten	⬭	⬭	⬭	⬭	⬭
Een grote praktische opdracht (project)	⬭	⬭	⬭	⬭	⬭
Een mondeling	⬭	⬭	⬭	⬭	⬭

PROGRAMMEER TOETSEN

Denk voor de volgende delen aan de frequentie en de toetsvormen die u gebruikt voor toetsing bij lessen programmeren.

PROGRAMMEER TOETSVORMEN

16. **Geef voor ieder van de volgende toetsvormen aan hoe frequent u deze gebruikt voor LESSEN PROGRAMMEREN.** *
Markeer slechts één ovaal per rij.

	0 (Nooit)	1 (Weinig)	2	3	4	5 (Veel)
Een (niet-digitale) schriftelijke toets met open en / of gesloten vragen	⬭	⬭	⬭	⬭	⬭	⬭
Een digitale toets met open en / of gesloten vragen	⬭	⬭	⬭	⬭	⬭	⬭
Een reeks kleine, niet op de computer uit te voeren, praktische opdrachten	⬭	⬭	⬭	⬭	⬭	⬭
Een reeks kleine, op de computer uit te voeren, praktische opdrachten	⬭	⬭	⬭	⬭	⬭	⬭
Een grote praktische opdracht (project)	⬭	⬭	⬭	⬭	⬭	⬭
Een mondeling	⬭	⬭	⬭	⬭	⬭	⬭

Ga naar vraag 17.

Afronding

17. **Maak de volgende zin af: Iets dat zou helpen omtrent toetsing bij lessen programmeren is ...**
*

____

F Survey results open question

Maak de volgende zin af: Iets dat zou helpen omtrent toetsing bij lessen programmeren is ...

Contact met collega's. Een kader met daarin gewenste niveau's per leerjaar
Korte opdrachten met feedback
Kleine praktische opdrachten
(eigen) standaardisering van de toetscriteria
Slimmere educatieve compilers / interpreters bovenop de standaard compiler/interpreter.
Meer tijd
Effective Questioning
Duidelijke normen i.c.m. een leerlijn
Differentiatie en een grote hoeveelheid voorbeeld opdrachten/toetsen.
Een duidelijk idee van wat je wilt bereiken
Meer geautomatiseerde feedback
Leren door doen, leren structureel een probleem op te lossen.
Kleine opdrachten die leerlingen in een centraal toetsmoment kunnen maken als bewijs van hun vaardigheden
Minder cijfers/toetsen
Een context die de leerling aanspreekt
Voorbeeldtoetsen
Google Forms/quizzes met meer toetsingmogelijkheden
Goede toetsen
Een nulmeting om te kunnen beoordelen waar ieder kind individueel staat.
Een instrument dat frequentie / doorwerken kan meten.
(computer)tijd
Een digtaal werkboek
Een grote variatie aan opdrachten
Veel voorbeelden
Het ontwikkelen van grote(re) PO'S.
Goede opdrachten, uitwerkingen en normeringen voor de betreffende 'taal'.
Meer kennis over het kunnen toetsen van programmeren.
Zelftoetsend lesmateriaal, gevarieerde envoudige opdrachten om ontwikkelen van vaardigheid (meer dan kennis)
Een programma, dat ervoor zorgt dat voortgang op individueel niveau wordt gemeten en waar bij de docent directe ondersteuning kan aanbieden
Niet toetsen maar praktijkopdrachten
Meer praktijkgerichte opdrachten, samen met andere scholen.
Het praktisch te houden
Voorbeeld toetsen, kant en klare toetsen, lijst met aan te vinken vragen, nakijkmodel, ervaring van andere docenten bij een afgegeven toets.

Een database met voorbeeldvragen waaruit je je eigen toets kunt maken.
Een programmeeropdracht
Programmeeropdrachten laten maken b.v. een spelletje
Niet nodig
Automatische controle van (delen van) de code
Een duidelijke rubric
Creativiteit
Een benchmark
Een set opdracht waarbij extreme differentiatie mogelijk is.
Voorbeeld toetsen
Een voorbeeldtoets of module voor programmeren in mijn vakgebied.
Is een goede formatieve opdracht, waar leerlingen zelf leren wat ze nog niet juist doen
Het bepalen van complexititeit, gekoppeld aan onderwijsniveau. Ook zouden eindtermen per niveau gunstig zijn.
Het kunnen volgen van het denkproces
Puntenverdeling in de code kunnen typen

G
Discussion: Programming and CT

If we ask someone to think about CT, what is it exactly we expect them to think about? If we look at the definition from Wing [80] and the definitions used in literature based on her work [4, 11], there seems to be no definitive answer on what CT truly is beyond its generic component: problem solving.
Such a generic component could be interpreted in a way that every person, even someone who says they are thoroughly planning a vacation, is practicing computational thinking. If we are to believe that CT is unique because it requires us to formulate and solve problems in such a way that they can be solved by the computer we can argue that it is no different than the same thinking processes that take place in programming education.
This brings us to the following questions:

Can you teach CT without teaching anything on programming?
Some could argue you can if you look at CT as 'problem solving' which is subject specific. In this scenario one could reason that this type of thinking already existed before the actual creation of the computer, where solutions to problems were described mathematically forming algorithms on paper. Thus if we look at CT as 'problem solving by thinking how a computer can do this for us in a computerized world', it inevitably means we cannot teach CT without also teaching parts of programming. If we then want to teach programming we would have also practice the same steps of thinking, abstraction, generalization and decomposition that is currently being labeled as CT.

If programming did not exist, would there still be any computational thinking?
We could argue that without the existence of programming (e.g. during the time computers did not exist) CT would inevitably be reduced to 'problem solving' and is therefore too generic. This could imply that the term computational thinking should not be used, but instead its vessel (e.g. programming) should be given a more prominent position as discussed in 2.3.3. It is possible to visualize this in a Venn diagram with an implication based on the idea that CT (C) is problem solving in a computerized world (thinking about how a computer can solve a problem) as a subset of Programming (P).

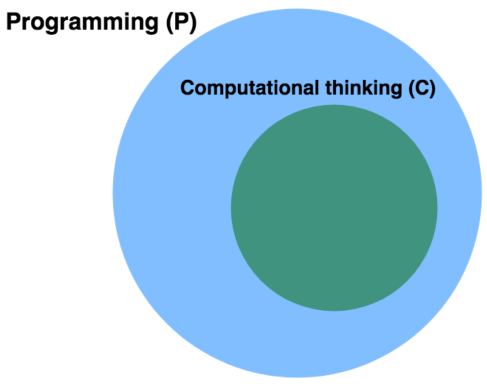


Table G.1: A possible implication for CT and programming

#	Implication	Result
1.	We teach CT ->We teach P	TRUE
2.	We teach CT ->We don’t teach P	FALSE
3.	We don’t teach CT ->We teach P	TRUE
4.	We don’t teach CT ->We don’t teach P	TRUE

1. If we teach CT we are also teaching a part of programming; hence TRUE.
2. If we teach CT it is impossible to not be teaching any part of programming; hence FALSE.
3. If we don’t teach CT we can still be teaching parts of programming; testing/debugging, reuse/remix, syntax. (coding), expressing yourself with code.. ; hence TRUE.
4. If we don’t teach CT we can also not be teaching any parts of programming; hence TRUE.