International Journal of Computer Science Education in Schools, Jan 2018, Vol. 2, No. 1
ISSN 2513-8359

Computational Thinking in Secondary Education: Where does it fit? A systematic literary review

James Lockwood[1]
Aidan Mooney[1]

[1]Maynooth University

DOI: 10.21585/ijcses.v2i1.26

Abstract

Computational Thinking has been described as an essential skill which everyone should learn and can therefore include in their skill set. Seymour Papert (Papert, 1980) is credited as concretising Computational Thinking in 1980 but Jeanette Wing (Wing, 2006) popularised the term in 2006 and brought it to the international community's attention. Since then, increased focus and attention have been placed on Computational Thinking and more and more research has been conducted on Computational Thinking in education.

The first aim of this systematic literary review is to give second-level educators who are looking to include Computational Thinking into their schools and classrooms ideas and options when looking at how to achieve this. The hope is also to present reasons as to why it is important to teach Computational Thinking, along with potential issues.

Secondly, we aim to give education researchers an overview of what work has been carried out in the domain, as well as potential gaps and opportunities that still exist. Thirdly, this is the first stage in a longer-term project to develop a Computational Thinking based curriculum which is taught using Computer Science. It is hoped that the problems, opportunities and ideas that are presented here will underpin this curriculum.

Overall it was found in this review that, although there is a lot of work currently being done around the world in many different educational contexts, the work relating to Computational Thinking is still in its infancy. Along with the need to create an agreed-upon definition of Computational Thinking lots of countries are still in the process of, or have not yet started, introducing Computational Thinking into curriculum in all levels of education. It was also found that Computer Science/Computing, which could be the most obvious place to teach Computational Thinking, has yet to become a mainstream subject in some countries, although this is becoming more common. Of encouragement to educators is the wealth of tools and resources being developed to help teach Computational Thinking as well as more and more work relating to curriculum development. For those teachers looking to incorporate Computational Thinking into their schools or classes then there are bountiful options which include programming, hands-on exercises and more.

Keywords

Computer Science Education; Computational Thinking; Literary review; K-12 Education;

1. Introduction

In Ireland, as with lots of other countries, Computer Science (CS) or Computing is not yet a subject that students can sit a state exam in. Although steps to include it have been taken (it will become examinable from 2020), so far, all that is available to students in the curriculum is a coding short course (for review purposes, 2017). Although programming is a very useful skill and one that can be beneficial to students in a wide variety of

careers and paths in life, it is not the only component of CS. (Lu & Fletcher, 2009) compare programming in CS to a literary analysis in English or proof construction in mathematics; it is a more advanced tool than say just being able to read and write. In her seminal paper Jeanette Wing (Wing, 2006) outlined how she believed Computational Thinking (CT) represents "a universally applicable attitude and skill set everyone...would be eager to learn and use". She believes all children should be taught CT, placing it alongside reading, writing and arithmetic in terms of its importance. Although academics since then have failed to agree on a universal definition she states that it "involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science." She states that it is not programming and that it means "more than being able to program a computer. It requires thinking at multiple levels of abstraction". In 2008 Wing gave a further description of CT (Wing, 2008). She discussed how CT is influencing research across disciplines and that it is a skill that is being and should be used and taught to everyone. She gives two visions which are as follows:

- CT will be instrumental to new discovery and innovation in all fields of endeavour
- CT will be an integral part of childhood education

She also poses questions to CS, learning sciences and education communities with one question being: "What are effective ways of learning (teaching) CT by (to) children?" This in turn raises further questions about what concepts to teach, the order in which these are taught, and which tools should be used to teach it. She also discusses technology, societal and science drivers towards CT.

Since Wing's first publication (Wing, 2006) a lot of work has been done around the world and at all education levels to try and incorporate CT into schools, colleges, afterschool clubs, etc. This has mainly been done through Computer Science or Computing classes/courses. As CT is vitally important to a Computer Scientist this makes sense, however, it should be noted that from the outset it is generally agreed that being able to think computationally, which includes skills such as decomposition, abstraction, algorithmic thinking and pattern matching, is of huge benefit to all disciplines.

As part of this movement researchers in our university designed a program. The hope was to introduce secondary school students, and teachers, to Computer Science through both programming and algorithms, with the hopes of improving the vital skill of CT in participating students (citation removed for peer-review purposes). Now in its 4th year, the program has been delivered in over 60 schools and to over 1000 students.

With the introduction of programming into the curriculum and the call from administrators and governments to include more CS content in schools, the hope of the group is to develop a course which will teach CS topics whilst focusing on teaching CT. To this end a systematic literary review has been carried out to see how other countries and institutions have included CT into their contexts.

The hope is that this review will be of benefit to other educators, researchers, teachers and industry members who hope to introduce this vital 21st century skill to the next generation.

2. Research Questions

The first stage of this review was to design the research questions to be addressed. To this end, several research questions were defined to cover the potential studies involved in looking at CT in secondary education. For the purposes of this review, we took secondary education to be from age 11/12 till the completion of final country or state-wide exams at aged 17/18. The defined questions are as follows:

- RQ1: What topics/subjects have been used to teach Computational Thinking?
- RQ2: What tools/methods have been developed/used to teach Computational Thinking?
- RQ3: What methods/tests/tools exist to test students' Computational Thinking ability/improvement?
- RQ4: Why is Computational Thinking important for educational institutions to incorporate into their curriculum? i.e. What benefits does CT have?
- RQ5: What problems/barriers are in place to incorporating Computational Thinking? Or what difficulties exist to introducing Computational Thinking into schools?

3. Method

3.1 Introduction

This systematic literature review is based on Kitchenham's method (Kitchenham & Charters, 2007) as applied to software engineering. This method of performing a review was chosen as the process is well documented and is derived from review processes that were previously well established in the medical community.

The paper outlines how to identify the need for the review, how to develop a strict protocol to follow for the review and how to report the findings from the review. The following steps are listed in the method:

- Identify the need for a systematic literature review and define your research questions.

  Addressed in Section 1 & 2.
- Carry out an exhaustive search for studies.

  Described in Section 3.2, 3.3 and 3.4.
- Assess quality of accepted studies.

  Discussed in Section 3.5 and the Appendix.
- Extract data from accepted studies.

  Presented in Section 3.6.
- Summarise and synthesise study results.

  Discussed in Section 5.

The correct application of these steps leads to a rigorous, exhaustive and reproducible meta-review (Kitchenham & Charters, 2007).

3.2 Search terms

In this study one primary search term was used, this was "Computational Thinking".

Secondary terms were used along with this and were as follows: "school", "secondary school", "high school", "post-primary school", "middle school", "second level", "benefits", "assessment", "test".

3.3 Resources searched

Using the search terms introduced above an extensive search of the following databases was carried out between October and November 2016:

- ACM Digital Library
- IEEE Xplore
- ERIC
- Google Scholar

3.4 Document selection

Inclusion and exclusion criteria were developed as recommended in the systematic review guidelines (Kitchenham & Charters, 2007). Texts were included that:

Directly answered one or more of the research questions

Were related to the teaching of Computational Thinking in second level educational institutions

Studies were excluded that:

- Were in the form of a book or grey literature (opinion pieces, technical reports, blogs, presentations etc.)
- Were not published in reputable (i.e. peer-reviewed) sources
- Did not answer any research questions
- Were not published in English

The first step was to eliminate papers based on their titles using the above criteria. Abstracts of papers that made it through this stage were then read with further cuts being made based on whether the content of the paper was relevant. The papers that made it through these stages were then studied in detail and assessed as described in Section 3.5.

3.5 Quality Assessment

The following questions were taken from Kitchenham's framework (Kitchenham & Charters, 2007) and each paper was analysed using them. The most relevant questions were taken from a set of 18 and applied to this review. These questions are:

- Question 1: How credible are the findings?
- Question 2: How well does the evaluation address its original aims and purpose?
- Question 3: How well was data collection carried out?
- Question 4: How clear and coherent is the reporting?
- Question 5: How has knowledge or understanding been extended by the research?
-

A scoring system was developed to analyse each paper and this scoring system is described below (For a full overview of the quality assessment one should consult Table 3 in the Appendix).

Question 1: Y (yes), the findings are very credible due to where the study is published, P (partly), the findings are credible, but the source is questionable, N (no), the findings nor the source are credible.

Question 2: Y (yes), the evaluation addresses the original aims and objectives, P (partly), the evaluation addresses the original aims and objectives implicit, N (no), the evaluation does not address the original aims and objectives.

Question 3: Y (yes), the data collection was carried out well and outlined clearly, P (partly), the data collection was carried out well but not outlined clearly, N (no), the data collection was not carried out well.

Question 4: Y (yes), the paper was clear and coherent, P (partly), the paper was somewhat clear and coherent, N (no), the paper was not clear and coherent.

Question 5: Y (yes), knowledge or understanding has been extended, P (partly), knowledge or understanding have been somewhat extended, N (no), knowledge or understanding has not been extended.

The scoring used was as follows. Where a question received a Y a score of 1.0 was applied, where a P was received a score of 0.5 was applied and where a N was received a score of 0.0 was applied.

Papers that did not receive a rating of three or above were excluding. Table 3 in the Appendix gives the results for all papers that were accepted on this basis.

3.6 Analysis of papers

Figure 1 shows the year the papers that were analysed in this literary review were published. It can be seen that little research was carried out on CT relating to second level education prior to 2010. The amount of studies has generally increased in recent years. The year 2016 is lower probably due to the timing of the searches, with some papers not having been published and/or made available at the time of search.

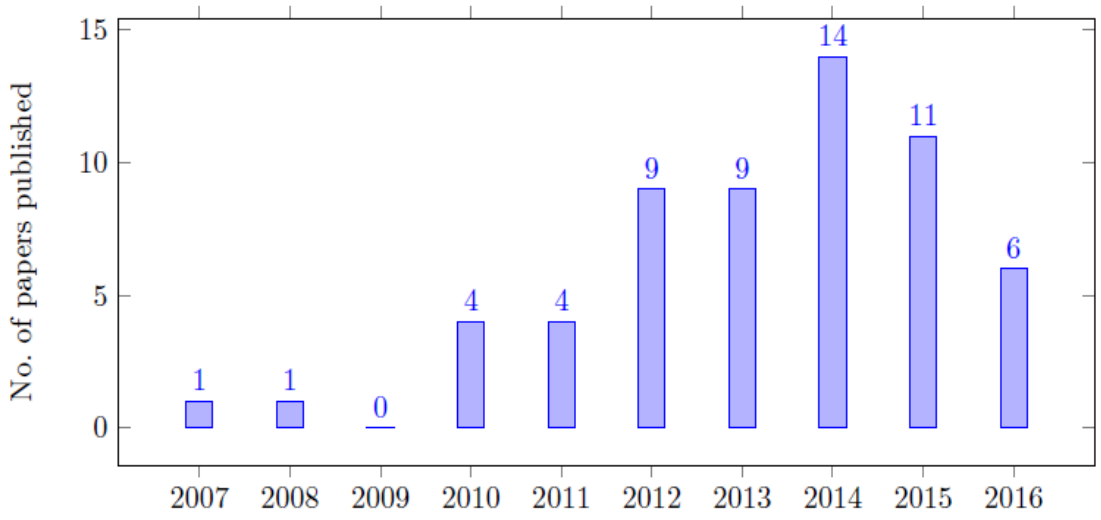

Figure 1. Year of Papers Published

Figure 2 shows the number of the papers included in the review by the database they were found in. One thing of interest is that no papers were included in the final corpus from ERIC. This was in part due to the small number found in the initial searches, only nine unique papers were returned using the search terms. We feel that this might show the need for more education researchers to be involved in studies on CT in secondary schools.

Note: Many papers found in ACM and IEEE were also found through Scholar, the number listed for Scholar includes only those found exclusively there.

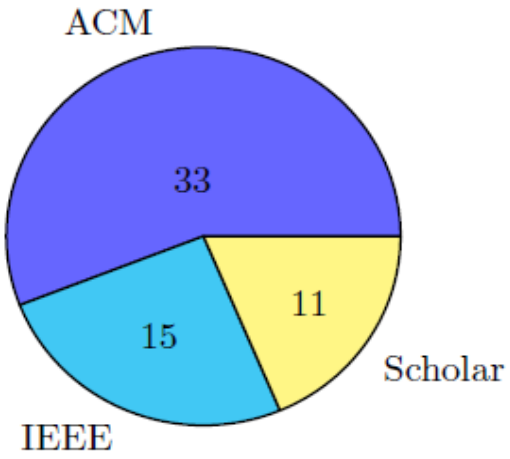

Figure 2. Number of papers sourced from each database

Table 1 and 2 provide further analysis of the papers included in this study. Table 1 provides a summary of all the papers including a general topic area and publishing location. These topic areas were assigned by us as reviewers and is given to show the general focus of the paper. However, it may be that other topics are also covered in the papers and they are not limited to the topics given here. Table 2 provides a list of all the publishing locations of the papers that are included.

Table 1. Summary of Papers

Reference	Year	Publishing location	Topic area
Atmatzidou & Demetriadis, 2016	2016	Journal	Teaching methodologies, results, definition of CT
Ahamed et al., 2010	2010	Conference	Teacher workshop, activity description, survey
Bargury et al., 2012	2012	Conference	Course design, teacher info
Basawapatna, Repenning, Koh, & Savignano, 2014	2014	Conference	Learning methods
Basu, Kinnebrew, & Biswas, 2014	2014	Conference	Assessment, tool, results
Blum & Cortina, 2007	2007	Bulletin	Teacher workshop, survey
Brancaccio et al., 2015	2015	Conference	E-learning, teacher training
Carvalho et al., 2013	2013	Workshop	Challenges to CT, CT attempts in education
Caspersen & Nowack, 2013	2013	Conference	Course design and learning methodologies
Chiprianov & Gallon, 2016	2016	Conference	& implementation, discussion
Cho, Pauca, Johnson & James 2014	2014	Conference	Teacher workshop, survey
Cortina et al., 2012	2012	Conference	Teacher workshop, survey
Curzon, 2013	2013	Conference	Activity description and outreach summary
Curzon, McOwan, Plant, & Meagher, 2014	2014	Conference	Teacher workshop, survey
Davies, 2008	2008	Conference	Teaching methodology, results
Falkner, Vivian, & Falkner, 2015	2015	Conference	MOOC, course design, survey
Folk, Lee, Michalenko, Peel, & Pontelli, 2015	2015	Conference	Course examples, teacher training
Fronza, El Ioini, & Corral, 2015	2015	Conference	Course description, results
Grover & Pea, 2013	2013	Conference	Activity description, learning method, results
Grover, Cooper, & Pea, 2014	2014	Conference	Assessment, curriculum design, survey, tests

Continued on next page

Reference	Year	Publishing location	Topic area
Grover, Pea, & Cooper, 2015	2015	Journal	Course design, assessment, results
Haddad & Kalaani, 2015	2015	Conference	Assessment, results
Imberman, Sturm, & Azhar, 2014	2014	Journal	Tools
Jenkins, Jerkins, & Stenger, 2012	2012	Conference	Teacher workshop
Kalelioglu, Gülbahar, & Kukul, 2016	2016	Journal	Literary Review
Koh, Basawapatna, Nickerson, & Repenning, 2014	2014	Conference	Assessment
Koh, Repenning, Nickerson, Endo, & Motter, 2013	2013	Conference	Course description, questions on it
L'Heureux, Boisvert, Cohen, & Sanghera, 2012	2012	Conference	Course design, activity design, survey
Lee et al., 2011	2011	Magazine	Activity suggestions
Li, Hu, & Wu, 2016	2016	Conference	Activity description, survey

Lishinski, Yadav, Enbody, & Good, 2016	2016	Conference	Assessment, results
Lye & Koh, 2014	2014	Journal	Literary review
Mannila et al., 2014	2014	Conference	CS education worldwide, survey
Mensing, Mak, Bird, & Billings, 2013	2013	Conference	Tools, activity description
Mooney et al., 2014	2014	Conference	Course design, survey
Morreale & Joiner, 2011	2011	Journal	Teacher workshop, survey
Morreale, Goski, Jimenez, & Stewart-Gardiner, 2012	2012	Journal	Teacher workshop, survey
Morreale, Joiner, & Chang, 2010	2010	Journal	Teacher workshop, survey
Nesiba, Pontelli, & Staley, 2015	2015	Conference	Course examples, teacher training, results
Pokorny & White, 2012	2012	Journal	Teacher workshop, survey
Repenning, Webb, & Ioannidou, 2010	2010	Conference	Tools
Ribeiro et al., 2013	2013	Workshop	Definition of CT, challenges to CT
Roscoe, Fearn, & Posey, 2014	2014	Conference	Course description
Shailaja & Sridaran, 2015	2015	Journal	Definition of CT, how can it be taught?
Sherman & Martin, 2015	2015	Journal	Assessment, results
Sysło & Kwiatkowska, 2014	2014	Conference	Activity design
Van Dyne & Braun, 2014	2014	Conference	Course design, assessment, results
Vieira & Magana, 2013	2013	Conference	Teachers workshop
Voogt, Fisser, Good, Mishra, & Yadav, 2015	2015	Journal	Definition of CS, How can it be taught?
Webb & Rosson, 2013	2013	Conference	Activity design, evaluation
Werner, Denner, Campe, & Kawamoto, 2012	2012	Conference	Assessment, results

Continued on next page

Reference	Year	Publishing location	Topic area
Wolz, Stone, Pulimood, & Pearson, 2010	2010	Conference	Curriculum design, activity design, pilot study, survey
Wolz, Stone, Pearson, Pulimood, & Switzer, 2011	2011	Journal	Teacher & student workshop, survey
Worrell, Brand, & Repenning, 2015	2015	Conference	Assessment, teaching methodology
Yadav et al., 2014	2014	Journal	Teacher module, survey
Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011	2011	Conference	Trainee teachers, MCQ
Yevseyeva & Towhidnejad, 2012	2012	Conference	Activity design
Zur Bargury, 2012	2012	Conference	Curriculum description and evaluation

Table 2. Publishing location

Location	No. of papers
ACM Annual Southeast Regional Conference	1
ACM Inroads	1
ACM Transactions on Computing Education	2
Australasian Computing Education Conference	2
Baltic Journal of Modern Computing	1
Computer Science Education	1
Computers in Human Behavior	1
Constructionism and Creativity	1
Education and Information Technologies	1
Emerging eLearning Technologies and Applications (ICETA)	1
Frontiers in Education (FIE)	7
Integrated STEM Conference (ISEC)	1
International Computer, Software and Applications Conference	1
International Conference on Engaging Pedagogy (ICEP)	1
International Conference on Interactive Technologies and Games (iTAG)	1
International Journal of Advanced Networking & Applications Special Issue	1
ITiCSE	5
Journal of Computing Sciences in Colleges	6
Robotics and Autonomous Systems	1
SIGCSE	12
SIGITE	2
Society for IT & Teacher Education International Conference	1
International Conference on Intelligent Tutoring Systems	1
Visual Languages and Human-Centric Computing (VL/HCC)	2
WiPSCE	3
Workshop-School on Theoretical Computer Science	2

4. Related Work

There have been several literary style reviews and overviews written on the current state of CT in schools in specific countries as well as suggestions for frameworks for CT. The following section presents several of these papers which were found during the search process. Some conclusions that can be drawn from these papers are as follows:

- Visualisation of the output of programming helps students learn CT (Lye & Koh, 2014)
- Having a scaffolding process and giving students an authentic problem helps students (Lye & Koh, 2014)
- The inclusion of CT aspects in the curriculum is relevant in all countries (Mannila et al., 2014)
- CT concepts can be taught through various subjects (Mannila et al., 2014)
- There are many positives to informal activities, however, they are not usually integrated into curriculum, so students may not see the connections of CT concepts. There is also no consensus on what should be taught (Mannila et al., 2014)
- The top five skills related to CT, as found in the literature, are (Kalelioglu et al., 2016):
    - Abstraction

  - Algorithmic thinking
  - Problem solving
  - Pattern recognition
  - Design-based thinking
- There are serious challenges in defining CT (Voogt et al., 2015) Suggested research gaps:
  - Explore more class-based interventions (Lye & Koh, 2014)
  - Explore more studies in computational practices and computation perspectives (Lye & Koh, 2014)
  - Examine the programming process (Lye & Koh, 2014)
  - There is no accepted or well-known definition of CT and an agreed-upon definition would be helpful to educators (Kalelioglu et al., 2016)
  - More work is needed in relation to integrating CT in education and integrating a CT curriculum (Voogt et al., 2015)

5. Findings

5.1 RQ1: What topics/subjects have been used to teach CT?

A huge range of topics has been used to teach CT to students. This includes explicitly teaching what CT is (Bargury et al., 2012; Grover et al., 2015; Li et al., 2016; Zur Bargury, 2012) and introducing them to concepts such as abstraction (Atmatzidou & Demetriadis, 2016; Caspersen & Nowack, 2013; Roscoe et al., 2014; Shailaja & Sridaran, 2015), modelling (Caspersen & Nowack, 2013), algorithms (Atmatzidou & Demetriadis, 2016; Folk et al., 2015; Grover et al., 2014, 2015; Grover, Pea, & Cooper, 2016; Mooney et al., 2014; Nesiba et al., 2015), decomposition (Atmatzidou & Demetriadis, 2016) and problem solving/critical thinking skills (Roscoe et al., 2014; Shailaja & Sridaran, 2015). CT has also been taught through a wide range of CS topics, these include robotics (Atmatzidou & Demetriadis, 2016; Bargury et al., 2012; Chiprianov & Gallon, 2016; Roscoe et al., 2014; Zur Bargury, 2012), web development (Bargury et al., 2012; Zur Bargury, 2012), searching and sorting (Folk et al., 2015; Li et al., 2016; Yevseyeva & Towhidnejad, 2012), circuit boards (Roscoe et al., 2014), logic (Caspersen & Nowack, 2013; Grover et al., 2015) and product design/software engineering (Bargury et al., 2012; Caspersen & Nowack, 2013; Roscoe et al., 2014; Zur Bargury, 2012).

As well as teaching CT independently, or through CS, there have been many efforts made to teach it in existing second-level subjects. One paper (Nesiba et al., 2015) discusses the DISSECT project and they present one approach in which CT practices are combined with composition and literature through a 12th grade English Literature course. One example of this is a unit on the play Macbeth. Using a drag and drop comic creation tool called ToonDoo (Inc., 2012) students had to use algorithmic thinking to create a story board of the scene. Another example in the English domain is where an overview of a summer school and after school program in Interactive Journalism are presented (Wolz et al., 2011, 2010). This was designed for middle school students and teachers to develop a competency and interest in CT. Students and teachers conducted research and interviews into how to develop news stories that are presented using Scratch animations, text and video. Others (Sysło & Kwiatkowska, 2014) discuss how CT concepts can be incorporated into traditional school mathematics. Examples of the topics and how they say they are linked to maths include:

- Representation of numbers – polynomials
- Reduction and composition - Given the sides of a triangle, is it a valid triangle?
- Approximation - rounding errors - quadratic equation

Other examples of the links CT has to Mathematics and English include Cartesian Coordinates and blogging (Mensing et al., 2013).

One interesting study was a report on the FACT (Foundations for Advancing Computational Thinking) curriculum, developed for middle school by Stanford University, USA. The curriculum was trialled both as an online version and face-to-face and it was found that the online version worked as well if not better than the face-to-face version (Grover et al., 2015). With the rising popularity of online courses and MOOCs, this is worth considering when developing any content.

It was also found that a collaborative classroom design helped students retain a high level of information in

situations where they may not be in the class for the whole semester. This design also allowed late-comers to be integrated quite easily (Worrell et al., 2015).

It can be seen from the presented papers that introducing CT does not have to be done exclusively through new courses or even through Computer Science. CT is a skill that can be used in a possibly surprising range of disciplines and can benefit students studying in any area. The ability to break down a problem and develop a manageable solution is one that all students will find useful in both their academic and work lives. However, it must be acknowledged that Computer Science is the most obvious place to teach CT to students. All the skills that make up CT (abstraction, algorithmic thinking etc.) are vital to a Computer Scientist. It is important though, that students who have never studied CS learn these skills, not only for their studies but also for their future careers.

5.2 RQ2: What tools/methods have been developed/used to teach CT?

Programming is the most popular way of implementing CT skills as well as being a great tool to show students how CT can be applied to real-world problems. Programming concepts such as conditionals (Bargury et al., 2012; Caspersen & Nowack, 2013; Grover et al., 2014, 2015, 2016; Roscoe et al., 2014; Shailaja & Sridaran, 2015; Zur Bargury, 2012), iteration (Bargury et al., 2012; Caspersen & Nowack, 2013; Grover et al., 2014, 2015, 2016; Li et al., 2016; Shailaja & Sridaran, 2015; Zur Bargury, 2012) and data/information handling (Bargury et al., 2012; Caspersen & Nowack, 2013; Grover et al., 2014, 2015, 2016; Zur Bargury, 2012) are often taught and can be either add-ons to CT or even perhaps key parts of it. Secondary school programming has been taught using a variety of technologies including Scratch (Bargury et al., 2012; Grover et al., 2015; L'Heureux et al., 2012; Li et al., 2016; Webb & Rosson, 2013; Zur Bargury, 2012), spreadsheets (Bargury et al., 2012; Zur Bargury, 2012), Python (Brancaccio et al., 2015; Mooney et al., 2014; Shailaja & Sridaran, 2015), Java (Shailaja & Sridaran, 2015), BASIC (Shailaja & Sridaran, 2015) and VB (Shailaja & Sridaran, 2015). It is worth noting that it has been shown that using a graphical programming language (Scratch, for example) or making simple simulations might allow algorithmic skills essential to CT to be taught in a way that is not as intimidating to students as a formal language (Java, Python etc.) (Basawapatna et al., 2014; Grover et al., 2015; Roscoe et al., 2014).

CT skills have also been taught using other "technologies" such as Minecraft (Roscoe et al., 2014), Printcraft (Roscoe et al., 2014) and Unplugged activities (Curzon, 2013; Folk et al., 2015; Li et al., 2016) which include the popular "CS Unplugged" activities, magic shows, hunting for a thief in CCTV footage and even a physical "Doll-house" with its own unique set of rules! These kinds of activities usually promote interaction and one course which used App Inventor specifically emphasised interaction as important when teaching CT (Grover & Pea, 2013). In their course, they included whole-class discussions and questions as well as working in pairs to develop apps. One thing noted about App Inventor is that it was found to compare favourably with Scratch and Alice but offers the benefits of having something very tangible to show (Grover & Pea, 2013). Fronza et al. (Fronza et al., 2015) also used App Inventor during their week-long summer school called Mobilise. Through this course, they aimed to use the "curiosity of students for developing mobile apps to introduce and teach CT via programming mobile applications". Both groups found that students enjoyed designing apps and the familiarity with apps can help to engage students (Fronza et al., 2015; Grover & Pea, 2013).

Another method of developing CT is through game design. One major group who consider this are the Scalable Game Design (SGD) group who are in the University of Colorado, Boulder, USA. They have created a spin-off company called AgentSheets Inc. (http://www.agentsheets.com/) which has developed AgentSheets and AgentCube. These are tools that let people create their own agent-based games and simulations and publish them on the Web through a user-friendly drag-and-drop interface. They state that building games teaches students Computer Science concepts, logic, and algorithmic thinking. One study of note presents a checklist for "computational thinking tools" (Repenning et al., 2010). They claim that to have an impact, a CT tool used in K-12 should fulfil the following conditions:

- Have a low threshold (should be easy to learn)
- Have a high ceiling (should allow sophistication)
- Scaffolds Flow (progressing in sophistication should be as straight forward as possible)
- Enables transfer (can students apply what they learn to other scenarios)
- Supports equity (is it interesting & engaging)
- Systematic and sustainable (they need to be used)

A variety of tools have been developed to assist the teaching of CT. Although several of these are in the early stages of development, it is encouraging to see efforts to make CT fun and accessible to students of all ages, genders and abilities. The benefits for educators are many and include a variety of options of how to integrate CT into their classrooms. Whether in a computer lab, a regular classroom or outside, in a one-on-one session or with a class of 30+ there is a tool out there which will suit educators needs; and if there isn't then the evidence suggests that there might well be soon!

5.3 RQ3: What methods/tests/tools exist to test students CT ability/improvement?

One form of assessment that has been designed for secondary schools is discussed by Werner et al. (Werner et al., 2012). In their course, students were taught using Alice and engaged with CT in a three-step progression called Use-Modify-Create (Lee et al., 2011). At the end of the semester, students were given up to 30 minutes to individually complete the "Fairy Assessment". They designed this "Fairy Assessment" as an Alice program which they hoped would analyse thinking algorithmically and making effective use of abstraction and modelling. The assessment involved solving three tasks which occurred during the playback of a narrative scenario in Alice. Failure in any of the three tasks did not result in the inability to complete the other two and they believed that for students to perform well they would "have to understand the narrative framework of the story underlying the program and to understand existing program instructions which create the framework". They found a large variety of results but in general their findings suggest that the Fairy assessment is a promising strategy for assessing CT because it is "motivating"; only 30 of the 311 participants did not attempt to modify the program. They also suggest the test was successful in picking up a range of CT across students and a variety of types of CT across the three tasks.

Another assessment was developed to calculate CT levels in students' models (Basu et al., 2014). These models were developed in "Computational Thinking in Simulation and Model-Building" (CTSiM) which is a learning environment that combines CT with middle school science. They designed pre- and post-tests to measure both science content and CT skills. The models students' developed were evaluated by comparing them against the "expert model" for that activity, which gave a "correctness" value.

Regarding the assessment of App Inventor projects, one paper introduced a rubric for analysing "mobile computational thinking" (MCT) (Sherman & Martin, 2015). They define MCT as a "superset of CT..., where the device changes location and context with its user". They claim that existing CT assessment tools do not cover these new ideas and therefore they tested their rubric in a mixed-major course that taught App Design using App Inventor. They measured 14 different properties.

This comprised of six "general CT" concepts:

- Naming
- Procedural abstraction
- Variables
- Loops
- Conditionals
- Lists

and eight MCT concepts:

- Screen interface
- Events
- Component abstraction
- Data persistence
- Data sharing
- Public web services
- Accelerometer
- Orientation sensors & location awareness

Each of these properties were rated on a "2-to 4-point scale, with increasing points representing more sophistication with the concept being measured". Detailed examples of this scaling system are presented in their paper.

The SGD group have also developed several ways of assessing students' CT skills, specifically when using their AgentSheets/Cubes game design software. Basawapatna et al. (Basawapatna et al., 2014) wanted to see if students could recognise "Computational Thinking Patterns" (CTP), which they defined as "abstract programming patterns that enable agent interactions not only in games but also in science simulations". Some examples of CTP's are generation, absorption, diffusion and transportation, which they describe in more detail in the paper. The SGD group have also built on this work by developing a Cyberlearning tool to help teachers see which high-level concepts students have mastered and which they are struggling with as students code in real-time (Koh et al., 2014). The system is called REACT (Real Time Evaluation and Assessment of Computational Thinking) and it displays the CTP's that students are currently implementing. REACT also shows the CTP's students haven't used, as well as the correctness of previously implemented patterns. REACT offers a variety of visualisation tools and is designed to be used with SGD teams AgentSheets & AgentCubes software.

Work relating to the testing and assessing of CT is in its infancy. Most of the examples presented in this section are in the early stages of development. Tools and methods do exist such as the "Fairy Assessment" and the tools developed by the Scalable Design Group but there is a need for more research in this area if CT is to become a common skill taught in schools. From the papers presented here as well as our experiences in teaching CS to students, we believe an adaptable system in which unique problems are produced for students to solve would be a hugely beneficial resource. This would allow for a range of problems that test all areas of CT but require no specific knowledge of the problem area. The Bebras problems aim to do this and so it is our belief that these problems could be used effectively as a central part of a CT assessment. We are currently developing a CT curriculum and will administer sets of Bebras problems to the students taking the course as a form of assessment, both prior to and upon completion of the course.

5.4 RQ4: Why is CT important for educational institutions to incorporate into their curriculum? i.e. What benefits does CT have?

Research shows that the earlier we can introduce students to computing the sooner we can get them attracted to the field and that it can have several side effects such as building confidence in dealing with complexity and dealing with open-ended problems (Yevseyeva & Towhidnejad, 2012).

Problem-solving skills can be extended and transferred (Koh et al., 2013) as well as improving students' analytical skills (Lishinski et al., 2016; Van Dyne & Braun, 2014). In relation to these it has been found that students' self-efficacy for computational problem solving, abstraction, debugging and terminology can be increased (Webb & Rosson, 2013). It has also been found that teaching CT can provide a better understanding to people that programming is about solving a problem and not just the code and improve female students' attitudes and confidence towards programming (Davies, 2008).

One especially interesting finding is that CT can be used as an early indicator and predictor of academic success and that CT scores correlate strongly with general academic success (Haddad & Kalaani, 2015). Another finding to note is that it as found that CT skills significantly improved as training proceeded and that students developed the same level of CT skills at the end, independent of age (junior high (15-year-olds) vs high vocational (18-year-olds)). The authors administered tests after four classes and after ten classes and found a significant improvement in results. They suggest that this shows CT skills need a considerable number of sessions to develop (Atmatzidou & Demetriadis, 2016). It was also found that it can be difficult for students to transfer from one platform to another. Sometimes sticking to one and using scaffolded examples is a good option (Webb & Rosson, 2013).

Although some encouraging signs have been seen, CT and research into it are still in the early stages. Therefore, long-term effects, as well as additional benefits still need to be researched further. The above findings are encouraging and show that CT is a beneficial skill to all, though more research is required before the extent of the impact of teaching CT can have on students is known.

5.5 RQ5: What problems/barriers are in place to incorporating CT? Or what difficulties exist to introducing CT into schools?

Although becoming more commonplace, the introduction of CT into second level education has been slow. This could be down to many reasons and some provided in the literature include:

- Lack of infrastructure i.e. a lack of computers (Carvalho et al., 2013) Rearrangement of the curriculum i.e. should CT be a new subject or incorporated into others? (Carvalho et al., 2013; Ribeiro et al., 2013)
- Government Policies are also cited as a potential issue and it is remarked that it is important that governments build workgroups to lead the inclusion of CT into education (Brancaccio et al., 2015; Ribeiro et al., 2013)
- People, and traditions in learning, mean that people might be reluctant to take up CT as they might be hard to convince that it is a skill they don't already possess (Ribeiro et al., 2013)

Students may not show much interest in learning (Mooney et al., 2014). One suggestion to counter this is via "stealth teaching". This is where the technical concepts are hidden to begin with and the student's interest is caught by the topic first (Yevseyeva & Towhidnejad, 2012).

Two other major roadblocks are the qualification and backgrounds of instructors i.e. 1) do they need a computing degree? (Bargury et al., 2012; Carvalho et al., 2013; Ribeiro et al., 2013; Zur Bargury, 2012) and, 2) strategies to disseminate CT i.e. workshops, textbooks etc. (Bargury et al., 2012; Carvalho et al., 2013; Zur Bargury, 2012). This leads to the question of how to give teachers, both trained and currently training, the skills, resources and knowledge to incorporate CT into their classrooms. To this end many training workshops have been run and below are studies found on these during this literary review.

The main aims of the workshops seem to be to inform teachers about what CT is, how it can be useful and then give them practical training to help integrate it into their classrooms. Information on what CT is, why it should be taught are a common starting point (Blum & Cortina, 2007; Curzon et al., 2014; Morreale et al., 2012; Morreale & Joiner, 2011; Morreale et al., 2010; Vieira & Magana, 2013). Other topics included career opportunities (Blum & Cortina, 2007; Morreale et al., 2012; Pokorny & White, 2012) and females in CS (Pokorny & White, 2012).

Most workshops appear to be multi-day events, usually occurring in the summer. Most common were two to four daylong (Ahamed et al., 2010; Blum & Cortina, 2007; Cho et al., 2014; Cortina et al., 2012; Vieira & Magana, 2013) workshops though a number were also a series of sessions (Curzon et al., 2014; Morreale et al., 2012; Morreale & Joiner, 2011; Morreale et al., 2010). What all had in common was that they contained a lot of practical, hands-on sessions where teachers are given a chance to try out different methods of teaching CT. Another way this has been done is through MOOC (Falkner et al., 2015); modules were developed and then videos made on those topics. They also used a Google+ group to foster collaboration.

A common theme was teaching educators the different ways in which CT can be taught through different programming languages and technologies. These included Scratch (Cho et al., 2014; Imberman et al., 2014; Pokorny & White, 2012; Vieira & Magana, 2013), Alice (Cortina et al., 2012; Morreale et al., 2012; Morreale & Joiner, 2011; Morreale et al., 2010), Python (Ahamed et al., 2010; Cortina et al., 2012; Jenkins et al., 2012), Java (Cortina et al., 2012; Morreale et al., 2012; Morreale & Joiner, 2011; Morreale et al., 2010) and App Inventor (Cho et al., 2014; Imberman et al., 2014). Another popular method was showing examples of "Unplugged" activities, where CT is taught without the need for computers (Blum & Cortina, 2007; Cho et al., 2014; Curzon et al., 2014; Imberman et al., 2014; Morreale et al., 2012; Morreale & Joiner, 2011; Morreale et al., 2010; Pokorny & White, 2012; Yadav et al., 2014). Examples included magic shows (Curzon et al., 2014), games (Curzon et al., 2014) and roleplaying (Yadav et al., 2014). This also includes teaching CT to those with no experience or through different subjects. Examples of this include DNA strings and matching algorithms (Blum & Cortina, 2007), food (Blum & Cortina, 2007), Harmonic Oscillation simulation (Morreale & Joiner, 2011), friction simulation (Ahamed et al., 2010), Towers of Hanoi (Yadav et al., 2011), debugging (how to fix a lamp) (Yadav et al., 2014, 2011) and mathematical topics (Ahamed et al., 2010; Jenkins et al., 2012). A final common aspect of these workshops is that many offered the teachers an opportunity to design lesson plans with the help of their colleagues as well as the workshop staff (Ahamed et al., 2010; Cho et al., 2014; Cortina et al., 2012; Falkner et al., 2015; Pokorny & White, 2012; Vieira & Magana, 2013).

One slightly different course is a one-week module designed for trainee teachers (Yadav et al., 2014, 2011). This did not include training in programming languages or technologies but focussed on day-to-day examples. The students were introduced to ideas such as problem identification, decomposition, algorithms and debugging.

They were given examples such as teaching algorithms through kinaesthetic activities and were shown a recursion example using the Towers of Hanoi (Yadav et al., 2014).

Some problems arose after these courses, which are important to highlight to teachers wanting to integrate CT into their educational institutes. One paper noted a lack of support from IT departments (for example when wanting to use Scratch) and the difficulty in and lack of time to refine lesson plans and develop competency in the material (Cho et al., 2014; Pokorny & White, 2012). This shows the need for the whole school/group to buy into the idea of CT and to understand what it is and its potential benefits. Another issue raised which links in with support from the whole school is that one group of teachers felt a classroom assistant would be beneficial (Cho et al., 2014).

Other findings from these workshops were a change in educators' views of what CT is (Blum & Cortina, 2007; Morreale et al., 2012, 2010; Pokorny & White, 2012; Vieira & Magana, 2013; Yadav et al., 2014, 2011). This usually meant a shift from either no knowledge of CT or thinking CT is programming (or at least heavily related to it) to an understanding of its application as a way of thinking/problem solving which can be applied to many parts of education and life. The first point, of teachers lacking understanding of what CT is, and even what CS is, is especially important for researchers and curriculum developers to note. Without an understanding, teachers will struggle to understand why they should take time to learn and teach these concepts. It also gives rise to the need for the area to come to a consensus of what we mean by CT, without a definition it is hard for integration and change to occur. Also, the practical sessions are often seen as hugely beneficial (Blum & Cortina, 2007; Falkner et al., 2015; Morreale et al., 2012) and give teachers the confidence and opportunities to introduce the lessons and ideas discussed into their classes (Falkner et al., 2015; Morreale et al., 2012; Pokorny & White, 2012; Yadav et al., 2014).

From these various papers and studies there are several ways in which teachers' enthusiasm for, knowledge of and ability to teach CT and CS in their classrooms can be improved/increased. The most common seem to be day-long workshops and workshops that are heavily practical in nature. The ideas, tools and lessons that are given during these workshops seem to give teachers a greater understanding of what CT is and how it can be useful for their students, whilst also giving them very practical ways to implement this in a variety of contexts. Wide-reaching initiatives such as Google's CS4HS can help teachers that otherwise lack the skills and knowledge to teach CT topics. Interestingly, it seems that one significant barrier to CS and CT in education is teachers and educators' misconceptions about what these are. One advantage of having teachers attend these training days and workshops is that these misconceptions and misunderstandings can be corrected, which is successfully done in most of the described studies. It can also be seen from these papers that teachers' willingness and interest in teaching CS/CT is vital in its implementation in secondary education.

6. Discussion

From the papers found and discussed in this review, the following findings are of significance:

6.1 More work needs to be done on the assessment of CT

Currently, there are several tool-specific assessment methods, such as those developed for Agentsheets, plus adapted methods such as Bebras problems. It is hard to judge their effectiveness at assessing CT and more work is required. This research gap demands attention as the widespread acceptance and introduction of CT requires educators to be able to assess students' abilities in this, as well as allowing researchers and educators to be able to assess their own courses and curricula.

6.2 CT is becoming more common place, but problems still exist

The introduction of CT, whether through CS, independently or through other subjects, is becoming more common place. This is encouraging from both a CS perspective, as it suggests more students will be introduced to CS, and an educational perspective, as it could lead to more analytically capable students. However, problems such as defining CT, disagreements on whether programming is an essential part of CT, and whether CT should be taught alongside pre-existing subjects, part of CS, or as a standalone module/course must be researched further.

6.3 Teacher workshops are vital and very effective

Teacher workshops that aim to inform, teach and instruct teachers about CT are the topic of many of the studies presented here. It can be seen from these studies that workshops are extremely effective at changing teachers' perceptions of what CT is and how it can be taught. They are also important in equipping teachers who don't have a technical background in CS to know how to teach CT through concepts such as programming. The potential for more long-term and formal teacher training is something that researchers should look at in the coming years.

6.4 Programming is key to advancing CT skills, but the basics and introduction can be done without computers and this needs to be done more

It can be seen from the number of studies, courses and workshops that incorporate programming in one guise or another that it goes hand-in-hand with teaching CT. However, it should be noted that CT skills, and even CS concepts, can be taught without programming and even without computers. Programming is one way in which CT skills can be evaluated as a student's ability to program a solution to a problem shows that they can use key skills such as abstraction, algorithmic thinking and decomposition. It is important that as work continues that the balance between using programming and teaching students to Computational Thinkers is struck. Although a useful skill, programming is more domain specific whereas it is widely agreed that CT is a skill that can be used across the educational spectrum and everyday life.

6.5 CT appears to have benefits, but further research is needed, especially on long-term effects

Due to the research of CT being in its infancy, very few long-term studies have been undertaken to see whether the teaching/learning of CT at different stages has an impact on things like academic success, career or study paths or even long-term problem-solving skills. Although it is commonly viewed as an essential skill that all students should be learning, so far, the evidence to support the inclusion of CT alongside reading, writing and arithmetic is limited, and more longer-term studies must be conducted to support this claim.

7. Conclusion

The aim of this paper is to provide teachers and researchers with an overview of the current research and work around teaching Computational Thinking at second level education. As stated in the introduction, CT is an important skill that we should be teaching students of all ages. To that end, the research questions posed in this paper were selected with the hope that educators from all contexts will be able to find examples of how CT can be incorporated into their classroom. Whether it is through a dedicated CT course/module, an already existing subject or just as a one-off event, CT can be taught in a fun and engaging way.

In addition, whilst learning CT, students can gain vital skills that can be applied across the curriculum as well as in daily and work life. There are also several different tools and ways in which one can apply them in classrooms, and although more work is needed, there are ways in which CT can be assessed. However, there are potential obstacles and difficulties when attempting to integrate CT. These include a lack of trained teachers and potential difficulties with government policy or school administrators. This literary review will also form the basis for a curriculum designed to teach CT and the hope is that providing teachers with a curriculum will lessen other common problems. These problems include preparation time and a fear in teaching these concepts, usually based around a lack of understanding. Until these problems are resolved it will be difficult for the widespread introduction of CT to take place.

References

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010, March). Computational thinking for the sciences: a three-day workshop for high school science teachers. In Proceedings of the 41st acm technical symposium on computer science education (p. 42-46).

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661-670.

Bargury, I. Z., Muller, O., Haberman, B., Zohar, D., Cohen, A., Levy, D., & Hotoveli, R. (2012, October).

Implementing a new computer science curriculum for middle school in Israel. Proceedings Frontiers in Education Conference (FIE), 1-6.

Basawapatna, A. R., Repenning, A., Koh, K. H., & Savignano, M. (2014, March). The consume-create spectrum: Balancing convenience and computational thinking in stem learning. In Proceedings of the 45th acm technical symposium on computer science education (p. 659-664).

Basu, S., Kinnebrew, J. S., & Biswas, G. (2014, June). Assessing student performance in a computational-thinking based science learning environment. In International conference on intelligent tutoring systems (p. 476-481). Springer International Publishing.

Blum, L., & Cortina, T. J. (2007, March). Cs4hs: an outreach program for high school CS teachers. ACM SIGCSE Bulletin, 39 (1), 19-23.

Brancaccio, A., Marchisio, M., Palumbo, C., Pardini, C., Patrucco, A., & Zich, R. (2015, July). Problem posing and solving: Strategic Italian key action to enhance teaching and learning mathematics and informatics in the high school. In Proceedings computer software and applications conference (compsac) (p. 845-850).

Carvalho, T., Andrade, D., Silveira, J., Auler, V., Cavalheiro, S., Aguiar, M., . . . Reiser, R. (2013, October). Discussing the challenges related to deployment of computational thinking in Brazilian basic education. In Proceedings of 2nd workshop-school on theoretical computer science (weit) (p. 111-115).

Caspersen, M. E., & Nowack, P. (2013, January). Computational thinking and practice: A generic approach to computing in Danish high schools. In Proceedings of the 15hth Australasian computing education conference (p. 137-143).

Chiprianov, V., & Gallon, L. (2016, July). Introducing computational thinking to k-5 in a French context. In Proceedings of the 2016 acm conference on innovation and technology in computer science education (p. 112-117).

Cho, S. S., Pauca, V. P., Johnson, D., & James, Y. V. (2014, March). Computational thinking for the rest of us: A liberal arts approach to engaging middle and high school teachers with computer science students. In Proceedings of the society for information technology & teacher education international conference (p. 79-86).

Cortina, T. J., Dann, W. P., Frieze, C., Ciminillo, C., Tananis, C., & Trahan, K. (2012, October). Work in progress: Activate: Advancing computing and technology interest and innovation through teacher education. In Proceedings frontiers in education conference (fie) (p. 1-2).

Curzon, P. (2013, November). cs4fn and computational thinking unplugged. In Proceedings of the 8th workshop in primary and secondary computing education (p. 47-50).

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014, November). Introducing teachers to computational thinking using unplugged storytelling. In Proceedings of the 9th workshop in primary and secondary computing education (p. 89-92).

Davies, S. (2008, October). The effects of emphasizing computational thinking in an introductory programming course. In Proceedings frontiers in education conference (fie).

Falkner, K., Vivian, R., & Falkner, N. (2015, January). Teaching computational thinking in k-6: The cser digital technologies mooc. In Proceedings of the 17th Australasian computing education conference (ace) (p. 30). ((Vol. 27))

Folk, R., Lee, G., Michalenko, A., Peel, A., & Pontelli, E. (2015, October). Gk-12 dissect: Incorporating computational thinking with k-12 science without computer access. In Proceedings frontiers in education conference (fie).

Anonymous for review purposes, A. (2017). Coding. (N.p. Web 28 Feb.).

Fronza, I., El Ioini, N., & Corral, L. (2015, September). Students want to create apps: leveraging computational thinking to teach mobile software development. In Proceedings of the 16th annual conference on information technology education (p. 21-26).

Grover, S., Cooper, S., & Pea, R. (2014, June). Assessing computational learning in k-12. In Proceedings of conference on innovation & technology in computer science education (p. 57-62).

Grover, S., & Pea, R. (2013, March). Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. In Proceedings of the 44th acm technical symposium on computer science education (p. 723-728).

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. Computer Science Education, 25 (2), 199-237.

Grover, S., Pea, R., & Cooper, S. (2016, February). Factors influencing computer science learning in middle school. In Proceedings of the 47th acm technical symposium on computing science education (p. 552-557).

Haddad, R. J., & Kalaani, Y. (2015, March). Can computational thinking predict academic performance? In Proceedings integrated stem education conference (isec) (p. 225-229).

Imberman, S. P., Sturm, D., & Azhar, M. Q. (2014). Computational thinking: expanding the toolkit. Journal of Computing Sciences in Colleges, 29 (6), 39-46.

Inc., J. (2012). Toondoo.com. http://www.toondoo.com. ([accessed June 23, 2017])

Jenkins, J. T., Jerkins, J. A., & Stenger, C. L. (2012, March). A plan for immediate immersion of computational thinking into the high school math classroom through a partnership with the Alabama math, science and technology initiative. In Proceedings of the 50th annual southeast regional conference (p. 148-152).

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. Baltic Journal of Modern Computing, 4 (3), p.583.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering version 2.3. Engineering, 45(4ve).

Koh, K. H., Basawapatna, A., Nickerson, H., & Repenning, A. (2014, July). Real time assessment of computational thinking. In Proceedings visual languages and human-centric computing (vl/hcc).

Koh, K. H., Repenning, A., Nickerson, H., Endo, Y., & Motter, P. (2013, March). Will it stick?: exploring the sustainability of computational thinking education through game design. In Proceeding of the 44th acm technical symposium on computer science education (p. 597-602).

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2 (1), 32-37.

L'Heureux, J., Boisvert, D., Cohen, R., & Sanghera, K. (2012, October). It problem solving: an implementation of computational thinking in information technology. In Proceedings of the 13th Annual Conference on information technology education (p. 183-188).

Li, W. L., Hu, C. F., & Wu, C. C. (2016, July). Teaching high school students computational thinking with hands-on activities. In Proceedings of the 2016 ACM conference on innovation and technology in computer science education (p. 371-371).

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016, February). The influence of problem solving abilities on students' performance on different assessment tasks in cs1. In Proceedings of the 47th ACM technical symposium on computing science education (p. 329-334).

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. ACM SIGCSE Bulletin, 41 (1), 260-264.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for k-12? Computers in Human Behavior, 41, 51-61. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in k-9 education. In Proceedings of the working group reports of the innovation & technology in computer science education conference (p. 1-29).

Mensing, K., Mak, J., Bird, M., & Billings, J. (2013, October). Computational model thinking and computer coding for us common core standards with 6 to 12-year-old students. In Proceedings emerging eLearning technologies and applications (ICETA) (p. 17-22).

Mooney, A., Duffin, J., Naughton, T., Monahan, R., Power, J., & Maguire, P. (2014). Pact: An initiative to introduce computational thinking to second-level education in Ireland. In Proceedings of International Conference on Engaging Pedagogy (ICEP).

Morreale, P., Goski, C., Jimenez, L., & Stewart-Gardiner, C. (2012). Measuring the impact of computational thinking workshops on high school teachers. Journal of Computing Sciences in Colleges, 27 (6), 151-157.

Morreale, P., & Joiner, D. (2011). Changing perceptions of computer science and computational thinking among high school teachers. Journal of Computing Sciences in Colleges, 26 (6), 71-77.

Morreale, P., Joiner, D., & Chang, G. (2010). Connecting undergraduate programs to high school students:

teacher workshops on computational thinking and computer science. Journal of Computing Sciences in Colleges, 25 (6), 191-197.

Nesiba, N., Pontelli, E., & Staley, T. (2015, October). Dissect: Exploring the relationship between computational thinking and English literature in k-12 curricula. In Proceedings frontiers in education conference (fie).

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

Pokorny, K. L., & White, N. (2012). Computational thinking outreach: reaching across the k-12 curriculum. Journal of Computing Sciences in Colleges, 27 (5), 234-242.

Repenning, A., Webb, D., & Ioannidou, A. (2010, March). 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 (p. 265-269).

Ribeiro, L., Nunes, D. J., Da Cruz, M., & Matos, E. (2013, October). Computational thinking: Possibilities and challenges. In Proceedings of 2nd workshop-school on theoretical computer science (weit) (p. 22-25).

Roscoe, J. F., Fearn, S., & Posey, E. (2014, October). Teaching computational thinking by playing games and building robots. In Proceedings international interactive technologies and games conference (itag).

Shailaja, J., & Sridaran, R. (2015, May). Computational thinking the intellectual thinking for the 21st century. International Journal of Advanced Networking & Applications Special Issue, 2015, 39-46.

Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. Journal of Computing Sciences in Colleges, 30 (6), 53-59.

Sysło, M. M., & Kwiatkowska, A. B. (2014). Learning mathematics supported by computational thinking. Constructionism and Creativity, 258-268.

Van Dyne, M., & Braun, J. (2014, March). Effectiveness of a computational thinking (cs0) course on student analytical skills. In Proceedings of the 45th acm technical symposium on computer science education (p. 133-138).

Vieira, C., & Magana, A. J. (2013, October). Using backwards design process for the design and implementation of computer science (CS) principles: A case study of a Colombian elementary and secondary teacher development program. In Proceedings frontiers in education conference (fie) (p. 879-885).

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies, 20 (4), 715-728.

Webb, H., & Rosson, M. B. (2013, March). Using scaffolded examples to teach computational thinking concepts. In Proceeding of the 44th acm technical symposium on computer science education (p. 95-100).

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The fairy performance assessment: measuring computational thinking in middle school. In Proceedings of the 43rd acm technical symposium on computer science education (p. 215-220).

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49 (3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, 366 (1881), 3717-3725.

Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. ACM Transactions on Computing Education (TOCE), 11, 2.

Wolz, U., Stone, M., Pulimood, S. M., & Pearson, K. (2010, March). Computational thinking via interactive journalism in middle school. In Proceedings of the 41st acm technical symposium on computer science education (p. 239-243).

Worrell, B., Brand, C., & Repenning, A. (2015, October). Collaboration and computational thinking: A classroom structure. In Proceedings symposium on visual languages and human- centric computing (vl/hcc) (p. 183-187).

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. ACM Transactions on Computing Education (TOCE), 14.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011, March). Introducing computational thinking in education courses. In Proceedings of the 42nd acm technical symposium on computer science

education (p. 465-470).

Yevseyeva, K., & Towhidnejad, M. (2012, October). Work in progress: Teaching computational thinking in middle and high school. In Proceedings frontiers in education conference (fie). Zur Bargury, I. (2012, July). A new curriculum for junior-high in computer science. In Proceedings of the 17th acm annual conference on innovation and technology in computer science education (p. 204-208).

Appendix

Table 3: Quality assessment of papers

Reference	Q1	Q2	Q3	Q4	Q5	Total
(Ahamed et al., 2010)	Y	Y	Y	Y	N	4
(Atmatzidou & Demetriadis, 2016)	Y	Y	Y	P	P/Y	4/4.5
(Bargury et al., 2012)	Y	Y	Y	Y	P	4.5
(Basawapatna et al., 2014)	Y	Y	N	Y	P	3.5
(Basu et al., 2014)	P	Y	Y	Y	Y	4.5
(Blum & Cortina, 2007)	Y	Y	Y	Y	Y	5
(Brancaccio et al., 2015)	Y	P	N/A	Y	P	3
(Carvalho et al., 2013)	Y	Y	N/A	Y	N	3
(Caspersen & Nowack, 2013)	Y	Y	N/A	P	P	3
(Chiprianov & Gallon, 2016)	Y	Y	P	Y	P	4
(Cho et al., 2014)	P	Y	Y	Y	P	4
(Cortina et al., 2012)	Y	Y	Y	Y	N	4
(Curzon et al., 2014)	Y	Y	Y	Y	P	4.5
(Curzon, 2013)	Y	Y	P	Y	N	3.5
(Davies, 2008)	Y	Y	Y	Y	Y	5
(Falkner et al., 2015)	Y	Y	Y	Y	P	4.5
(Folk et al., 2015)	Y	Y	P	Y	P	4
(Fronza et al., 2015)	Y	Y	N	Y	P	3.5
(Grover & Pea, 2013)	Y	Y	P	Y	P	4
(Grover et al., 2014)	Y	Y	Y	Y	Y	5
(Grover et al., 2015)	Y	Y	Y	Y	Y	5
(Grover et al., 2016)	Y	Y	Y	Y	Y	5
(Haddad & Kalaani, 2015)	P	Y	Y	Y	Y	4.5
(Imberman et al., 2014)	Y	Y	Y	Y	N	4
(Jenkins et al., 2012)	Y	Y	Y	Y	N	4
(Kalelioglu et al., 2016)	Y	Y	Y	Y	P	4.5
(Koh et al., 2013)	Y	P	Y	Y	P	4
(Koh et al., 2014)	Y	Y	P	Y	Y	4.5
(L'Heureux et al., 2012)	Y	Y	Y	Y	N	4
(Lee et al., 2011)	P	Y	N/A	Y	P	3
(Li et al., 2016)	P	Y	P	Y	N	3
(Lishinski et al., 2016)	Y	Y	Y	Y	Y	5

(Lye & Koh, 2014)	Y	Y	Y	Y	N	4
(Mannila et al., 2014)	P	Y	Y	Y	P	4
(Mensing et al., 2013)	Y	P	N	Y	P	3
(Mooney et al., 2014)	Y	Y	Y	Y	P	4.5
(Morreale & Joiner, 2011)	Y	Y	P	Y	P	4
(Morreale et al., 2010)	Y	Y	P	Y	P	4
(Morreale et al., 2012)	Y	Y	N	Y	P	3.5
(Nesiba et al., 2015)	Y	Y	Y	Y	Y	5
(Pokorny & White, 2012)	Y	Y	P	Y	P	4
(Repenning et al., 2010)	Y	Y	N/A	Y	Y	4
(Ribeiro et al., 2013)	P	Y	N/A	Y	Y	3.5
(Roscoe et al., 2014)	Y	Y	P	Y	P	4
(Shailaja & Sridaran, 2015)	P	Y	N/A	P	Y	3
(Sherman & Martin, 2015)	Y	Y	P	Y	Y	4.5
(Sysło & Kwiatkowska, 2014)	P	Y	N/A	Y	P	3
(Van Dyne & Braun, 2014)	Y	Y	Y	Y	Y	4.5
(Vieira & Magana, 2013)	Y	Y	Y	Y	P	4.5
(Voogt et al., 2015)	Y	Y	N/A	Y	Y	4
(Webb & Rosson, 2013)	Y	Y	Y	Y	P	4.5
(Werner et al., 2012)	Y	Y	Y	Y	Y	5
(Wolz et al., 2010)	Y	Y	Y	Y	P	4.5
(Wolz et al., 2011)	Y	Y	Y	Y	Y	5
(Worrell et al., 2015)	Y	Y	P	Y	P	4
(Yadav et al., 2011)	Y	Y	Y	Y	Y	5
(Yadav et al., 2014)	Y	Y	Y	Y	Y	5
(Yevseyeva & Towhidnejad, 2012)	Y	Y	N	Y	N	3
(Zur Bargury, 2012)	Y	Y	Y	Y	Y	5

Question 1: Y (yes), the findings are very credible due to where the study is published, P (partly), the findings are credible, but the source is questionable, N (no), the findings nor the source are credible.

Question 2: Y (yes), the evaluation addresses the original aims and objectives, P (partly), the evaluation addresses the original aims and objectives implicit, N (no), the evaluation does not address the original aims and objectives.

Question 3: Y (yes), the data collection was carried out well and outlined clearly, P (partly), the data collection was carried out well but not outlined clearly, N (no), the data collection was not carried out well.

Question 4: Y (yes), the paper was clear and coherent, P (partly), the paper was somewhat clear and coherent, N (no), the paper was not clear and coherent.

Question 5: Y (yes), knowledge or understanding has been extended, P (partly), knowledge or understanding have been somewhat extended, N (no), knowledge or understanding has not been extended.

The scoring used was as follows. Where a question received a Y a score of 1.0 was applied, where a P was received a score of 0.5 was applied and where a N was received a score of 0.0 was applied.