Computational Thinking Skills in Education Curriculum

Zuena Mgova

Master's thesis

ITÄ-SUOMEN YLIOPISTO

School of Computing

Computer Science

March 2018

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry,
Joensuu
School of Computing
Computer Science

Zuena Mgova: Computational Thinking Skills in Education Curriculum
Master's Thesis, 62p.
Supervisors of the Master's Thesis: PhD Solomon Sunday Oyelere | PhD Jarkko Suhonen
March 2018

Abstract

In the 21st century, among other important problem-solving skills needed is together with computational thinking skills. Computational thinking includes elements such as problems formulations, decomposition, algorithm, pattern recognition and abstractions thinking skills. This study informs education stakeholders, researchers, teachers and individual educators about what has been covered in computational thinking and the potential gaps and opportunities need to be fixed for successful implementation of computational thinking in school curricula. We reviewed the trends of computational thinking in education curricula including pedagogical tools for teaching and assessing computational thinking in classes. In general, we found out that currently there are no agreed-upon guidelines as to what computational thinking should include in all levels. The study also found that, computer science as an entrance for teaching computational thinking is not taught in many public schools in Tanzania which serves as our case study and century of focus. To pave the way for computational thinking in classes, there has to be a system of encouraging teachers with computational thinking resources, pedagogical tools, and teachers self-training tools. A platform for computational thinking resources was created to allow teachers, individuals and educations stakeholders get started with computational thinking concepts via Scratch.

Keywords: Computational thinking, computational thinking skills, Problems solving skills, computational thinking in education, computational thinking in school curricular

K-12: Compulsory education:

Acknowledgements

This thesis was done at the School of Computing, University of Eastern Finland during spring 2018.

Firstly, I would like to express my deepest gratitude to my supervisors PhD Solomon Sunday Oyelere | PhD Jarkko Suhonen for accepting to be the supervisors of my thesis work. I am also deeply grateful to my thesis reviewers for the constructive feedback.

Meanwhile, I want to also to thank Professor Matti Tedre for providing professional and personal inspiration during the co-founding of the thesis topic. Likewise, I am grateful to the Mehackit Company in Helsinki and Arts and Craft School of Robotics in Espoo for giving me an opportunity to learn from them.

I am sincerely grateful for the free-tuition in Finnish education system that has enabled me to achieve an excellent education through the fine comprehensive School of Computing at the University of Eastern Finland.

Finally, greatest thanks to my husband David Keno Kowanga, our lovely son Ivan David and my family for the love and support during my study time.

List of abbreviations

ACM	Association for Computing Machinery
IEEE	Institute of Electrical and Electronics Engineers
ICT	Information and Communications Technology
TCU	Tanzania Commission for Universities
STEM	Science, Technology, Engineering, and Mathematics
CSTA	Computer Science Teachers Association
ITSE	International Society for Technology in Education
HTML	Hypertext Markup Language
CS	Computer Science
API	Application programming interface
MIT	Massachusetts Institute of Technology
UK	United Kingdom
ACM	Association for Computing Machinery
CD-ROM	Compact Disc Read-Only Memory; medium for data storing
UEF	University of Eastern Finland

Contents

1 Introduction

Computational thinking has been declared as fundamental skills for solving complex problems we encounter every day (Wing, 2006). However, the school curriculum of the United Republic of Tanzania currently has not yet considered these fundamental skills to students. Computer science as an entrance to computational thinking has not yet also been declared as a compulsory subject to many public and private schools. Although the government has put efforts under the umbrella of improving ICT in schools, so far, all which have been done is the use of ICT tools in few schools. Although the use of ICT tools is very helpful to students and can be used in various careers, it is not the only part of computational thinking. The Computational thinking goes far beyond the use of ICT tools in school curricula. Webb et al. (2015) reported that the old ICT curriculum in the UK was also focused on the "so-called basic digital skills at the expense of a deeper understanding of concepts" which was far from the skills needed to solve the current problems in today's society.

We live in a digital society with complex problems which need computation thinking skills to deeply understand and solve them. The most cited article presented by Wing (2006) describes computational thinking as fundamental skills which should be induced to everyone. She emphasized on adding computational thinking skills to everyone is equally important as adding writing, reading and arithmetic skills. This is because computational thinking skills are universal skills set not only for a computer scientist but for everyone. According to Webb et al. (2015), developing computational thinking skills to students is among the problem-solving approach that helps learners to find solutions of any given problem from various disciplines including humanities, mathematics and science.

Educational stakeholders and researchers are encouraged to adapt computational thinking into school curricula to increase students understanding of problems presentation, reasoning at multiple levels of abstraction, and being able to develop an innovative solution (Lee et al., 2012). Barr et al. (2011) describes the reasons to include computational thinking in curriculum especially in this digital age of 21st century. In his article, he explains computational thinking as a thought process that gives students' abilities to formulate the solution of problems, organizing and analyzing data logically, presenting the abstractions of data in terms of models and simulations and automating the algorithmic thinking solutions. Moreover, the computational thinking seems as an approach for students to deeply understand computer science related subjects, STEM subjects and non-STEM subjects (Ar-raki et al., 2014; Miller et al., 2013). Students develop computational thinking skills when solving STEM or non-Stem problems using computational tools and, they develop thinking skills in that subject, for example, mathematics thinking skills.

We wanted to know what has been reported in the research studies for computational thinking skills in education curriculum during the past ten years. We prepared the following questions to understand the status of computational thinking studies in the education curriculum.

1. What is computational thinking in education curricula?
2. What are the core concepts of computational thinking in practice?
3. Which subjects of study has integrated computational thinking?
4. Which tools have been developed to practice computational thinking?
5. What tools have been developed to assess the transfer computational thinking?

We wanted to know the trends of computational thinking in education curriculum by investigating the current studies and analyze them into three main areas: 1) the meaning and core concepts of computational thinking, 2) the study subjects incorporated with computational thinking and 3) the tools to practice and assess the transfer of computational thinking skills.

We focus on computational thinking in education curriculum to inform and provide guidance to educators, designers and individual stakeholder who intend to integrate computational thinking into educational curricula.

2 Method

To address the issues posed on research questions, we performed a systematic literature review following the process described by Kitchenham's (2004) who explained it as a fundamental scientific activities for identifying, analyzing, evaluating, interpreting and summarizing what is already known in that subject. In systematic literature review small pieces of information are extracted from large quantities of information for digestion. The literature survey aimed to systematically analyze scientific published articles related to computational thinking in education curricular from 2006 to 2016. Since computational thinking in education curriculum is becoming more popular, articles before 2006 are excluded to this work. The data was collected in 2016, which means articles published after 2016 were not included.

2.1 Aims of the thesis

Computational thinking has been trending in different countries and each country is putting enough effort to define what computational thinking should look like in their school's curriculum especially in compulsory education.

The aims of this study were first, to find out the meaning of computational thinking in education curriculum and how it can easily be defined to some schools that has not yet declared it as a compulsory subject.

Second, to find out the main elements or core concepts of computational thinking that needs to be considered when integrating computational thinking into education curricular.

Third, to find out the study subjects that has been incorporated with computational thinking concepts. And lastly, to find out the tools that have been developed to practice and asses the transfer of computational thinking skills.

2.2 Articles selection

Searching for and identifying scientific articles related to computational thinking with keywords “computational thinking” and “education” was the first step to start with in the literature review. The published articles from scientific journals and conference proceedings were included to our work while unpublished articles, dissertations, blogs letter and opinions were excluded. With the use of keywords “computational thinking” and “education”, we searched the scientific articles in ACM, IEEE Explore, Springer Link databases and Google Scholar as an entrance to different databases.

The manual search through uef-finna, nelli, josku, science direct from the University of Eastern Finland was also performed as the main entrance to ACM, Springer Link, and IEEE digital libraries when using home computer. Google Scholar was the main entrance used to tell articles that were unpublished by either ACM or IEEE or Springer Link.

In this review, 86 articles are included. Including the articles in the review, each article needed to include at least one primary keywords which was “computational thinking” followed by the secondary keywords which are primary level, compulsory education, school curricular, elements, benefits, assessments, challenges and tools. A total of 105 articles were collected in data collection phase.

In the ACM Digital Library, we collected 61 articles and after the screening, we kept 42 articles which answered at least one of our research questions.

In the IEEE database, we collected 9 articles and after eliminating articles which were not answering at least one of the research question. Thus, 5 articles were kept.

In the Springer International Publishing AG, we found 9 articles and after the screening, 8 articles were kept. The same method was used for articles which were not in ACM, Springer or IEE

The same method was used for articles which were found in other scientific journals such as Eric, SAGE Education Journals, Science Direct, The Learning and Technology Library, Life Science Journal and others. 36 articles were found using the same methods 21 articles were kept.

2.3 Data analysis

Analyzing of the articles was the second step to follow in the data collection. The first step was to explore the gathered articles focusing on the abstract, discussion and titles. The aim was to find out if the selected articles were related to computational thinking in the education curriculum. The second step was to use Mendel's in analyzing the relationships of the articles selected. Some articles here were screened out because they were not relevant to computational thinking and not related to any of the articles collected. Focusing on our designed questions 89 articles were selected based on their focus and outcomes.

2.4 Thesis Structure

This thesis presents the literature review which covers the trends of computational thinking skills in the education curricular. The studies focused on the meaning and core concepts of computational thinking that can be considered when introducing computational thinking into classes. The focus was also in the tools developed to practice and asses the transfer of computational thinking skills to another discipline.

The work structure of the thesis is arranged as follows

Chapter 3 explains the related work, the similar articles of different authors describing computational thinking skills in education.

Chapter 5-8, serves as the focus of this thesis addressing the result of the research questions from study 1-5 as illustrated below

Study (1) describes the meaning of computational thinking from the most cited article written by (Wing, 2006) followed by different authors. The study concludes with the widely accepted definition of computational thinking from Google For Education and the operation definition from (CSTA & ISTE).

Study (2) covers the core concepts of computational thinking in practice. The most accepted concepts of computational thinking from Google For Education and (CSTA & ISTE) were suggested to be used as the guidelines on bringing computational thinking into practice.

Study (3) presents the study subjects which has integrated computational thinking. In general, the study concludes that computational thinking can be integrated into different subjects from humanities, sciences, engineering and technology

Study (4) covers the tools developed to practice computational thinking. Several tools were covered mainly visual programming language, robotics activities and game and game design

Study (5) defines tools developed to assess the transfer of computational thinking skills from the learners. And lastly, chapter 9 which discusses the overall of the study and suggesting what should be done on bringing computational thinking into classes.

3 Related work

Computational thinking offers numeral opportunities in enhancing the current educational curriculum settings including finding solutions of any given problems and designing intelligent solutions using different levels of abstraction and decomposition, algorithms, mathematical concepts in secure and helpful manner while understanding and determining the impact of the solutions chosen for its efficiency, economic and social impacts (Wing, 2006). We have identified several studies related to computational thinking in education curriculum, which have direct connections to this study.

Garneli et al. (2015) present a systematic literature review which focused on the trends of computing education in K-12 schools. The study describes how using programming tools can improve learning to K-12 students. The benefits and the challenges of using programming tools in K-12 were also explained. The focus was also on the most common instructional practices. Among the findings from their survey based on the three areas of concentration includes:

- The 33 modern programming languages with usable interfaces were listed
- Introducing computing skills to students via game design, tangible kits and robotics is very popular
- Choosing the right instructions methods with suitable parameters seems to be hard.

Duncan & Bell (2015) presents the extracted computer science curriculum from the CSTA (Computer Science Teachers Association), Australia and England. The extracted curriculum was purposeful for teaching computational thinking to primary school's pupils. It included topics such as algorithms and programming, presentation of data, digital devices applications and infrastructure, human and computers. These topics were summarized from computing, computational thinking plus computer science fields.

The age and time for adapting computational thinking were also described. Lastly, it was suggested that teaching computational thinking alone might be difficult. Therefore, embedding it in programming can be a better option.
Schulte et al. (2012) presents a survey overview report about computer science subject in schools. In a workshop conducted at Koli Calling conference in 2011, they received 84 responses from the online survey which was distributed in 22 countries. The survey respondents were from a different background including institutions, schools and industries. Using SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis according to (Schulte et al., 2012) it was found out that: Computer science is more available in upper secondary education. Most topics listed in upper education included: Introductory to programming which seen as the most important in upper secondary education, algorithms, advanced programming and projects and lastly HTML.

The goals mentioned for teaching computer science in upper secondary education were rated as develop thinking skills, develop problem-solving skills, learn to program, develop algorithmic thinking skills and databases skills which included design and queries.

For primary schools, computer science subject was seen on the use of computer and its applications. The overall teaching methods for all levels were rated as classroom-based teaching, the use of text processing mail, class-assignment for individual and small group and programming projects.
Among the problems of teaching computer science in schools were: Lack of trained teachers and computer science being perceived as an ICT. Moreover, the Trending of Computer Science in the curriculum was found out that:

- New computer science curricula to be introduced in many countries
- CS is already popular to high schools with demand

Qualls & Sherrel (2010) presents detailed review showing why computational thinking has been a focus across the fields in school curricula. Their study presented different courses developed in early computer science courses for teaching high school students. The courses developed focused on developing problems solving skills, concepts of programming and processes of software development. The study further describes the integration of computational thinking in College courses whereby Alice visual programming language was a choice for teaching programming concepts.

In their literature review Lye & Koh (2014) answered the below questions

- How has programming been incorporated into K-12 curricular?
- What is the performance outcome of student in computational thinking dimensions?
- Which tools have been used to develop computational thinking Skills?

Based on their finding (Lye & Koh, 2014) presents the following results: -

- The most programming language used by students to learn language and math's were sophisticated and easy to pick up
- Variables and loops are computational thinking concepts that students learn via programming
- Pair programming and mind-mapping improves students' performance
- Visualizing programming code output helps students to understand computational thinking

To help students understand computational thinking better (Lye & Koh, 2014) suggested the following

- Constructing of programs with scaffolding
- Reflection
- Reinforcing computational concepts
- Processing of Information

Mannila et al. (2014) present the current state of computational thinking K-9 education in multiple countries. They discuss the two forms (informal and formal) of education in multiple European countries plus the United States. Their contribution to the study was the paper survey distributed to K-9 teachers to measure how much computational thinking has been on practices in classrooms. Based on their findings from the survey distributed to 961 responders in five countries, reviews and documents (Mannila et al., 2014) reported that:

- Teachers are already practicing computational thinking in classroom
- Inclusion of computational thinking is relevant in all countries
- Adapting programming in class will deal with problem decomposition, algorithms and abstractions
- Computational thinking can be introduced into classes without enhancement of curriculum
- Special attention needed for teaching automation, simulations and parallelization

Mannila et al. (2014) concluded with the impact of programming in early education to develop computational thinking skills

Gross et al. (2014) present a review aimed to induce computational thinking skills to engineer students. In their review, they extracted six best practices of computational thinking to engineer's students. They introduced class labs and discussion related to MATLAB® and LEGO® Mindstorms® NXT robots. The course focused on familiarizing students in NTX hardware and peripherals. Students were introduced to RWTH Aachen MATLAB meets LEGO Mindstorms which was accompanied first by the introducing students to foundation skills which are: MATLAB as a programming language, mathematical concepts of engineering and signal processing basics.

Voogt et al. (2015) describe computational thinking overview from Papert's work on LOGO. In their paper, Voogt and his colleagues present the challenges and benefits of integrating computational thinking in formal and informal education. The detailed description of what and how computational thinking can be taught in education has been clear stated. Voogt et al. (2015) stressed the computational thinking curriculum content which should be adapted in classes. It was concluded that more study on how computational thinking can integrated across the subjects is needed.

In our study, we targeted to contribute to the discussion of computational thinking in education by 1) exploring the meaning and the concepts of computational thinking in education curriculum 2) Identifying the pedagogical tools for practicing and assessing computational thinking 3) discussing computational thinking in practice based on research exploration 4) Implementing a web-based platform to inform teachers, individual and education about how to start bringing computational thinking into classes.

4 Computational thinking in education curriculum

4.1 Introduction

The term computational thinking gained popularity in 2006 when (Wing, 2006) described it as thoughts processes that can either be executed by human or machine. In this article, she insisted that computational thinking should be adapted to every child and to everyone the same way we adapt to know how to read, write and perform basic mathematics. Introducing computational thinking in early education was the main concern pointed out because, even at their minimum age of four years, they can still cultivate the skills from engineering, technology, and computer programming while building computational thinking skills especially when engaged with construction-based activities (Bers et al., 2014). To benefit the potential of computational thinking then teaching computational thinking need to start in early education (Wing, 2006).

On her, later article Wing (2008) described computational thinking as analytical thinking to solve problems, design systems and understanding human behavior. In this article abstraction was a key important element for formulating solutions of any given problems. For example, to highlight a step by step of any problem to be executed requires abstractions to think which pieces of data should be removed and kept (Hu, 2011). Wing (2008) distinguished to what computational thinking means to everyone and what it means to scientists, engineers and professionals.

Wing's (2006) definition raised many discussions and debate among researchers from various disciplines especially in computer science, cognitive and educators. Their concerns were about the nature, definition, and application of computational thinking in real life. Wing (2006) argues that computational thinking involves all critical thinking skills such as problems solving, designing a system which draws some concepts from mathematical and engineering thinking.

Ater-Kranov et al. (2010) defined computational thinking to include critical thinking and problems solving skills accompanied with the help of a computer.

The Computer Science Teachers Association (CSTA and ITSE) presented their operation definition of computational thinking to include: - problem-solving, algorithms, data representation, modeling and simulation, and abstraction.

Hu (2011) proposes the definition of computational thinking which focused on abstractions. While Selby (2013) described computational thinking as thoughts process that gives learners the ability to think in terms of decomposition, algorithmically, evaluation and generalizations but not limited to problem-solving.

Barr & Stephenson (2011) described computational thinking to include formulating problems, logically organizing and analyzing data, presentation of data in abstractions, and automating solutions.

Following up, there seem to be multiple definitions of computational thinking. This brings some difficulties for educators who intend to integrate computational thinking in school curricular (Barr et al., 2011). However, despite the unclear definitions of computational thinking which leads to the difficulties of implementing computational thinking in education, multiple studies have outlined the key elements to consider when integrating computational thinking in the curriculum. Computational thinking is all about practices and increasing competences in problems solving and ability in formulating solutions (Werner et al., 2014; Hu, 2011). The focus of computational thinking has been shifted from caring about the definition to caring on how to make it happen in classes (Selby, 2013).

Hu (2011) describes the computational thinking elements to consider when integrating computational thinking into the school curriculum. These include: - Formulating solutions, algorithm, logic, pattern, simulation, decomposition and abstractions.

From Google For Education present computational thinking to include elements such as decomposition, pattern recognition, abstractions and algorithmic design.

The widely accepted operation definition from (CSTA & ISTE) defines computational thinking to include elements such as; problem-solving, algorithms, data representation, modeling and simulation and abstraction (CSTA, 2011). Similar elements have been described by (Selby, 2013; Werner et al., 2014; Barr et al., 2011) as core concepts to consider when practicing computational thinking in classes

Overall, many studies included at least one of the following elements when implementing computational thinking in classes: - Problem-solving, simulation, algorithm, abstractions, decomposition and logic are among the computational thinking elements which have.

4.2 Conclusion

In general, there are two definitions of computational thinking which are accepted by many computer science communities. The first definition is from The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) which defines computational thinking as processes of solving problems. The processes mentioned were: -

- Formulate problems in a way that computers can be used as helper tool to leaners
- Organize and analyze data logically
- Present data in models and simulation
- Automate solutions through algorithmic thinking
- Identify, analyze and implement the efficient and effective most possible solutions with combination of steps and resources
- Generalize and transfer the problem-solving process to another discipline

The definition from ISTE and CSTA was a result of the collaboration from higher education leaders, industry, and K–12 education. ISTE and CSTA did a survey of nearly 700 computer science teachers, researchers and practitioners who showed the strong agreement to the formed definition. The second definition is from Google For Education which described computational thinking as a problem-solving process that includes four elements; Decomposition, pattern recognition, abstractions and algorithms (Google, 2011). Both definitions describe computational thinking as a technique or process of solving problem computationally. The described process of solving problems defined by ISTE's and CSTA's and Google's can be used as a guideline for educators who want to integrate computational thinking into their school curriculum. Consideration of computational thinking and its related concepts should also be available in teaching methods such as games, robotics, visual programming and other pedagogical tools

5 The Core Concepts of Computational Thinking in Practice

5.1 Introduction

The core concepts of computational thinking are defined on specific computational thinking frameworks (Barr & Stephenson, 2011) designed to be used in classes. There are different frameworks for teaching computational thinking and each framework is designed to facilitate specific core concepts or elements of computational thinking.

Barr & Stephenson (2011) presents a framework of building computational thinking skills and its capabilities in the specific area of context. The presented framework includes the collection of data, analysis of data, presentation of data, decompositions of problem, abstraction, algorithms and procedures, automation, parallelization and simulation. The framework designed enhances the development and understanding of the subjects for example "computer science, mathematics, science, social studies and language arts" (Barr & Stephenson, 2011). The combination of these skills from processing and data analyzation enables learners to create artifacts that can be used in arts and STEM fields.

Grover & Pea (2013) described the following core concepts that build computational thinking skills: - Formulating problems, logically organizing and data analysis, data representation, algorithmic thinking, identifying, analyzing, and implementing possible efficient and effective solutions and generalizing and transferring problems-solving skills into other disciplines.

Brennan & Resnick (2012) proposes three concepts that build computational thinking skills:

- Computational concepts: The concepts students employ when coding. These include sequences, loops, events, parallelism and data.
- Computational practices: The problem-solving practice occurs during the coding process. The practices mentioned are "experimenting and iterating, testing and debugging, reusing and mixing, and abstracting and modularizing" (Brennan & Resnick, 2012).
- Computational perspective: This defines the ability of students to work in the group, understand their solutions by "expressing, connecting and questioning" (Brennan & Resnick, 2012).

Gouws et al. (2013) designed a computational thinking framework as a foundational resource center for creating computational thinking resources. The framework serves in a two-dimensional grid whereby the first dimension collects the set of skills that build computational thinking. The collected set of skills (Gouws et al., 2013) described in the first dimension includes "processes and transformations, models and transformation, patterns and algorithms, inference and logic, and evaluations and improvements". The second dimension determines the different levels where these skills may be practiced for example "recognize, understand, apply, and assimilate" (Gouws et al., 2013).

Weintrop et al. (2015) proposed a taxonomy framework consisting of four core practices of computational thinking to be applied in mathematics and sciences. These include "data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices".

Data practices skills: Serves many purposes in STEM subjects from collecting and to the use of data. The associated data skills overlap with computational thinking skills. The skills available in data practices are data collection, data creation, data manipulation, data analysis, and data visualization as (Weintrop et al., 2015) reports.

Modeling and Simulation skills: The skills learners build in modeling and simulation follows under; how to use Computational models to understand a concept, understanding how and why computational models work, how to assess computational models, how to use computational models to find and test solutions, and how to Build new/Extend the Existing Computational models

Computational Problem-Solving Skills: STEM subjects are full of problems and challenges which requires researching and developing the understanding of these problems. Since problem-solving is common in STEM field, using computational tools and strategies enables learners to deepen the knowledge and understanding of problems in STEM field. On the other hand, while learners solve problem and challenges in STEM field using computational tools, they develop computational thinking skills which enable them to interpret their solution in the computer. Learners develop problem-solving skills in this category from "Troubleshooting and Debug, Programming, Choosing Effective Computational Tools, Assessing Different Approaches/Solutions to a Problem, Developing Modular Computational Solutions, Using Problem Solving Strategies, and Creating Abstractions" (Weintrop et al., 2015).

Systems Thinking Skills: Computational thinking tools applied in STEM field is incorporated with skills which are crucial in identifying elements of a system and how they function and interact. These skills enable learners to understand and reason about the functionality and interaction of the elements involved in the system as the whole and its behavior. Practitioners of system thinking skills harvest the following skills as mentioned by (Weintrop et al., 2015): - Investigating the whole system, understanding the relationships within a system, thinking in levels, and visualizing systems, and identifying, understanding and managing complexity of problems

5.2 Conclusion

Several authors tried to implement computational thinking in different ways. Some implemented the computational thinking activities which help students to deeply understand either the concepts of algorithms, interactive, control or abstractions in specific fields which both aimed to enhance the development of computational thinking skills when solving the problem in given area of context. Most of the concepts implemented in are at least from ISTE’s & CSTA’s and Google’s core concepts of computational thinking which are believed to be the most accepted concepts in enhancing the development of computational thinking skills. Their operation definition included formulating problems, organizing and analyzing data logically, representing data in models and simulations through abstractions and algorithmic thinking are widely mentioned and accepted by many authors. ISTE’s & CSTA’s (2011) also described implementing the most efficient and effective solutions for Identifying and analyzing the steps and resources and using the similar problem-solving process to solve problems from different disciplines. Imparking the core concepts mentioned to learners believed to increase the ability in dealing with complexity, ambiguity, open-ended and difficult problems and it also increases collaboration (ISTE & CSTA, 2011). Google described computational thinking for educators to include decompositions, pattern recognition, algorithm and abstractions. Together the mentioned core concepts from ISTE & CSTA and Google can be used as a guideline for educators who want to integrate computational thinking into their school curriculum at least to include in practice one of the mentioned core concepts. The mentioned core concepts can also be used as a starting point for creators of computational thinking recourses to design resources or activities which at least follow the mentioned core concepts.

6 Subjects of Study that has integrated computational Thinking Concepts

6.1 Introduction

The computational thinking concepts have been integrated into multiple subjects from humanities, science, mathematics, engineering, arts and technology. The integration of computational thinking into the study subjects is purposely to increase the deep understanding of the subjects when solving problems of that subject using computational thinking tools. The following subjects of study have been adopted the use of computational thinking tools.

6.2 STEM Fields

Computational thinking has been used to facilitate deep understanding and knowledge development in STEM subjects. Different pedagogical and motivational models (Repenning et al., 2010) such as education robotics, enhanced activities, games and visual programming with integrated computational thinking concepts have been used to enhance the understating of the mentioned subjects. The subjects included in STEM fields are: - Science, Technology, Engineering and Mathematics. Computational thinking has been integrated into either one of the subject or by combining all subjects as we try to illustrate below: -

6.3 CT-STEM

Jona et al. (2014) discusses the approach of embedding of computational thinking (CT) in Science, Technology, Engineering and Mathematics (STEM) subjects. The aims of the study were to increase and attract students in computational courses while deepening the understanding in STEM fields. In their paper, they claimed that blending computational thinking in STEM courses reaches many audiences. And, it also addresses lack of competent teachers and lack of student's interests in computational thinking when it is taught as a separate subject. They further believed that blending computational thinking in STEM courses is important for students majoring in STEM courses as it builds students competence in computational thinking while increasing interests in computing fields to all students. Based on their investigation they concluded that bringing computational thinking in STEM classes provides the realistic view of STEM fields to students. This encourages students to prepare themselves for STEM careers especially when they are majoring in STEM fields

6.4 C3stem

Dukeman et al. (2013) present c3stem, a framework designed to address the issue of problem-solving skills in STEM fields. The framework aims to provide the engaging learning environment which allows students to solve the real world-problems exists in STEM fields. Their target was to expose students to the real-world problems that go beyond classes. The c3stem framework supported students to analyze the traffic flow with vehicle kinematics and basic driving behavior. In this study, the students worked on a reality-based challenge from traffic domain where they applied some concepts such as velocity and acceleration from physics. A traffic congestion problem is one of the STEM problems. With the use of traffic expert tool, students could be able to test their hypothesis and conclude their suggestions in which they assumed it will help the local community around.

In connection to that Weintrop et al. (2015) conducted a three-year study to support mathematics, chemistry, biology and physics teachers in embedding computational thinking into the science classroom. Embedding computational thinking in sciences classroom addresses the issue of fewer teachers to teach computational thinking as a separate subject, lack of student's interests in a course and underrepresented group in the course. In their paper, they further explain that bringing computational thinking classroom prepares students in modern STEM field disciplines especially when they practice the real-world challenges available in STEM fields. It also deepens the understanding of STEM fields when students' use computational thinking tools in solving STEM problems and finally it reaches many to many students. Their work to adapt computational thinking into the classroom was divided into three categories: -

- Defining computational thinking in STEM field
- Developing computational thinking activities based on STEM lessons
- Creating an online assessment for evaluating the success.

Among the issues the study was addressing is self-selecting of the subject which can cause some group to be underrepresented. However, during evaluation, they found out that there was no difference between boys and girls in computational thinking aptitude but a larger difference in attitudes and perceptions towards computational thinking. Girls in the first year of their study, the evaluation showed that they were not comfortable using computational tools and they had no interests in selecting STEM fields as their careers. Based on their feedback, the result was different when both students were exposed to the computational thinking embedded in STEM fields. Girls gained positive confidence in computational thinking in STEM while interests towards STEM fields and pursuing STEM fields as careers increased for both students

6.5 Computer Programming

Computational thinking has been a focus on improving programming skills. It has also been confusing subject to many researchers as to whether computer programming subject is the same as computational thinking. But it was found out that computer programming learners need computational thinking skills to tackle computer programming problems (Lu & Fletcher, 2009). Computer programming is like other subject's which demands the clear understanding of computational thinking concepts. To facilitate easy understanding of programming, different approaches including games with well-integrated computational thinking activities have been used. Games are believed to be one of the effective tools to enhance the development of problem-solving skills to students particularly in programming itself (Repenning et al., 2010). Because of its engagement environment and fun way, games are preferred as a technological tool to easily teach different subjects via well-designed programming activities. Among the games designed for teaching different subjects includes Serious game brought by (Kazimoglu et al., 2012) which was designed to help students learn computer programming. Students develop problem-solving skills when solving computer programming tasks.

In some school's games have been used to interests and retain students in some subjects such as STEM and computer programming (Kazimoglu et al., 2012). With well-designed hands-on activities incorporated in the subjects, games allow students to cultivate critical thinking skills which can be used to solve various problems from different disciplines. Computer programming has been considered as one of the difficult subjects for students. Researchers have found the easy way to teach and impart the programming skills to students is via games incorporated with computational thinking activities (Werner et al., 2014). The Serious game is one of the games for teaching programming courses in schools. The game was introduced to help the student in acquiring problem-solving skills required for solving computer programming tasks. The Serious game allowed students to create, apply, evaluate, debugs and stimulates algorithm for a specific problem. Playing the game and solving computer programming games seemed to develop the problem-solving skills.

The Scalable game design is another approach mentioned to bring lively programming in middle school curriculum. According to Repenning et al. (2010), the designed game program aimed to teach programming to students during summer time. Their main target was to attract more students in programming but also to enhance the development of computational thinking skills to students. In a survey done by (Repenning et al., 2010) shows that most of the students believe programming is a hard subject. Referring to one student who was excited to see the LEGO blocks during summer training and he asked what they would be doing with it. They explained that they are going to work with it for programming tasks. "Programming, oh no!" the student replied, "I know what is going to happen as (Repenning et al., 2010) reports. The teacher writes a program onto the blackboard, we type it into the computer it never works." This proves that applying different approaches especially games in computer programming will not only simplify the understanding of the subjects but also it will change the attitude of students towards what they believe. Teaching computational thinking concepts before getting into higher programming courses has also been pointed out by (Lu & Fletcher, 2009)

Werner et al. (2014) presents the approach of teaching programming via games. The game was integrated with computational thinking concepts. The game implemented allowed students to design their own games which enabled them to acquire the fundamentals elements of computation thinking such as problem-solving skills, algorithms skills, modeling skills, and abstraction skills. In their paper, they explain that, when students left to design their games, first they gain the knowledge of designing, language used and debugging. Secondly, they gain the strategies of designing, implementing and debugging the program using programming languages. Lastly, with game design, students become effective in choosing models for the problem, program and the actual program.

6.6 Ethics

Computational thinking has not only been applicable in STEM field or computer science fields, it is also applicable in ethics studies (Seoane-Pardo, 2016).” moral machines” was a course developed by (Seoane-Pardo, 2016) to allow students to make logical decisions in extreme situations when working in programming cars. Based on the reasoning of different ethical approaches, students were asked to logical develop a scheme of more ethical decision making for a machine to respond. Developing the logical schemes required students to think computationally

6.7 Sciences

Ahamed, et al. (2010) conducted a three-day workshop for high school students and science teachers. With the title “computational thinking for sciences”, they aimed to deepen the understanding connected between computer science, natural sciences and mathematics. The conducted workshop taught the following topics: - Computational thinking, simulations, biology, Chemistry, Physics, Probability, Mathematics, Careers in Computational Sciences and Lessons plans for teachers. The target of computational thinking tools in the mentioned subjects was to attract more students in computer science field.

Hambrusch, et al. (2009) introduced an introduction to computational thinking course for science majors. The course was designed by the faculty of computer science in collaboration with physics, chemistry and bioinformatics departments. Their aim was to lay the groundwork for computational thinking in physics, chemistry and bioinformatics. The course was taught via Python and Python libraries. Students were introduced to the concepts of Computational tools from Physics and Bioinformatics. The assessment process was done by proving assignments and four projects to students based on manipulation of digital Audio, Experimentation of Computational activities, Simulation of devices and Analyzing the interaction of Proteins. Feedback from the study shows that the designed course increased interest in computer science. And it further suggests that Python has an influence in teaching different disciplines

6.8 Computer Science

Computational thinking has also been a focus to teach complex concepts to students from computer science and education. Basawapatna et al. (2010) introduces scalable game design emerged with programming constructs and computational thinking concepts to students from middle to graduate level. The approach allows students to create their own games which enhance the understanding of complex concepts in the field. The study explains that using scalable games design to teach computer sciences exposes and interest’s students in computer science field. But also, it enables students to transfer the skills learned into another discipline.

Kafura & Tatar (2011) present computational thinking for student’s majors in computer science. The aim was to build the fundamental concepts of computer science to students in their early study. The Computational thinking was also used to increase steady enrolment in computer science fields for nonmajor’s students.

Cortina (2007) presents an introduction to computer science course for non-computer science students. The course was designed using the principles of computational thinking. Its target was to increase the interest and enrolment in computer science subjects.

Computational thinking has been used as an entrance for students to study computer science in South African Universities. Gouws et al. (2013) designed a computational thinking test to the first year's students to measure the ability of students to undertake computer science courses. Their aim was to determine what students possess and how much they have improved after their first semester.

6.9 Biology/life Sciences

Qin (2009) describes the course developed to teach Bioinformatics to undergraduate biology students. The course was intended to strengthen quantitative and critical computing skills to biology students. Students were taught computational thinking and hands-on learning activities which mapped the key concepts from computational thinking and Bioinformatics. For example, biological database (MYSQL) which allowed students to request data from the database.

Rubinstein & Chor (2014) introduces" Computational Approaches for Life Scientists" a course designed to bridge the gap between computational thinking and life sciences students. The course aimed to expose students to the concepts of algorithms, abstraction and logical thinking. To perform better in the course, having some basic programming was a prerequisite for allowing biologists students to practice computational thinking concepts. The introduced concepts increased the solid understanding of biological computational tools and the development of mathematical and computational thinking models applied in life sciences. The map of the course had three parts of concepts which are; biological content, computational topics and computational thinking concepts. Biological content contained topics such Biological networks, biological sequences, System simulations and Biological images. These four topics were spanned into computational topics and its related concepts. In their feedback, the understanding of basic programming is important for the implementation of computational thinking concepts which enables students to solve important problems in biology.

6.10 Journalism

Wolz et al. (2010) presents an Interactive Journalism program designed for middle school students and their teachers during summer school and after school. The program aimed to enhance the development of computational thinking to both teachers and students. It was designed to increase the competence and interest in computing fields, especially for non-computer science students. Students and teachers from arts, technology and guidance counselor worked together in conducting interviews and developing new stories which later were gathered in text, video and procedural animations in Scratch as story package. Based on their feedback they believe that the program changed the perceptions of students about programming because programming can be fun for students especially when using scratch. They also believed that the program increased competences and ability to program while increasing good writing process which is like software design process

6.11 Music

Ruthmann et al. (2010) developed Sound thinking course to induce computational thinking to art students. The course was initiated by the collaboration of computer science and music department. Students in this course were first exposed to web pages created using HTML and JavaScript API and embedding music file into their web pages created. Students were migrated from using HTML and JavaScript to scratch which seemed to be strong in inducing computational thinking concepts students. Using Scratch to create music, students could learn many computational concepts including loops, initialization, variables, changing variables algorithmically, modularization and event processing.

6.12 Conclusion

Integrating computational thinking into subject does not only enhance the development of computational thinking but it also increases the understanding of the subject. Subjects such mathematics and sciences in nature are among the subjects which demand thinking skills. The rapid change of practicing in the professional world is the reason for demanding thinking skills.

7 Tools Developed to Practice Computational thinking

7.1 Introduction

Developing computational thinking into the classroom have been approached using different tools from games, robotics activities, Scaffolded exercise, visual programming environment, the real-life projects and many as we try to illustrate below.

7.2 Scratch

Scratch is a web based visual programming language developed and released in 2005 by MIT Media Lab. It enables youth to develop computational thinking skills through creating their own interactive animations, games, and stories that can be shared in an online community. Scratch which has over 29 million shared projects as for now has claimed to increate creativity and collaboration to young leaners. Learners snapped together programming-instruction blocks when creating interactive stories, games, animations, and simulations in scratch environment. It is among the active visual programming language which has been approached to teach computational thinking as we describe below: -

7.3 DISSECT

Peel et al. (2015) presents DISSECT (Discovering Science through Computational Thinking) an intuitive program organized in New Mexico University to incorporate computational thinking to the 6th grade. Variously modules were introduced in classrooms including scratch. Teaching through Writing algorithms and Scratch were among the techniques used to reinforce computational thinking concepts. Above all, teaching computational thinking via scratch was received positively by students and successfully increased creativity and interest in most of the students according to the feedback.

Brennan & Resnick (2012) developed a new framework to impart the skills of computational thinking to young learners via the scratch. This framework aimed at introducing design-based activities from computational thinking concepts, computational thinking practices and computational thinking perspectives. Scratch was chosen to facilitate the learning of these computational thinking (concepts, practices and perspectives)

Grover et al. (2014) developed a six-week module titled as "Foundations for Advancing Computational Thinking" (FACT) to teach computational thinking to middle school. The six-week module aimed at teaching students the concepts of computational thinking found in scratch. The module was assessed using the existing pre-and post-test instruments applied in Israel middle school students including quizzes and Scratch assignments. Results from the assessment were compared with the results of middle school students from Israel national exams and tests. The main difference of the performance was on one question which had 10 blanks to fill in the Scratch script. This question required the highest level of thinking skills which already US students had from their module which focused on a deeper understanding of concepts and practices" in tracing existing code and reading/writing pseudo-code".

Webb & Rosson (2013) introduces scaffolded examples for teaching computational thinking concepts to middle school girls. Scratch was used to foster the understanding of computational thinking concepts after the trial of other tools such as Alice and Lego Mindstorms which seems to confuse students when transferred from one platform to another. Teaching scaffolded examples from Scratch increased the understanding of computational concepts such as problem-solving skills, abstractions, debugs and vocabularies.

Sound Thinking is another course developed by (Ruthmann, 2010) as an approach for exposing students to computational thinking concepts. The course was designed by the collaboration of two departments, computer science and music. At first, students were introduced to webpages creation using HTML and JavaScripts API where they could embed music files of different formats. Creating music with HTML and JavaScript was not enough to combine the concepts of computational thinking and music. This resulted to move from HTML and Javascript API to scratch which can generate and play sounds using various components from sounds category. After the course, they found out that scratch helped to master the concepts of computational thinking. And, they found out that some of the students developed a strong interest in computational thinking as a means of solving problems in the music industry.

7.4 Entry Visual Programming

Han et al. (2015) brings a mobile-based visual programming designed to facilitate learning of computational thinking to elementary pupils. The environment of this educational tool allowed elementary pupils to cultivate the concepts of computational thinking from basic to an advanced level whereby complexity increased as they progressed. It allowed learners to learn the professional languages such as JavaScript and Python by switching between the mentioned languages from the interface of the tool. In their paper, they described that the tool allowed learners to create program, post and share their own programs created. Entry visual programming aims at: -

- Providing Problem Based Learning (PBL) in which students master the principles of programming naturally via hands-on problems solving. The principles learned can help students to create their own projects.
- Providing interface with natural language, the entry environment allows learners to use their everyday language instead of commands symbols used by computer engineers. The example of the word" Repeat 5" was given and compared to the language of hard programming for example" (for i=0; i<5; i++)". Using" Repeat 5" times is easier for leaners to get the clear concept of programming. Apart from understanding the concepts of computer language, learners also develop logical thinking skills and computational thinking skills which are necessary for solving problems.

Entry visual programming interface consists of three parts namely; learning, making and Looking Around. The learning part includes PBL (Problem Based Learning) using games. Making part as a graphics-based authoring environment allows users to practice programming and implement it. The development of math's and physics in this part uses blocks engine. The last part is Looking Around which acts as a system for users to share, apply and develop source code. Assessing the effectiveness of the tool was done by conducting workshops in 16 elementary school's teachers consisting of 10 males and 6 females. Based on the feedback analyzed, the application was effective to be used as a learning tool for children aged from 11 ~13 although it was not tested to children as the main users.

7.5 Games and game design

Games and game design have been mentioned and applied in different discipline of studies to make learning environment fun and engaging. Because of its "engagement and motivational nature, especially for younger age groups, games have been omnipresent in education since earliest times (Kazimoglu et al., 2014). Researchers on gaming areas provide positive feedback of digital games in facilitating the easy of learning and understanding specifically in the areas of science, technology, engineering, and math (STEM). More specifically games have been defended by several studies as attractive and engaging pedagogical tool to induce computational thinking concepts to learners (Kazimoglu et al., 2014). In secondary and tertiary education, computer video games have become widely used as a vehicle for imparting core knowledge of different subjects and as a tool to attract and retain students in some subjects.

Wu & Richards (2011) argues that to embark the logically and abstract skills to students, digital game-based learning via game design can be used as a pedagogical tool for teaching students to think computationally as they engage in problem-solving tasks. Games are basically designed for different purposes; some games are designed for entertaining and others for educational and entertaining.

Games designed for educational and entertainment acts as a catalyst to foster the development of computational thinking skills. Wu & Richards (2011) tells that digital game-based learning (DGBL) is an educational tool which helps students to think logically and abstractly when designing and practicing problems. Educational games keep a social play in space while fostering the development of computational thinking skills to students. Learning via games engages learners as (Lee et al., 2014) tells that children even at the age of four can be engaged when practicing computational thinking via games. We discuss several games that have been developed to teach computational thinking.

7.6 Program your robot

Kazimoglu et al. (2014) discusses “program your robot” a game which was developed to support the acquisition and access to computational thinking skills. The game was incorporated with computational thinking concepts which are necessary for solving problems in the introduction to programming constructs. In their paper, Kazimoglu et al. (2014) tells that one of the challenges they were facing in their school was how to attract students to choose computer programming subject. Program your robot was designed to address the necessary skills needed in solving problems included in computer programming. Apart from students being able to solve problems in programming subject, the game was also meant to attract student in this subject. Through this game, students were introduced to the fundamental concepts of computational thinking and terminologies used in computer programming instead of only coding.

The fundamental concepts of computational thinking encompassed in the program your robot includes problem-solving, algorithms building, debugging, simulations and socializing. Students developed these skills by solving puzzles simulated in the game via solution algorithms commands. Puzzles were solved by typing some commands which helped a robot to escape from one teleporter to another while complexity of the games increases at each level. The Student used a designed solution algorithms commands which included action command and programming commands. Action command helped to move a robot from one place to another while programming command enhanced the designing of algorithms. Based on the feedback from the evaluation of the game played by 25 students, (Kazimoglu et al., 2014) tells that student cultivated the problem-solving skills which are necessary for learning computer programming construct. Since the game was incorporated with necessary skills related to computational thinking skills. The skills acquired acts as a strategy for solving problems in an introduction to programming construct.

7.7 CTArcade

Lee et al. (2014) discusses an education social game play called Tic-Tac-Toe which was initialized in CTArcade system. The game teaches computational thinking related patterns to children via playing. The system of the game designed had three goals which include: - Integrate tacit knowledge within the game interface, moving from concrete to abstract computational thinking and reduce cognitive load and split attention. From the user interface, the CTArcade system is integrated with computational thinking patterns such as templates, virtual characters, suggestions, feedback and visual analyzing tool. Children cultivate computational thinking skills through naturally playing Tic-Tac-Toe game encompassed with scaffolded learning activities. The game was evaluated by conducting a study involved 18 participants 53% female aged from 10-15 with an average age of 11 years old. In their study, they found out that teaching using CTArcade environment helps children in articulating algorithms thinking skills through playing Tic-Tac-Toe naturally

7.8 RaBit EscApe

In their paper Apostolellis et al. (2014) describe RaBit EscApe game, a tangible wooden piece's board game designed to challenge children aged from 6-10 to understand the computational thinking. The game contributes to the emotions, social and physical development in children when playing. Children cultivate the skills of computational thinking by helping the rabbit to escape from the fierce apes. Helping the rabbit to escape is done by putting the bits together in a predefined path. The skills integrated into the game were inherited from International Society for

Technology in Education (ISTE) operations definitions which have been explained in our core concepts of computational thinking.
The evaluation of the game was done by the formal study of students aged from 8-10 years old. The game was believed to have a low tech, but it contributes to the interactions, social physical development and engaging for teaching computational thinking.

7.9 AgentSheets

Reppening et al. (2010) present checklist requirements tools for adapting computational thinking into public schools. The checklist contained a combination of computational thinking tools curriculum which comprised of "computational thinking pattern inventory, authoring tools, and teacher training". Their aim was to build a scalable game using AgentSheets tool to increase motivation, engagement and interest in computer science subjects to middle school students. In their paper, they claimed that to systematic impact a computational thinking tool in K-12, then it should fulfill the following conditions

- Easy to pick
- Availability of sophisticated techniques
- Scaffolds Flow (stepping from one level to another should be sophisticated)
- Enables transfer (should enable students to use the skills into other scenarios)
- Supports equity (should be accessible and motivate all gender and ethnicity without boundaries)
- Systematic and sustainable (the tool should support all teachers and all students)

AgentSheets was used as an environment for students to design their games since it seemed to have all the mentioned requirements.

In their paper Sengupta et al. (2013) describes a theoretical framework for teaching computational thinking concepts and simulations models to middle school science students. They identified and synergized the importance of learning programming and computational models for biology and physics students. Agent-based computational tools and visual programming environment where suggested among the tools that simplify the learning of programming subject. Teaching via Logo was described to facilitate the learning of models and computational concepts such as abstractions, loops, and recursion. While teaching programming using AgentSheets, Scratch, Alice and StarLogo made programming skills more accessible even for non-computer science students. Learning-by-design progression was a captured framework developed to include Computational Thinking in Simulation and Model building. This environment allowed students to cultivate computational thinking skills while learning to program via agent-based and visual programming tools.

7.10 Game Computational Sophistication

Werner et al. (2014) proposes a new framework called "Game Computational Sophistication" for analysing how computer game programming enhances the development of computational thinking skills in children. They conducted a workshop were 231 games developed by students using Alice environment. The games developed were assessed based on the techniques applied. 11 games out of 231 created by students had most of the mechanics that games experts and researchers termed as computational sophistication. Two types of games were selected among the 11 games and these are: - M808 Super Battle Tank game developed which was developed by the pair of boys and Fish Attack was developed by a girl.

7.11 Light-Bot

Gouws & Wentworth (2013) describes a framework built in Light-Bot to plan what computational thinking involves and how to evaluate the activities involved in Light-Bot. The framework built on Light-Bot was based on programming commands which increased the awareness of computational thinking concepts. Students achieved the computational thinking concepts by providing the series of finite set instructions to a robot so that it lights up all blue blocks on a board. The complexity of the blocks increased as student progressed from one level to another. Meaning that, each level required computational thinking skills to accomplish. After evaluation, they found out that, using the Light-Bot game to instruct a robot with programming constructs enhanced the specified computational thinking concepts by 75%.

7.12 Computational Thinking via Education Robot

Education robot is another approach proposed to develop computational thinking skills to students. Student's gains the concepts of computational thinking when engaged with education robotics activities. Engaging in construction-based robotics activities has suggested as an effective approach that enhances the development of powerful skills in technology, engineering, computer programming. And it also supports children in developing fine motor skills and eye coordination when working in teams.

Computational thinking via education robotics is achieved in different approaches. Among the approach mentioned is using specific education robotics incorporated with programming languages. This method encourages students to constructs and control robotics activities using programming commands. A survey done by Bers et al. (2013) indicates that learning through education robots enables students to develop higher order thinking skills or critical thinking skills, teamwork and abilities to solve the complex problem in the areas of mathematics and science. Activities incorporated in education robot transforms students from being passive to active learners. It allows students to create their own meaningful projects through play to learn while learning to play. Students are transformed from being not only users of the technology but being creators of the technology as they discover things and build knowledge by themselves during working with robotics activities. The joy and fun learning environment is not only for students but also for teachers to improve teaching in the classroom.

In their paper Bers et al. (2013) explains that teaching via education robotics do not only help learners to build physical artifacts but it develops the understanding of creating computer programs, algorithms instructions from the point of starting to build the robot to sense, move or respond to certain actions.

7.13 Lego Mindstorms NXT robotics kits

Van & Braun (2014) present Lego Mindstorms NXT robotics kits to teach computational thinking to students lacked mathematical skills and were majoring in engineering and computer science in Montana Tech (University of Montana). The university developed course focused on logical and mathematical skills and problem-solving skills. The course was taught through lectures and labs activities whereby in labs activities Lego Mindstorms NXT robotics kits was used as an illustrator tool of the concepts learned in class. Algorithms, data and variables, decisions, problem solving, functions and sorting were among the topics taught in the lab using robotics kits. Students were assessed with different lab assignments with one assignment called" feral robot". The robot had two conditions back away and attack, where the robot back away actions happens when an intruder interrupts at a certain distance and it attacks if the intruder is closer. In this assignment students learned condition constructors, looping and develops algorithmic logic. The evolution of the course based on pre-test and post-test was done using Whimbey Analytical Skills Inventory showing that, the course increased the analytical thinking skills to students.

Atmatzidou & Demetriadis (2016) presents education robot (ER) learning activities designed to support the development of computational thinking skills to secondary school students. The education robot was incorporated with computational thinking concepts such as constructs-abstractions, generalization, algorithm, modularity and decomposition. The idea of the robot was to investigate the development of the mentioned concepts from students when solving robotics programming problems. Age and gender were assessed to see if it affects the development of computational thinking skills to students. 8 series of trainings were conducted to the 164 Junior and High school students in Thessaloniki area. Students were placed in groups to solve robotics programming problems using an educational robotic kit called Lego

Mindstorms NXT 2.0. The training conducted was divided into an 11th session where each session emphasized on specific concepts of computational thinking and the complexity of the session increased after every session. Meaning that for students to proceed to the next session, acquiring computational thinking skills of the previous sessions through solving robotics task were needed. Teaching via education robotics activities was evaluated and found out that age and gender did not show much difference in understanding computational thinking. However, they suggested that more training is needed to deepen the understanding of computational thinking. Girls also needed much more time in tackling robotics programs than boys.

7.14 Lego WeDo

Pinto-Llorente et al. (2016) present software Lego Education WeDo in natural science subjects. They focused on testing if using Lego WeDo to create 3D models enhances the development of computational thinking. In their papers, they used the existing materials in Lego WeDo developed by MIT Lab. Students were given a task of creating 3D Dancing birds and Spine Smarter using Lego WeDo in which the concepts behind it can be transferred to science, technology, engineering and mathematics. From the feedback on the paper, they found out that using Lego WeDo in natural science promotes the awareness of computational thinking concepts, for example, decision making and solving problems in a logical way. Moreover, the Lego WeDo engaged students in programming.

7.15 TangibleK

Bers et al. (2014) presents TangibleK an educational robotics curriculum to teach computational thinking concepts to kindergarten children. The curriculum developed in the TangibleK Robotics focused on developing activities that were built in the robotic commercial kit and CHERP (Creative Hybrid Environment for Robotics Programming). The TangibleK curriculum included the concepts from computer science particularly in robotics, engineering design process, sequencing and control flow, loops and parameters, sensors, and branches. These concepts together help children to develop skills in computational thinking, robotics and programming. Teaching children in playful environment invites them to become creators of technology, for example, building their own robotics projects like cars that follow the light, elevators that work with touch sensors, and puppets that can play music when learning to play. In TangibleK, kids explore the use of gears, levers, motors, sensors, and programming loops when working with construction-based robotics activities. The author explains that working with robotics is not only about building physical artifacts but also working with computer programming which brings robots "life". For example, creating computer programs or algorithms which give instruction to the robot to make movement, sense and respond to their environment. In their study, they found out that, exposing children in building robotics activities with specific computer programming languages, improves their visual memory and basic numbers, sense, as well as develop problem-solving techniques and language skills

To sum up, teaching children via educational robotics specifically TangibleK program enhances the development of computational thinking skills to appropriate age children. In the paper Bers et al. (2014) concludes that with given appropriate constructive technologies, curriculum, and pedagogical tools, children can engage in computer programming and robotics activities actively.

7.16 App Inventor

App Inventor is a web-based platform invented by MIT Lab in 2009 to democratize and encourage the process of software development for all. The application enables the novice to practice mobile apps development from simple to advance by just drag and drop building blocks (http://codingforyoungpeople.eu/learn-computational-thinking/).

Grove & pea (2013) present the course taught using App Inventor to reinforce programming concepts to students in middle school. In their course entitled as "discourse-intensive" aimed at teaching computational thinking in a more social way, interactively, discussions and questions. With the use of App Inventor, students were introduced to programming concepts which included variables, loops and conditions. From their feedback teaching programming to students in App Inventor develops computational thinking skills especially when they work in collaborative, sociable, interactive environments. Because of their teaching style of discussions and questions, students were able to raise more questions related to the computational thinking. Apart from that the feedback of using App Inventor was neutral when practicing the concepts of computational thinking compared to robotics and game which seems to be more liked by men than women.

Morelli et al. (2011) investigated if App Inventor for Android can be helpful in bringing computational thinking to K-12. They conducted a project by combining two teachers from high schools, two students from noncomputer science courses, one leader from a community and computer science college instructor. The aim of the summer project focused on building mobile Apps for Android smartphones, generating ideas and preparing lesson plans that can be used to bring computational thinking in K-12. Students taught themselves how to create the mobile apps for about four weeks. On their way going, four Apps were created both having some concepts of computational thinking. Even though it was early to conclude the effectiveness of App Inventor in bringing computational thinking to K-12, however, the participants admitted App inventor to be, powerful and easy to use, allow students to focus on problem, provides framework which advances computational thinking skills to students and lastly app inventor was received as strong motivation tool

8 Tools to Assess the Transfer of Computational Thinking Skills

8.1 Introduction

The assessment of adapting computational thinking into practice has been presented by several authors. To ensure that there is successful in delivering the required computational thinking concepts, there is a need for an assessment. Researchers present different methods and tools for assessing the transfer of computational thinking skills in different ways depending on what they want to achieve from what it has been implemented.

Basawapatna et al. (2011) presents AgentSheets to assess the transfer of computational thinking skills in game programming. In his study, he wanted to know what skills students gain when working with game programming. Computational Thinking Pattern (CTP) graph was used to evaluate the transfer of skills in programming game design. Their purpose was to see if students can create science simulation using the skills learned from game designing. They prepared a puzzle quiz as an assessments tool for students and teachers who participated in the game programming class. The puzzle quiz was the assessments tool for assessing the transfer of skills via game experience. Based on their feedback, they found out that students and teachers managed to solve puzzle quiz problems using the skills learned in game designing.

In connection to that Basawapatna et al. (2013) insisted more on his later article about evaluating the existence of computational thinking to students and teachers via project first known as Zones of Proximal Flow. With the use of Computational Thinking Pattern Analysis (CTPA), (Basawapatna et al., 2013) discovered that teachers and students transfer some skills acquired via project first in solving puzzle problems. Creating games help students to acquire programming skills which can help them to create science simulation. According to Basawapatna et al. (2013), science simulation is closely related to computational thinking concepts. Meaning that students cultivate the claimed elements of computational thinking skills such as problem formulation, logic, data abstractions such as models, algorithmic thinking, implementing effective solutions optimally, and transferring the solution to solve a wide variety of problems.

Creative Thinking Exercises is another assessment tool employed to improve learning of computational thinking in computer science courses (Miller et al., 2013). The tool supports computational thinking blended with creative thinking skills which are applicable in STEM and non-STEM subjects. The main target was to "foster the development of Epstein's creative competencies by engaging multiple senses, requiring imaginative thought, presenting challenging problems and combining both individual and group effort" (Miller et al., 2013). Miller et al. (2013) assessed the skills achieved in this tool through analysis and reflection questions (computational thinking test) designed to enhance the development of computational thinking and the applications of these creative computational skills.

Seiter & Foreman (2013) present Progression of Early Computational Thinking (PECT) for assessing the understanding of computational thinking for primary grades (Grades 1 to 6). The tool is broken into three parts:

- Computational thinking concepts
- Computational thinking Design Pattern
- Evidences Variable (EV)

Evidences variables in scratch, uses written blocks of code to measure the levels of student's work if it contains scratch programming blocks or variables termed as categories which included; Looks User Interface Event, Parallelization, Initialize Location and Initialize Look. The aim was to measure the designed and implemented programs that used primary concepts involved in computational thinking. PDV's measures the strategy and models used in implementing the tasks. It uses evidence variables to determine the understanding level of computational thinking concepts.

Lastly, Seiter (2013) measured students work in the general use of computational thinking concepts such as Procedures and Algorithms, Problem Decomposition, Parallelization and Synchronization, Abstraction and Data Representation (Barr et al., 2011; CSTA, 2012). The mentioned set of computational thinking has been mentioned by several authors as the guidelines for enhancing the development of computational thinking skills. However, Seiter & Foreman (2013) did not describe how to assess them in students work developed using scratch environment.

Above all Brennan & Resnick (2012) summarized the three approaches for assessing the development of computational thinking for young people engaged in developing programs using scratch. Scratch itself first develops computational concepts, practices and perspectives to the students. Together the three elements allow students to develop thinking skills in sequences, loops, parallelism, events, conditionals, operators, and data, incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing, expressing, connecting and questioning. To be sure that student develops these necessary skills, the three proposed approaches were used.

- First, analyzing students' projects portfolio. Scratch allows students to create their programs and uploads it in an online community. They used a Scraper tool to analyze if the uploaded projects used all necessary blocks available in Scratch.
- Artifacts-Based Interviews. They did an interview with 31 scratchers aged from (8-17) who participated in creating scratch programs. The most important part was to ask why the certain code was used or why the certain block was used and why to choose some artifacts.
- Selecting design scenarios. In this part, they designed three sets of projects and asked students to select one from each and describe what the selected project does, how could they extend it, how to fix bugs and how to add another feature.

Dr. Scratch is another assessment tool presented for automatically assessing the projects created in scratch visual language (Moreno-leon et al., 2015). The tool aims to first provide feedback to teachers and students plus assigning the score for the project tested. Based on author's arguments, assigning the score and providing feedback for computational thinking created project will help students to improve their programs. Moreno-León et al. (2015) argues that making corrections based on the feedback improves the understanding of computational thinking level

8.2 Conclusion

Assessing the transfer of computational thinking depends on specific discipline or vocabulary such as debugging, code-tracing, problem decomposition and pattern generalization (Grover et al., 2015). The assessment has also some challenges for some teachers who assess the final product for example project created by the student. Measuring the understanding of computational thinking concepts based on final projects created by students might not be an effective way. Because some visual programming tools such as Alice has some predefined code program which allows students to modify them as (Grover et al., 2015) report.

Apart from that, teachers encourage students to work in pairs. Working in groups or pairs might be an effective way for students to collaborate and teach each other. However, in their final projects it may even happen that some of the students may ask for help to their fellow students. This might lead to some students getting a score on their final project which they do not deserve. Hence assessing students final project created may not be an effective way to determine the understanding of computational thinking.

Educators and teachers are encouraged to use multiple assessment methods such as exercises using Scratch code, Scratch programming assignments and a final project of the student’s work. Visual programming like the scratch is one of the mentioned languages for easily learning to program. It provides drag and drop tools which make the job of designing games quickly. Using multiple assessment methods is encouraged. More tools and techniques for assessing the transfer of computational thinking are needed (Grover et al., 2015). For example, assessing the created project developed in an environment such as scratch or Alice requires multiple assessment tools. And again, assessing the programs designed by students from different age needs a special tool. For example, measuring the program designed by someone aged from 9-10 will require a different tool with someone aged from 15-19. Their understanding level and the use of those elements and their selection of designed patterns are expected to be different. More research and further studies are needed in assessing the understating of computational thinking and its related concepts in the whole age group

9 Discussion and Conclusion

9.1 Discussion

Computational thinking has been defined as the thoughts processes involved in solving problems computationally. These processes include; algorithms design, pattern recognition, decomposition and abstractions. To impart the skills from these processes to kids, different approaches have been suggested including tinkering, creating, debugging, persevering and collaborating. The practice of these approaches can be achieved via various technological tools such as scratch visual programming language, Alice, robotics activities, well-designed hands-on activities and real-life projects. In this study, we have designed a simple platform with resources focusing on bringing computational thinking skills into classes via Scratch. Maze game originally created by attechedu.com for grade 3rd to 4th was adopted as our starting point of getting kids into Scratch programming language. The game is not that challenging, however, the concepts incorporated into the game meets the target of inducing computational thinking concepts to kids. The use of various blocks in this game is enough for kids to understand various concepts of computational thinking via the scratch. The game was tested for kids of grade 3rd to 4th in the UK education system. Even though it was tested for grade 3rd to 4th in the UK education system, however, for a Tanzanian education system, it can be adapted to kids from grade 5th to 7th. To measure the concepts incorporated into the game, we re-created the game (https://scratch.mit.edu/projects/203766342/) and modified it following similar rules from the origin game. The aim was to identify how kids with different backgrounds can follow along to get into game design and coding in a scratch environment. The re-created game was not tested or evaluated for intended audiences; however, the concepts incorporated follows into computational thinking skills in two ways. For teachers, it covers most of the concepts and blocks available in Scratch. The concepts can also be applied to different visual programming languages. For elementary kids, it will build some computational thinking concepts including, conditional statement, looping, operators, coordinates and many more. The concepts learned can also be

transferred in solving some other problems using different visual programming language.
Scratch is among the active learning tools approached to bring computational thinking into classes. It was selected as a starting point in the platform created, for two major reasons. First, it is free and secondly, it has an offline and online mode for users to create their games, animation or stories with or without internet. With the offline or online creating mode, brings more value to some areas with slow or no connection to the internet.

At first, the platform will have computational thinking related resources and scratch guidelines. This is to let interested teachers or individuals start teaching computational thinking in classes using Scratch. The platform is also for individuals and education stakeholders who want to contribute to creating the resources necessary for teaching computational thinking skills to kids. Currently, Scratch has more extended blocks which need extended physical devices like Arduino Starter Kit. For this study, scratch in Arduino was not covered.

Practically, bringing computational thinking into classes has been trending in many countries including Finland. Apart from the national new curriculum of adapting computational thinking in compulsory education which aims to improve mathematics skills and algorithms skills, Individuals, education stakeholders and organization has already started teaching computational thinking within Finland and outside Finland.

Mehackit Company is the startup company teaching coding skills to improve mathematics skills to some high schools in Finland. Their curriculum is focusing on using available tools to teach high school students how to code. For example, one of their approaches is using Sonic Pi to teach kids in creating their own music. Processing language, robotics activities, Scratch in Arduino are among the approaches they are using to teach students.

Apart from that, individual groups of Finland have also taken a step farther to teach kids not only know how to code but to use coding skills to solve mathematics

problems. Art & Craft School of Robotics at Children's Cultural Center Little Aurora located in Espoo organizes workshops during summer and lessons every week to kids aged from 9-10 years old. This is happening in almost all the cities in Finland. Their curriculum focuses on adapting coding skills to kids using a graphical programming language such as Scratch and Electronic Craft activities like building robotic using Arduino kit. During the visit to Art & Craft School of Robotics, their first lesson was teaching kids how to create animations from scratch. After the scratch lesson, they moved to robotics activities and to 3D printer. In a scratch lesson, several questions were asked by kids and many kids after sometimes between the lessons were at least comfortable to move scratch blocks to working area. The attention and quietness for kids were much more during scratch lessons compared to other lessons where they started moving around and testing many learning devices which were inside the class including 3D printer and toys. Being comfortable with scratch blocks for Tanzanian kids might not be the same for many factors including the knowledge of using computer itself is limited. Most of the teachers in Art & Craft School of Robotics are from arts and media background, meaning that bringing computational thinking via the scratch or any technological forces for adapting computational thinking into classes do not necessarily need a solid background of programming or computer science. However, this might not be the same for some elementary teachers in Tanzania as most of them they are still not comfortable or motivated with anything that draws some concepts from computer science.
Computer-related courses have not yet been declared as a compulsory subject in elementary school and in secondary school’s curriculum. Although, the study target to bring computational thinking skills into education curriculum, computer basic skills have its own role in achieving the skills. Speaking of Scratch that the story of for example a game or quiz can start on paper, however animating and debugging the codes requires some basic computer skills and computer as a device.

The platform developed aims to help teachers, individuals, and elementary school’s owners with enthusiasm of bringing computational thinking skills into classes to get started. Scratch is the option to get started with computational thinking as it gives chances to even non-computer professions to get started with computational thinking concepts without thinking of the cost, installation and internet connection. Though, even with the easy use of Scratch, testing how easy it is for intended users is the core important part of this study to get the informative report. Speaking of countries like Tanzania where it has teachers with the different educational background and ages, the report might have different results considering their level for example education, age, being ready and motivated. Training these teachers with different background will have a different story and the time to get motivated and start bringing computational thinking into classes. To the elementary pupils from rural areas where they might have never touched or seen computers and the elementary pupils from urban areas where they might have seen or used computers but never practiced scratch, their story will always be different. However, getting the real story covering both angles involves money and time to invest.

9.2 Figures

Figure 1 shows a web-based platform for gathering computational thinking resources.

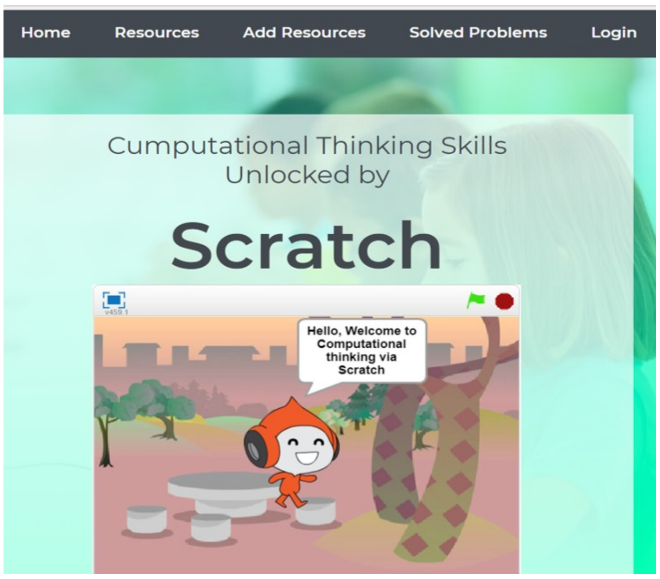
Home
Resources
Add Resources
Solved Problems
Login
Cumputational Thinking Skills
Unlocked by
Scratch
Hello, Welcome to Computational thinking via Scratch

Figure 1. A Web-Based Platform

The landing page or home page imbedded with the Scratch maze game where by its code runs from (https://scratch.mit.edu/projects/203766342/). The game has two parts. Part 1, introduces the learners the basic concepts of computational thinking and scratch through animating the character. The part 2, is playing the game which has two levels. Playing the games is not the only target for but instead, we want kids to play the game then think of the stories behind the game to create their own game.

9.3 Conclusion

The study aimed to inform teachers, education stakeholders, and individuals who wish to bring computational thinking in classes. As mentioned from the introduction part, that computational thinking is a vital skill for everyone not just for computer scientists, making it happen in classes can be a challenge if no clear guidelines are provided. Many studies have discussed the importance of it and the use of different pedagogical tools such as game design, robotics activities, programming languages and computational tools. The mentioned pedagogical tools and many others are believed to enhance the development of computational thinking to students. However, assessing the transfer of the skills gained via the mentioned pedagogical tools may seem like a challenge if it does no go beyond traditional classes especially in Tanzanian context. To the elementary teachers and pupils with different backgrounds, deep training, going beyond traditional classes, real-life projects, and motivation for both groups is required.

More training and motivation of computer related courses at the national level training center is required for elementary teachers who seem not even willing to teach any computer science subject with the reason that, they do not have foundation skills of computer-related subjects. We created a platform to get started with computational thinking skills using scratch as a starting point. In the future not only, the Scratch will be the only method but other pedagogical tools such Alice visual programming language, extended blocks of scratch in Arduino Starter Kit and Robotics activities. Apart from these, Google for education has created more resources to get started with computational thinking.

References

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Ealy, J. P. (2010). Computational thinking for the sciences: a three-day workshop forhigh school science teachers. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). ACM.

Apostolellis, P., Stewart, M., Frisina, C., & Kafura, D. (2014) RaBit EscApe: A board game for computational thinking. In *Proceedings of the 2014 conference on Interaction design and children* (pp. 349-352). ACM.

Ater-Kranov, A., Bryant, R., Orr, G., Wallace, S., & Zhang, M. (2010). Developing a community definition and teaching modules for computational thinking: accomplishments and challenges. In *Proceedings of the 2010 ACM conference on Information technology education* (pp. 143-148). ACM.

Arraki, K., Blair, K., Bürgert, T., Greenling, J., Haebe, J., Lee, G., & Hug, S. (2014). DISSECT: An experiment in infusing computational thinking in K-12 science curricula. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1-9). IEEE.

Astrachan, O., Hambrusch, S., Peckham, J., & Settle, A. (2009, March). The present and future of computational thinking. In *ACM SIGCSE Bulletin* (Vol. 41(1), pp. 549-550). ACM.

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*, (Vol 75, pp 661-670).

Barr, D., Harrison, J., and Conery, L. (2011) Computational Thinking: A Digital Age Skill For Everyone. ISTE Learning and Leading. (Vol. 38(6), pp. 20-23).

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *Acm Inroads*, (Vol. *2*(1), 48-54).

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school.

In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 224-228). ACM.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 245-250). ACM.

Basawapatna, A. R., Repenning, A., Koh, K. H., & Nickerson, H. (2013). The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 67-74). ACM.

Bers, M. U. (2010). The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice*. (Vol 12(2), pp2).

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*. (Vol. *72*, pp 145-157).

Blum, L., & Cortina, T. J. (2007, March). CS4HS: An outreach program for high school CS teachers. In *ACM SIGCSE Bulletin* (Vol. 39(1) pp. 19-23). ACM.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).

Cortina, T. J. (2007, March). An introduction to computer science for non-majors using principles of computation. In *ACM SIGCSE Bulletin* (Vol. 39(1), pp. 218-222). ACM.

Cole, E. C. (2015). On Pre-Requisite Skills for Universal Computational Thinking Education. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 253-254). ACM.

Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., ... & Kaza, S. (2011). A model for piloting pathways for computational thinking in a general education curriculum. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 257-262). ACM.

Dukeman, A., Caglar, F., Shekhar, S., Kinnebrew, J., Biswas, G., Fisher, D., & Gokhale, A. (2013). Teaching computational thinking skills in c3stem with traffic simulation. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data* (pp. 350-357).

Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39-48). ACM.

Falkner, K., Vivian, R., & Falkner, N. (2015). Teaching Computational Thinking in K-6: The CSER Digital Technologies MOOC. In *Proceedings of the 17th Australasian Computing Education Conference (ACE 2015)* (Vol. 27, p. 30).

Guzdial, M. (2008). Education Paving the way for computational thinking. *Communications of the ACM*, (Vol 51(8), pp 25-27).

García-Peñalvo, F. J., & Cruz-Benito, J. (2016). Computational thinking in pre-university education. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 13-17). ACM.

Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015). Computing education in K-12 schools: A review of the literature. In *Global Engineering Education Conference (EDUCON), 2015 IEEE* (pp. 543-551). IEEE.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). First year student performance in a test for computational thinking. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference* (pp. 271-277). ACM.

Gross, S., Kim, M., Schlosser, J., Lluch, D., Mohtadi, C., & Schneider, D. (2014). Fostering computational thinking in engineering education: Challenges, examples, and best practices. In *Global Engineering Education Conference (EDUCON), 2014 IEEE* (pp. 450-459). IEEE.

Grover S., Pea R. (2013). Computational thinking in K-12: A review of the state of the field. Educational Researcher, (Vol. 42 (2), pp. 59–69).

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 *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 723-728). ACM.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 57-62). ACM.

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

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, (Vol. 41(1), pp. 183-187)

Han, A., Kim, J., & Wohn, K. (2015). Entry: visual programming to enhance children's computational thinking. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers* (pp. 73-76). ACM.

Holbert, N. R., & Wilensky, U. (2011). Racing games for exploring kinematics: A computational thinking approach. In *Proceedings of the 7th international conference on Games+ Learning+ Society Conference* (pp. 109-118). ETC Press

Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 223-227). ACM.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, (Vol. 82, pp. 263-279).

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *future directions in computer science education summit meeting, Orlando, FL.*

Kafura, D., & Tatar, D. (2011). Initial experience with a computational thinking course for computer science students. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 251-256). ACM.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2010). Developing a game model for computational thinking and learning traditional programming through game-play. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, (Vol. 9, pp. 522-531).

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, (Vol. 33, pp. 1-26).

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

Lee, T. Y., Mauriello, M. L., Ingraham, J., Sopan, A., Ahn, J., & Bederson, B. B. (2012). CTArcade: learning computational thinking while training virtual characters through game play. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems* (pp. 2309-2314). ACM.

Lee, T. Y., Mauriello, M. L., Ahn, J., & Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction*, (Vol. 2(1), pp. 26-33).

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*, (Vol 41, 51-61).

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

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 2014 on innovation & technology in computer science education conference* (pp. 1-29). ACM.

Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., Ramsay, S., & Hazley, M. P. (2013, October). Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses. In *2013 IEEE Frontiers in Education Conference (FIE)* (pp. 1426-1432). IEEE.

Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., & Hazley, M. P. (2014). Integrating computational and creative thinking to improve learning and performance in CS1. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 475-480). ACM.

Morelli, R., De Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E., & Uche, C. (2011). Can android app inventor bring computational thinking to k-12. In *Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11)* (pp. 1-6).

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, (Vol. (46), pp. 1-23).

Peel, A., Fulton, J., & Pontelli, E. (2015, October). DISSECT: An experiment in infusing computational thinking in a sixth-grade classroom. In *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE* (pp. 1-8). IEEE.

Pinto-Llorente, A. M., Martín, S. C., González, M. C., & García-Peñalvo, F. J. (2016). Developing computational thinking via the visual programming tool: lego education WeDo. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 45-50). ACM.

Qin, H. (2009, March). Teaching computational thinking through bioinformatics to biology students. In *ACM SIGCSE Bulletin* (Vol. 41(1), pp. 188-191). ACM.

Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, (Vol. *25*(5), pp. 66-71).

Repenning, A., Webb, D., & Ioannidou, A. (2010). 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* (pp. 265-269). ACM.

Rubinstein, A., & Chor, B. (2014). Computational thinking in life science education. *PLoS computational biology*, *10*(11), e1003897.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters II, C. (2010). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 351-355). ACM.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59-66). ACM.

Seoane-Pardo, A. M. (2016). Computational thinking beyond STEM: an introduction to moral machines and programming decision making in ethics classroom. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 37-44). ACM.

Segredo, E., Miranda, G., León, C., & Santos, A. (2016). Developing Computational Thinking Abilities Instead of Digital Literacy in Primary and Secondary School Students. In *Smart Education and e-Learning* (pp. 235-245). Springer International Publishing.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, (Vol. 18(2), pp. 351-380).

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

Settle, A., Goldberg, D. S., & Barr, V. (2013, July). Beyond computer science: computational thinking across disciplines. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 311-312). ACM.

Schulte, C., Hornung, M., Sentance, S., Dagiene, V., Jevsikova, T., Thota, N., & Peters, A. K. (2012, November). Computer science at school/CS teacher education: Koli working-group report on CS at school. In *Proceedings of the*

12th Koli Calling International Conference on Computing Education Research (pp. 29-38). ACM.

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* (pp. 133-138). ACM.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends*, 1-4.

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, (Vol. 20(4), pp. 715-728).

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

Webb, M., Fluck, A., Cox, M., Angeli-Valanides, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2015). Curriculum-Advancing understanding of the roles of computer science/informatics in the curriculum.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining Computational Thinking for Science, Technology, Engineering, and Math.

Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. In *Learning, education and games* (pp. 37-53). ETC Press.

Werner, L., Denner, J., & Campe, S. (2015). Children programming games: a strategy for measuring computational learning. *ACM Transactions on Computing Education (TOCE)*, (Vol. 14(4), pp. 24).

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, (Vol. 49(3), pp33-35)

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

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)*, (Vol. 11(2), 9).

Wu, M. L., & Richards, K. (2011). Facilitating computational thinking through game design. In *International Conference on Technologies for E-Learning and Digital Entertainment (pp.220-227)*