A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions

Friday Joseph Agbo
School of Computing
University of Eastern Finland
Joensuu, Finland
fridaya@uef.fi

Solomon Sunday Oyelere
School of Computing
University of Eastern Finland
Joensuu, Finland
solomon.oyelere@uef.fi

Jarkko Suhonen
School of Computing
University of Eastern Finland
Joensuu, Finland
jarkko.suhonen@uef.fi

Sunday Adewumi
Department of Computer Science
Federal University Lokoja
Lokoja, Kogi State, Nigeria
sunday.adewumi@fulokoja.edu.ng

ABSTRACT

This study examined how computational thinking (CT) has been used to teach problem-solving skills and programming education in the recent past. This study specifically (i) identified articles that discussed CT approach for programming education at higher education institutions (HEIs), (ii) classified the different CT approaches and tools employed for programming education at HEIs, (iii) synthesised and discussed results that are reported by relevant studies that utilized CT for teaching programming at HEIs. A systematic literature review methodology was adopted in this study. Out of 161 articles retrieved, 33 of them that met the inclusion criteria were reviewed. Our study revealed that the use of CT at HEIs for programming education began in 2010; many studies did not specify the context of use, but the use of CT is found to be gaining grounds in many contexts, especially the developed countries; course design approach was mostly employed by educators to introduce CT at HEIs for programming education. Furthermore, this study pointed out how CT approach can be explored for designing a smart learning environment to support students in learning computer programming.

CCS CONCEPTS

• **Social and professional topics → computational thinking**

KEYWORDS

computational thinking, problem-solving, programming, higher education, undergraduates, smart learning

ACM Reference format:
Friday Joseph Agbo, Solomon Sunday Oyelere, Jarkko Suhonen and Sunday Adewumi. 2019. A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. In 19th Koli Calling International Conference on Computing Education Research *(Koli Calling '19), November 21-24, 2019, Koli, Finland. ACM, New York, NY, USA, 10 pages.*
https://doi.org/10.1145/3364510.3364521https://doi.org/10.1145/1234567890

Koli Calling '19, November 21–24, 2019, Koli, Finland

ACM ISBN 978-1-4503-7715-7/19/11…$15.00
https://doi.org/10.1145/3364510.3364521https://doi.org/10.1145/1234567890

1 INTRODUCTION

Computational thinking (CT) has the capacity to improve programming education by building the problem-solving and algorithmic skills of learners [1]. CT involves having thinking skills, i.e., "ways of thinking and practicing that are sharpened and honed through practice"—Denning and Tedre [2] (pp.6). Among other skills, CT are envisioned to make the developing of programming and problem-solving skills a flexible experience. Learning of computer programming remains one of the challenges facing students and educators [3] [4] [5]. While CT is the fundamental approach towards understanding, abstracting and modelling a problem with the intention of providing a solution, computer programming is the actual transcription of the abstraction into a computer understandable language. The stages of computer programming involve writing, testing, debugging and running a set of codes using different programming languages [6] [7]. These stages can be tasking and complex, especially for novices. Researchers [6] [8] [9] have shown that often, programming is relatively considered difficult among other science-related courses, and building the skill takes time and commitment [3]. However, efforts to ease the task of learning to develop one's problem-solving skills and programming education exist [4]. Some authors applied approaches such as visualization, games, puzzles, and computational thinking to motivate students and increase interactions between learners and educators [8] [9] [10]. For instance, Oyelere et al. [9], Li [11] designed a system/approach for teaching computing education with puzzles to motivate students' learning.

Computational thinking was made popular by Wing [12] in 2006, stating that CT "involves solving problems, designing systems, and understanding human behaviours, by drawing on the concepts fundamental to computer science" [12] (pp.33). Since then, varied definitions of CT have been provided.

In this paper, we discussed the various definitions of CT by authors from different perspectives and presented a definition that is focused on using CT for problem-solving as a fundamental step towards teaching and learning programming education. The study also investigated how CT is being utilized for teaching and learning

at higher education institutions (HEI). Furthermore, this study investigates the use of CT approach by educators and scholars in teaching programming education at the HEIs. As far as we know, there is no systematic literature review of CT as a programming teaching approach conducted within the premise of HEIs. Hence, this study aims to investigate different CT approaches used for teaching and learning programming at HEIs to explore the opportunity for ingraining CT feature into the design of a smart learning environment (SLE). This study is important because it helps to give direction to the design of a SLE to support programming education based on CT approach. Besides, CT skills are fundamental knowledge for students at the HEIs [1]. Furthermore, this study will provide insight to researchers and developers of contemporary learning environment, specifically in the area of computing education, on ways CT can enhance the development of a SLE. To achieve the aim of this paper, the following objectives were outlined for this study: (i) identify articles that discussed CT approach for programming education at HEIs, (ii) classify the different CT approaches and tools used for programming education at HEIs, (iii) discuss results that are reported by scholars that used CT approaches for programming education at HEIs. Based on the objectives, this study seeks to provide answers to the following research questions:

RQ1. How has the use of CT been explored for teaching programming in HEIs?

RQ2. What are the ways CT approach has been used to teach programming education in HEIs?

RQ3. How has the use of CT approach for teaching programming impacted students learning experience?

2 BACKGROUND

2.1 Definition of CT

According to Denning and Tedre [2] (pp.10), the historical perspective of computational thinking exists even before the emergence of electronic computers. However, since the discussion of CT by Wing [12]—centred on the fundamental knowledge for computer programming, it has fast become popular in the computer science domain [11, 1]. Although there exists no consensus definition of CT, and as presented by Velázquez [13], most of the definitions are ambiguous. For instance, Barr and Stephenson [14] provided what they called “operational definition” of CT as a problem-solving process that includes a set of characteristics. These charactersistices are: i) formulating a problem in a way that one can use a computer or other tools to solve them; ii) organising and analysing the data logically; iii) abstracting the data in form of models and simulations; iv) automating the solution through algorithmic steps; v) identifying, analysing, and implementing a possible solution with the goal of achieving the most efficient combination of steps and resources; vi) generalizing and applying the problem-solving steps to varieties of problems in other areas of endeavours. In addition, Mannila et al. [15] have defined CT as “a term meant to encompass a set of concepts and thought processes that aid in formulating problems and their solutions in different fields in a way that could involve computers.” [15] (pp. 1). One of the points stressed by many authors is the fact that the concept of CT is a fundamental background in computer science [16], which deals with scientific thinking in the technological age. Also, in many definitions of CT, the word ‘problem’ is mostly the focus. For instance, it is common to encounter words like problem-solving, problem decomposition, and problem abstraction. From these various definitions of CT, some keywords emerge; *problem-solving, systems design,* and *human behaviour* [12, 17, 18]. These keywords are relevant when discussing the fundamental concepts of computer science.

In trying to solve the very many human problems, CT promises to be a good approach by applying human-computer interaction and relationship perspectives [11]. According to Wing [12], CT is beyond computer programming or the ability to write codes, CT involves more thinking about multiple levels of abstractions. Problem decomposition and abstraction is the first approach towards problem-solving; next is the algorithm design, which eventually leads to computer programming, and then, a concrete solution (See Figure 1).

One of the ways to help the students develop programming knowledge is to build problem-solving skills. Problem-solving skills involved a critical thinking process about the understanding of the problem on which to develop concrete steps towards its solution. This skill is needed to develop students programming experience since an explicit path to the solution can be defined. The application of CT to problem-solving is not limited to computer science alone, but every field of science and indeed, all human endeavours. However, this paper discussed CT as a fundamental step towards building problem-solving skills that can aid programming education using the CT approach to achieve this objective.

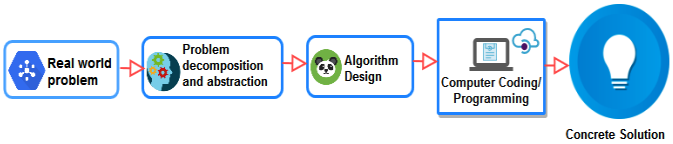

Figure 1: Definition of computational thinking and problem-solving steps

2.2 CT for problem-solving and programming education

Every day, humans are faced with real-world problems that require some thinking and logic in order to solve them. People apply the concept of CT consciously or unconsciously. CT’s concept may not be necessarily complex but attempts to solve every problem by defining some simple computational steps. These steps, however, can involve abstraction of the problems, creating models, designing algorithms, and verifying results, and ascertaining the viability of the solution [11]. Since the discussion on CT emerged, many researches demonstrate the practical application in problem-solving, algorithm, system modelling, and automation. Most of these researches presented results of CT application in teaching K-12 and STEM [14, 19]. Although studies that investigated the approaches for integrating CT in teaching specific disciplines exists [20], there seems to be limited attempt to study the application and

impact of CT at HEI to teach programming education and problem-solving. For example, Araujo et al., [21] studied the techniques and approaches used by researchers and teachers to assess the development of CT abilities in students. Their work explored previously proposed approaches to promote CT and classified them based on their capability to assess CT in education. Similarly, Moreno-León [22] reviewed recent investigations that study CT from different perspectives. The study further classified technologies to learn and teach CT into five, which includes i) unplugged activities; ii) arrow-based visual environment; iii) block-based visual environment; iv) textual programming language; v) connected with the physical world. In an attempt to also investigate how CT in schools is assessed, [22] highlighted three main approaches used by educators—CT-test, Bebras, and Dr. Scratch. Another recent study by [23] reviewed literature on CT and reported on the diversity in definitions, interventions, assessments, and models. In computing education, for example, Repenning et al. [24] employed the concept of designing 3D computer graphics as a CT approach in the teaching of computer science education. This approach, beyond motivating the learners, presents the visualization of computer programming to aid learners' understanding.

Similarly, Philip et al. [25] studied the relationship between CT, problem-solving and programming education with students of HEI. They tried to present a pragmatic approach towards developing CT and problem-solving skills for novices of computer programming. This study [25] showed that CT is fundamental in introductory programming; it demonstrates that in problem-solving, defining the abstraction and modelling precedes both programming logic and constructs of the solution. The two levels of problem-solving are problem analysis and design, and implementation (see Figure 1).

The role of programming in computing education, according to [26] has been a major debate. Nowadays, programming education has grown into a discipline and further into a professional career. The traditional ways of learning programming using languages such as Java and C++ can be challenging and boring. However, using visualization technique in supporting programming education can improve computational thinking and problem-solving skills of learners [19]. For example, when trying to teach freshmen the basics of programming education, it is better to use visual programming language such as Scratch rather than the traditional approach, which may be difficult to understand [25]. Visual programming has been researched recently to aid the students' understanding of computer programming [19] and simplify their learning experience. A block programming concept is used to present visualization of the traditional programming syntaxes, so that the students do not necessarily have to understand its syntax and semantics that may appear abstract. A practical example in Figure 2a-b depicts a simple program for the number system.

A study to investigate the use of CT and its adoption at the HEI was conducted by Czerkawski and Lyman [27]. Their study revealed that efforts are being made to design and introduce CT courses at the HEI and to implement CT in computer science curricula. All these studies play a significant role in advancing the research contribution in CT and provide more insight into the level of efforts being made by scholars regarding the current situation. However, they do not provide knowledge regarding the state-of-the-art tools and interventions of CT utilized at HEIs and their impact on students at HEIs.

Besides, our study intends to explore the opportunity of ingraining CT features into the design of SLE for programming education, since CT is believed to be a foundation for teaching and learning of programming by novices and freshmen.

```
let i = 0
i = 0
for (let i = 0; i <= 10; i++) {
    if (i % 2 == 0) {
        basic.showString("Even Number!")
    } else {
        basic.showString("Odd Number!")
    }
}
```

Figure 2a. Traditional coding in JavaScript

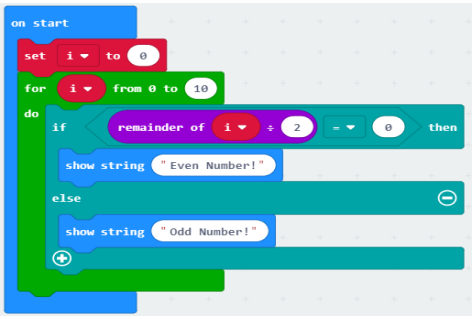

Figure 2b. Visual block programming in micro:bit

3 RESEARCH METHODOLOGY

3.1 Procedure

We began by collecting relevant literature on CT from selected multidisciplinary databases following the systematic review approach [23]. During the data collection, we considered only articles that were peer-reviewed and published in scholarly journals and conferences. Our search obtained relevant articles on studies regarding computational thinking, problem-solving, and programming education at the HEIs. Thus, our search keywords were *computational thinking*, *problem-solving*, *programming*, *computer science*, and *undergraduate*. The search was conducted in five databases—IEEE Xplore, ACM, ScienceDirect, and Springer Link. The decision to select these databases for our search was because they warehouse articles that are published in the field of education, STEM (science technology engineering and mathematics), and computer science [28].

In each of the databases, the same keywords were used to conduct an advanced search. The search was targeted at metadata aspects of the literature. The metadata for an article contains important information about that article, for example, the title of an article, keywords, and abstracts are part of metadata. An example of the combination of search string is

((((computational thinking) AND problem solving) AND programming) OR computer science). These strings were modified when necessary to suit the search pattern of each databases. The results returned by the search were subjected to list of criteria in order to select only articles that are relevant to our study.

3.2 Paper screening

Screening of the paper was conducted by applying some sets of inclusion and exclusion criteria. The process for selection started by scanning through the metadata before downloading the document for further screening. For instance, the title, keywords and abstract were first skimmed through to ensure that the article is focused on our study. The inclusion and exclusion criteria are presented in Table 1 while the results returned after running the queries in each database are shown in Table 2.

Table 1. Inclusion and exclusion criteria

Inclusion criteria	*Exclusion criteria*
Articles that focuses on computational thniking, problem-solving, and programming education	Articles that do not focus on the keywords or not written in English Language are excluded
Articles that are published in a peer-reviewed journals or conferences	Materials that are not peer-reviewed (audio/video files, PPT, e.t.c) are left out
Articles that either presented a concrete programming artefacts, evaluated a solution for programming or design a study to explore CT	Theoretical and conceptual studies are removed

Table 2 Summary of search results

Database	*Search Results*
ACM	47
IEEE Xplore	64
ScienceDirect	22
SpringerLink	28
Total	161

The next stage of the selection process considered only studies that are peer-reviewed and published in a journal or conferences. The last major selection stage examined whether the paper utilized a tool, artefact or designed a course to apply in a real-life teaching/learning scenario and presented empirical evaluation of the study outcome.

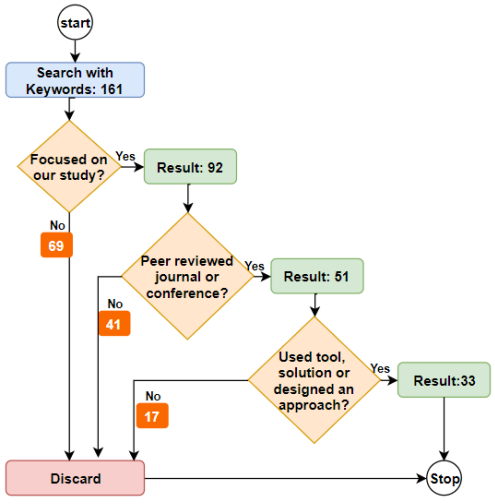

Figure 3: Processes for selecting relevant studies

A total of 161 data were collected from the search conducted. As depicted in Figure 3, most of the studies do not meet the conditions for inclusion after all the papers collected were read and criteria were applied. Studies that were not conducted in the context of HEIs were dropped.

Besides, some of the articles from the databases only showed the abstract, which is not sufficient for our review; also, some of the articles could not be accessed, while some are not written in the English language. At the final stage of the selection process, 33 studies were selected as relevant to our study.

4 RESULTS

This section presents the findings by first examining the trends in the use of CT for programming education at the higher education institutions (HEIs). Further, we investigate the different CT approaches and tools used for programming education at HEIs.

4.1 CT for programming education in HEIs

4.1.1 Studies distribution by year of publication. Figure 4 presents the studies distribution according to the year of publication. Although we do not delimit the study by defining date range, but data gathered revealed the trends when the topic "Computational Thinking" began to gain scholar's attention. Regarding the use of CT for programming education at HEIs, we found 3 studies that were conducted in 2010. More studies were found in 2017 and 2018.

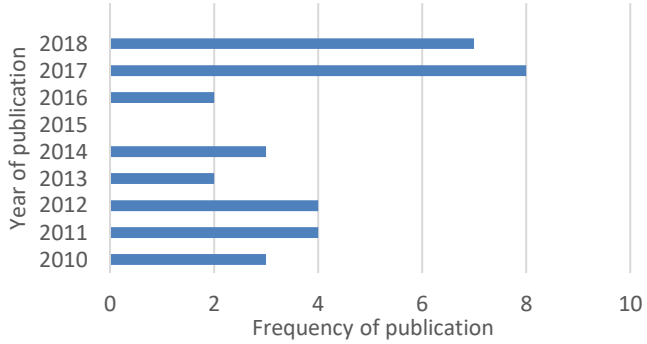

Figure 4: Distribution by year of publication

4.1.2 Studies distribution by outlets of publication. Our study was focused on collecting data from two major outlets (i.e., conferences and journals) as explained in section 2.1. Table 3 delineates the distribution of studies by the publication outlet.

Table 3 Collected studies distribution by outlets

Publication Outlet	*Frequency*
Journals	10
Conferences	23

4.1.3. Context of study. Table 4 shows the context in terms of country where each study was conducted. Although many of the study do not explicitly mentioned the country where it was conducted, our study shows that United States had more frequency among the countries.

Table 4 Collected studies distribution by context

Country	Canada	India	United States	China	Ireland	Japan	Unspecified
Frequency	1	1	6	1	1	1	22

Table 5 Analysis of studies that use CT approach in teaching programming education at HEIs

Paper title	Author(s)	Teaching approach & study technique	Learners	Context	Settings	Study impact/evaluation
Connecting undergraduate programs to high school students: teacher workshops on computational thinking and computer science	Morreale [29]	Workshop and short courses	Teachers of computer science at high schools	US high school	Formal	Pre- and post- workshop surveys show that these teachers' perceptions were positively affected regarding careers in computer science
Using video to explore programming thinking among undergraduate students	Wellington and Ward [30]	Recorded video furring programming activity	Computer Science undergraduate students	Context is not mentioned	Formal	The study shows a cognitive difference between freshmen and their different levels of experience in problem-solving and code reading
Promoting Computational Thinking with Programming	Selby [31]	Empirical study	Teachers, Students, and industrial employee in the area of problem-solving, CT and programming	Context is not mentioned	Informal	The study revealed that CT approach is a requirement for 21st century problem-solving and programming education
Computational Thinking in Educational Activities An evaluation of the educational game Light-Bot	Gouws et al. [32]	A computational thinking framework designed and evaluated using Light-Bot	Not specified	Context not specified	Formal	The evaluation of CTF using an educational game, Light-Bot shows to be a useful for practicing computational thinking, with an overall CT score of 74%
A freshman seminar on problem solving and algorithmic thinking	Lamagna [33]	Course designed for real-world problems with computational approach; students engaged in solving challenging puzzles with a mathematical theme	Computer science and mathematics students	Context is not mentioned	Formal	Assessments and attitude surveys conducted at the beginning and end of the course shows the effectiveness of the approach and a high level of students' satisfaction with their experiences in gaining problem solving-skills
A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class	Yuen and Robbins [34]	Course design using data-driven approach	Biology major students in a computer science course (CS0)	University in South Texas, USA	Formal	The evaluation of the study shows the visualization component of CT appears to provide valuable feedback for students in accomplishing the programming tasks
Computational Thinking is Critical Thinking: Connecting to University Discourse, Goals, and Learning Outcomes	Kules [35]	Explorative design to compare critical thinking CritT and computational thinking CompT	Entry level students at the University of Maryland	University of Maryland	Formal	The study provided guidelines towards building CompT
A Case Study of Computer-based Problem-Solving Skill Development by Using Spreadsheet Software	Chatvichienchai [36]	Seminar and course Design	second-year and third-year students	University of Nagasaki, Japan	Formal	Since this study is a report of an ongoing project, the author expressed that the use of Excel micro would allow students to spend less time for program development than other conventional programming languages
Teaching Inclusive Thinking in Undergraduate Computing	Palan et al. [37]	Course design to teach inclusive thinking	undergraduate computer science students	Context is not mentioned	Formal	Participants formed groups and performed tasks that led to design of mobile app, website, or desktop application while following a design process
Computational thinking and programming education principles	García-Peñalvo [17]	A highlight of presentations regarding CT at HEIs in TEEM Conference, 2018	Pre and entering university students	Context not specified	Formal	The report shows many contributions that discussed CT as an approach for programming education
Relationship between Computational Thinking and a Measure of Intelligence as a General Problem-Solving Ability	Boom et al. [38]	A course design with empirical survey	pre-service teacher students	Context not specified	Formal	The result from the study shows that there is a strong relationship between computational thinking and intelligence; i.e., as capability of CT increases, student's intelligence tends to increase as well
A Novel Interdisciplinary Course in Gerontechnology for Disseminating Computational Thinking	Yang, et al. [39]	Course design & workshop	Undergraduate students from three programs: CS, Gerontoloty, and Graphic design	Iowa State University	Formal	Results from this study shows that the course produced significant improvement in students' self-reported competency in computational thinking and computer technology
A Pragmatic Approach to Develop Computational Thinking Skills in Novices in Computing Education	Philip et al [25]	A course designed to develop computational thinking skills of novices	Computer engineering students	Engineering College in Kochi	Formal	The results and students' feedback from this study he students' feedback showed that the course has increased their level of confidence in computer programming and problem-solving skills
A Fun-Learning Approach to Programming: An Adaptive Virtual Reality (VR) Platform to Teach Programming to Engineering Students	Chandramouli et al. [40]	A Virtual Environment (VE) framework to teach programming concepts	students of STEM and STEAM	Context not specified	Formal	Although the authors claimed that the VR framework can enhance students' understanding of programming, there is no clear report about the evaluation of the proposed framework.
Teaching and Learning Computational Thinking through Solving Problems in Artificial Intelligence on Designing Introductoctory Enginnering and Computing Courses	Silapachote and Srisuphab [41]	A course designed to integrate AI activities	Engineering and computer science undergraduates	Context not specified	Formal	Results shows that solving AI problems is an effective tool in teaching and learning computational thinking for entering undergraduates in computing and engineering fields.
Design Thinking and Computational Modeling to Stop Illegal Poaching	Padmanabhan et al. [42]	A design thinking framework	Multi-level students (k-12, undergraduates, and graduates)	Context not specified but focused on developing countries	Informal	This work clearly helped to establish STEM education partnerships and alliances outside traditional boundaries and has helped to create new synergies between the three levels of education K-12, undergraduates and graduates in solving real-world problems

Teaching Computational Thinking to Entry-level Undergraduate Engineering Students at Amrita University	Shyamala et al. [43]	A course design approach	Entry-level undergraduate engineering students (numbering over 2500)	Amrita University, India	Formal	In this work, the authors reported that computational thinking course should be designed to have more emphasis on hands-on sessions with lectures supporting those sessions rather than theoretical sessions
CS for ALL: Introducing Computational Thinking with Hands-on Experience in College	Jung et al. [44]	A course design with practical exercises using Scratch and mBot	Undergraduate students	Context not specified	Formal	Students’ feedback shows a high level of enthusiasm and engagement during the course. This hands-on learning nature of the course helped the non-computing major students to have more understanding through interactive classroom experience within a team
The effect of simulation games on the learning of computational problem solving	Liu [45] et al.	Simulation games-based approach	First-year students in a university	northern Taiwan	Formal	The findings in this study shows that students when learning computational problem solving with the game were more likely to perceive a flow learning experience than in traditional lectures. Also, their intrinsic motivation to learn was enhanced
A serious game for developing computational thinking and learning introductory computer programming	Kazimoglu et al. [46]	Games-based learning approach	Undergraduate computer science students	Context not specified	Formal	Survey response from group of students in computer science and related degree programmes confirmed that they found the game enjoyable, and also beneficial in helping students to learn problem-solving skills for introductory computer programming
Learning Programming at the Computational Thinking Level via Digital Game-Play	Kazimoglu et al. [47]	Games-based framework to teach CT	Computer science students	Context not specified	Formal	The preliminary evaluation of this game revealed that the majority of participants found the game well suited to help students to understand introductory programming constructs
A validity and reliability study of the computational thinking scales	Kormaz et al. [48]	An exploratory study to determine the level of computational thinking skills among students	Undergraduate students and distance learning students	Amasya University	formal and informal	The study concluded that the scale used in the experiment is a valid and reliable measurement tool that could measure the computational thinking skills of the students. Also, the digital age individuals are expected to have the computational thinking skill
Computational thinking development through creative programming in higher education	Romero et al. [49]	Course design with creative programming projects	Non-computer science undergraduate students	Canadaian University	Formal	The outcome shows how students’ assessment in a creative programming can be automated as used in Dr. Scratch
A Case Study to Promote Computational Thinking: The Lab Rotation Approach	Cai et al. [50]	blended learning	Non computer science first year college students	Chinese higher education	Formal	The result shows improvement in students’ computational thinking method
From Computational Thinking to Constructive Design with Simple Models	Margaria [18]	short courses, bootcamps, and semester-long courses on Model-Driven Design’	Non computer science first year University students	Ireland University	Formal	The author reported that the study revealed enhancement in the constructive design and critical thinking skills of the students
Applying online externally-facilitated regulated learning and computational thinking to improve students’ learning	Tsai and Tsai [51]	Course design on ERL and CT	Undergraduate students	Context not mentioned	Formal	Students with ERL support developed their CT skills better than those without
Supporting Undergraduate Computer Science Education Using Educational Robots	Saad et al. [52]	Use of Robort for introductory CS	Computer science students	Armstrong Atlantic State University	Formal	The result shows that students are motivated to increase their interest to further learn introductory programming
Infusing Computational Thinking across Disciplines: Reflections & Lessons Learned	Pollock et al. [20]	Course Design	Undergraduate students from multiple disciplines	Context not specified	Formal	Provided a model for integrating CT into faculty of multiple disciplines
Computational Thinking in a Game Design Course	Settle [53]	Tic-Tac-Toe game design	DePaul University Undergraduate students	Chicago	Formal	The course helped students to better classify and understand loops through game design
MPCT – Media Propelled Computational Thinking	Freudenthal et al. [54]	Course design called MPCT	Entry students’ program for freshmen	University of Texas at El Paso (UTEP)	Formal	MPCT was evaluated to help students acquire computational reasoning, integrates both programming and mathematics
Comparing Block-Basedand Text-BasedProgramming in High School Computer Science Classrooms	Weintrop and Wilensky [55]	Course Design	Introductory programming students	Context not mentioned	Formal	The comparative study revealed the impact of the visualized programming and text-based programming, and helps to answer the question on how best to introduce today’s students to essential computing concepts
Teaching Computational Thinking to Non-computing Majors Using Spreadsheet Functions	Yeh et al. [56]	Course Design	College students	Context not mentioned	Formal	The study revealed some of the challenges and weakness of teaching CT to novice learners
Learning to think like a trainer: bringing Scratch for Educational Sciences professional’s formation	Almeida and Pessoa [57]	Course Design	Degree students of educational science	Context not mentioned	Un-specified	Shows that students' extra motivation to learn CT with this type of tool especially when working in groups by identifying problems and solving them using the opinions of the different members of the group

4.2 Tools, interventions and design approach to CT for programming education at HEIs

In order to provide an answer to the research objectives highlighted in section 1, the study investigated the tools and research design employed for teaching/learning of computational thinking at HEIs. Analysis shows that the majority of the study reviewed utilized course design approach to teach computational thinking at HEIs. Although some studies did not explicitly mention the approach used, however, workshop, seminar, and exploratory study forms part of the strategies adopted in introducing CT for computing education. A few studies introduced computational thinking at higher education institutions by using a concrete tool and technology. These tools are developed/adopted to build problem-solving, algorithmic, and programming knowledge among the freshmen. Some of the tools leveraged on the common software such as Microsoft Excel, while some are designed to be game/puzzle-based approach. However, many of the studies did not specify the name of the tool or technology utilized.

5 DISCUSSIONS

In this review, we analyse scholarly articles on the use of computational thinking approach for programming education at higher education institutions. By doing so, we investigated the study design, tools, and technology adopted, and the impact of these studies. This section discusses the findings presented in section 3 regarding the adoption of CT for programming education at HEIs and the impacts of such studies on the students within the context of discuss. Although studies on the use of CT approach for teaching programming education at lower education exists [58] [59] [60], our study has shown that CT courses have been designed and administered to undergraduate entry-level students to prepare them for introductory programming courses (IPCs) that requires rigorous logical and algorithmic skills. For example, Ref [46] [33] [34] [36] [49] [44] [39] utilized CT approaches and tools such as Wiki-based project, MATLAB, puzzles, spreadsheet micro, Scratch, and robot to introduce new students to IPCs and problem-solving.

5.1 Reflections from the use of CT approach for programming education at HEIs and its impact

Educators and researchers have reported their experiences from the conduction, analysis, and evaluation of the CT approach introduced among the freshers undergraduate at HEIs. While some results showed a positive outcome, others have mixed feelings about the way different categories of students reacted to the CT courses [39, 61, 25]. On the positive side, the introduction of CT as a prerequisite for introductory programming courses was shown to bridge the gap that exists between students with and without programming background [25]. For instance, feedback from students in a study shows that "the course has helped them to improve their problem solving skills…and they performed well in their regular programming sessions" [25](pp.202). In another study, the use of games to teach students CT and problem-solving skills showed that some students had flow experience; some were bored and anxious during the simulation; some demonstrated analytic reasoning strategies to learn computational problem-solving skills, while others applied the trial-and-error approach to problem-solving [33] [45]. In addition, a study that teaches programming within a data-driven context in a university and allowed students to learn computing concepts and computational thinking by writing programs in MATLAB was conducted [34]. This study shows that computational and visualization tasks appear to be closely linked, and the visualization component appears to provide valuable feedback for students in the programming tasks accomplishment.

However, a few of the articles did not present a concise result about the impact of CT on students at HEIs; fundamental concern raised about the distinctive boundary-line between computer science and CT; curriculum design that compliments the field of computer science also requires further discussion. Overall, the introduction of CT at undergraduate level have been reported to impact positively on the students by motivating interest in programming, enhancing problem-solving skills and cognitive capabilities, and improving interdisciplinary teamwork participation (see Table 5).

5.2 CT approach for programming education in a smart learning environment

Nowadays, there is high interest on adopting a smart learning approach for teaching and learning programming education [62]. Smart learning environment (SLE) is a new paradigm in the learning ecosystem that seeks to enhance the learning experience. The aim of SLE is to allow for learner-centred approach by personalizing, adapting, and contextualizing learning. The use of CT for problem-solving and programming education is a viable approach towards designing an interactive SLE that can aide the learning process for especially novices. This section discusses the use of CT for teaching/learning of programming education as a potential approach towards designing a SLE at HEIs.

In our previous work Agbo et al. [62] had identified four components that are relevant when designing a SLE. These components, which consists of user, context, pedagogy, and tools, are investigated in this study and are linked to the current findings. This study revealed specific context of use, for instance, undergraduate students in China [50] [25], University students in Ireland [18], computer science students in a Canadian University [49], and US high school teachers [29]. Similarly, the pedagogical component of these studies that we have reviewed are presented in Table 2. The result showed that the majority of the articles utilized course design approach as their pedagogy, and a few studies employed workshop/seminar approach. Others such as [36] have used project design approach and integrated into the curriculum of a semester or section to evaluate the students collaborative and cooperative problem-solving skills. For instance, some authors designed courses that were focused on

computer programming concepts such as variables, data types, algorithmic thinking, functions, recursion, and problem-solving techniques [39, 25, 42, 41, 43]. Besides, puzzle-based techniques, games, and creative exercises were other strategies adopted for introducing the concept of CT and problem formulation at the undergraduate level of education [33, 61, 17].

In addition, our analysis based on the type of tools and technology deployed to conduct the studies showed that majority did not specify the tool. However, tools such as Microsoft Excel Spreadsheet [36], MATLAB [34], Puzzles [33], Scratch and mBot [49] [44], were employed at different context of HEIs.

The integration of CT in a smart learning environment can create a more supportive platform for teaching/learning of programming. While the CT is a pedagogical approach adopted to present learning contents, the SLE allows the modeling of users to enhance learning process by personalizing, adapting, and motivating learning.

5.3 Sample cases of CT approach for programming concepts

In this section, we present some of the CT approaches used to teach students on how to develop their thinking ability, problem-solving skills, and improve their programming knowledge. These cases are adapted from Lagama [34]. The first case of CT approach used to teach programming concept is about an aged long historical problem with recreational mathematics. This problem is called ‘river crossing puzzle.’ This puzzle teaches the students how to think algorithmically. The storyline to the puzzle consists a Wolf, a Goat, and a Cabbage. “A farmer has to take all these items across a river, and the only boat available can transport the farmer and at most one of his possessions at a time” (see Figure 7 left). A second case has to do with a mathematical logic, which is an important aspect of building one’s computational thinking skill.

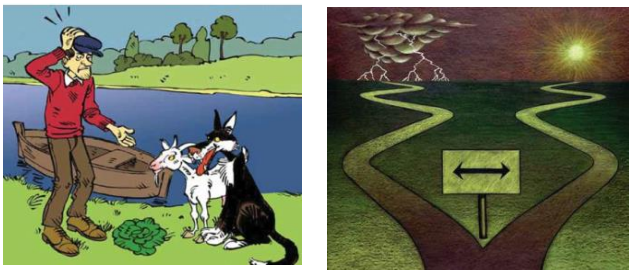

Figure 7: Case 1 & 2 of CT approach. adapted from [34]

Most real-world problems that require computer programming require logic expressions and operators. A puzzle called ‘a fork in the road’ was used to introduce logic. The situation is such that you’ve been out alone for a long walk on the island, and unfortunately, you got lost. You are very tired, and it’s already getting dark, at this point you come to a fork in the road with a conspicuous signpost on the ground, on it is written ‘one road leads back home and the other road leads to den of venomous snakes.’ Fortunately, at the fork, there’s a native who will only answer a yes/no question for you. But, you do not know whether the native is a Truth teller or Liar. What question should you ask to determine with certainty the way back home?

In building problem-solving skills for freshmen, the use of common and real world-problems enhances their understanding, to creatively think of computational steps towards a solution. Among other cases, the examples we adapted shows that CT can be used to teach programming concepts. Hence, it is important to design a learning tool that allow users teach/learn programming concept through CT approach while abstracting problems that are familier within a context.

6 CONCLUSIONS

Computational thinking has become a fundamental approach to building problem-solving skills. The use of CT approach to introduce freshmen to introductory programming courses at the higher education institutions has been gaining research interest in the recent past. This study has identified trends and approaches employed by educators to introduce CT at HEIs and its impact on students. It was discovered that there is increasing number of research publications regarding the use of CT to promote programming education at HEIs since 2017. The research publication regarding the use of CT for programming education seems to be more in the US Universities. The possible explanation for this trend can be linked to the presence of Computer Science Teachers Association (CSTA) annual conference[1] that holds in the US and has existed for about two decades.

The penetration of CT approach into the higher education institutions indicates good sign for computing education as it develops student’s cognitive ability and prepares the freshmen for the core programming courses. This way, the existing gap between the students who have programming background and those without the background can become narrow since CT teaches the fundamentals of programming concepts. Also, the educational technologists can use the CT approach to design a smart learning solution that enhances the students to have a better learning experience. In fact, we believe that the future of teaching/learning of computer programming is a SLE with the CT approach; that provides the opportunity for a more impactful learning experience. Our future research will be in the direction of exploring models that employed CT approaches to implement a SLE to aide teaching and learning of programming education.

7 REFERENCES

[1] C. Chang, “Using Computational Thinking Patterns to Scaffold Program Design in Introductory Programming Course,” in *5th IIAI International Congress on Advanced Applied Informatics*, 2016.

[2] P. Denning and M. Tedre, Computational Thinking, London: The MIT Press, 2019.

[3] Maleko, M; Hamilton, M; D'Souza, D, “Novices’ Perceptions and Experiences of a Mobile Social Learning Environment for Learning of Programming,” in *Proceeding in 12th International Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, Haifa, Israel, 2012.

[1] https://csteachers.org/page/history-of-the-csta-annual-conference

[4] L. Williams, E. Wiebe, K. Yang, M. Ferzli and C. Miller, "In support of pair programming in the introductory computer science course," *Journal of Computer Science Education,* vol. 12, pp. 197-212, 2002.

[5] S. Dasuki and A. Quaye, "Undergraduate students' failure in programming courses in institutions of higher education in developing countries: a Nigerian perspective," *The Electronic Journal of Information Systems in Developing Countries* , vol. 76, no. 8, pp. 1-18, 2016.

[6] V. Renumol, S. Jayaprakash and D. Janakiram, "Classification of cognitive difficulties of students to learn computer programming," *Indian Institute of Technology,* p. 12, 2009.

[7] R. A. M. S. Ahmed, S. M. Mahmood, R. M. Nabi and D. L. Hussein, "The Impact of Teaching Materials on Learning Computer Programming Languages in Kurdistan Region Universities and Institutes," *Kurdistan Journal of Applied Research (KJAR)* , vol. 3, no. 1, pp. 27-33, May 2018.

[8] I. Stamouli, D. E. and M. Huggard, "Establishing structured support for programming students," in *In Proceedings of the 34th American Society of Engineering Education ASEE/IEEE frontiers in Education Conference*, Savannah, 2004.

[9] S. S. Oyelere, J. Suhonen, G. M. Wajiga and E. Sutinen, "Design, development, and evaluation of a mobile learning application for computing education," *Educational Information Technolology, Springer,* p. 467–495, 2017.

[10] D. D'Souza, M. Hamilton, J. Harland, P. Muir and C. Thevathayan, "Transforming learning of programming: a mentoring project," in *In Proceedings of the 10th Australasian Computing Education Conference*, Wollongong, Australia, 2008.

[11] Y. Li, "Teaching Programming Based on Computational Thinking," in *IEEE Frontiers in Education Conference (FIE)*, USA, 2016.

[12] J. M. Wing, "Computational Thinking," *Communication of the ACM,* vol. 49, no. 3, pp. 33-35, 2006.

[13] Velázquez-Iturbied, J. Á., "Towards an Analysis of Computational Thinking," *International Symposium on Computers in Education (SIIE)*, Spain, 2018.

[14] V. Barr and C. Stephenson, "Bringing Computational Thinking to K-12: What is the role of the computer science community," *ACM Introads,* vol. 2, no. 1, pp. 48-54, 2011.

[15] L. Mannila, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo, L. Rolandsson and A. Settle, "Computational Thinking in K-9 Education," in *Proceedings of the working group reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, New York, USA, 2014.

[16] B. Liu and J. He, "Teaching Mode Reform and Exploration on the University Computer Basic based on Computational Thinking Training in Network Environment," in *The 9th International Conference on Computer Science & Education (ICCSE)* , Vancouver, Canada, 2014.

[17] F. García-Peñalvo and J. Benito, "Computational thinking in pre-university education," in *TEEM'16 conference*, Salamanca, Spain , 2016.

[18] T. Margaria, "From Computational Thinking to Constructive Design with Simple Models," in *Springer Nature Switzerland AG,* Switzerland , 2018.

[19] S. Y. K. J. H. L. Lye, "Review on teaching and learning of computational thinking through programming: What is next for K-12?," *Journal of Computers in Human Behavior,* vol. 41, pp. 51-61, 2014.

[20] L. Pollock, C. Mouza, K. Guidry and K. Pusecker, "Infusing Computational Thinking across Disciplines: Reflections & Lessons Learned," in *SIGCSE'19*, Minneapolis, USA, 2019.

[21] L. Araujo, A. W. L. and D. Guerrero, "A Systematic Mapping Study on Assessing Computational Thinking Abilities," in *IEEE,* USA, 2016.

[22] J. Moreno-León and M. Román-González, "On Computational Thinking as a Universal Skill," in *IEEE Global Engineering Education Conference (EDUCON)*, Spain, 2018.

[23] V. Shute, C. Sun and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review,* vol. 22, pp. 142-158, 2017.

[24] A. Repenning, W. D. C. C. Brand, F. Gluck, R. Grover, S. Miller, H. Nickerson and M. Song, "Beyond Minecraft- Facilitating Computational Thinking through Modeling and Programming in 3D," in *IEEE Computer Graphics and Applications*, G. Domik and S. Owen, Eds., USA, University of Colorado Boulder, 2014, pp. 68-71.

[25] M. Philip, V. G. Renumol and R. Gopeekrishnan, "A Pragmatic Approach to Develop Computational Thinking Skills in Novices in Computing Education," in *IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, Jaipur, India, 2013.

[26] M. Tedre, Simon and L. Malmi, "Changing aims of computing education: a historical survey," *Computer Science Education,* pp. 1-29, June 2018.

[27] B. Czerkawski and E. Lyman, "Exploring Issues About Higher Education," *TechTrends,* vol. 59, no. 2, pp. 56-65, 2015.

[28] D. Hickmott, E. Prieto-Rodriguez and K. Holmes, "A Scoping Review of Studies on Computational Thinking in K – 12 Mathematics Classrooms," *Digit Exp Math Educ* , vol. 4, no. 48, 2018.

[29] P. Morreale, "Connecting undergraduate programs to high school students: teacher workshops on computational thinking and computer science," *The Journal of Computing Sciences in Colleges,* vol. 25, no. 6, pp. 191-197, 2010.

[30] C. Wellington and R. Ward, "Using video to explore programming thinking among undergraduate students," in *ACM Consortium for Computing Sciences in Colleges*, 2010.

[31] C. C. Selby, "Promoting Computational Thinking with Programming," in *WiPSCE '12*, Hamburg, Germany, 2012.

[32] L. Gouws, K. Bradshaw and P. Wentworth, "Computational Thinking in Educational Activities An evaluation of the educational game Light-Bot," in *In proceeding of ITiCSE'13*, 2013.

[33] E. A. Lamagna, "A Freshman Seminar on Problem Solving and Algorithmic Thinking," in *Consortium for Computing Sciences in Colleges*, USA, 2014.

[34] T. Yuen and K. A. Robbins, "A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class," *ACM Transactions on Computing Education (TOCE),* vol. 14, no. 4, 2015.

[35] B. Kules, "Computational thinking is critical thinking: connecting to university discourse, goals, and learning outcomes," Copenhagen, Denmark, 2016.

[36] S. Chatvichienchai, "A Case Study of Computer-based Problem Solving Skill Development by Using Spreadsheet Software," in *Proceedings of the International Conference on Digital Technology in Education- ICDTE '17*, Taipei, Taiwan, 2017.

[37] N. Palan, V. Hanson and M. L. S. a. Huenerfauth, "Teaching Inclusive Thinking in Undergraduate Computing," in *In proceeding of ASSETS'17*, Baltimore, 2017.

[38] K. Boom, M. Bower and A. Arguel, "Relationship between Computational Thinking and a Measure of Intelligence as a General Problem-Solving Ability," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology* , Cyprus.

[39] H. Yang, P. Martin, D. Satterfield, B. R., J. Wong, M. Shelley and C. K. Chang, "A Novel Interdisciplinary Course in Gerontechnology for Disseminating Computational Thinking," in *41st ASEE/IEEE Frontiers in Education Conference*, Rapid City, SD, 2011.

[40] M. Chandramouli, M. Zahraee and C.-. Winer, "A Fun-Learning Approach to Programming: An Adaptive Virtual Reality (VR) Platform to Teach Programming to Engineering Students," in *IEEE International Conference on Electro/Information Technology*, USA, 2014.

[41] P. Silapachote and A. Srisuphab, "Teaching and learning computational thinking through solving problems in Artificial Intelligence: On designing introductory engineering and computing courses," in *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Bangkok, Thailand, 2016.

[42] P. Padmanabhan, B. Alexander, C. Caiseda, K. McLane, N. Ellanki, P. Seshaiyer, B. Kwon and E. Massawe, "Design Thinking and Computational Modeling to Stop Illegal Poaching," in *2017 IEEE Integrated STEM Conference (ISEC)*, Princeton, NJ, USA, 2017.

[43] C. K. Shyamala, C. S. Velayutham and L. Parameswaran, "Teaching Computational Thinking to Entry-level Undergraduate Engineering Students at Amrita University," in *2017 IEEE Global Engineering Education Conference (EDUCON)* , Athens, Greece , 2017.

[44] A. Jung, J. Park, A. Ahn and M. Yun, “CS for ALL: Introducing Computational Thinking with Hands-on Experience in College,” in *International Conference on Computational Science and Computational Intelligence,* Vegas, United States, 2017.

[45] C. Liu, Y. Cheng and C. Huang, “The effect of simulation games on the learning of computational problem solving,” *Journal of Computers & Education,* vol. 57, no. 3, pp. 1907-1918, 2011.

[46] C. Kazimoglu, M. Kiernan, L. Bacon and L. Mackinnon, “A serious game for developing computational thinking and learning introductory computer programming,” *Social and Behavioral Sciences, sciencedirect-Procedia,* p. 1991 – 1999, 2012.

[47] C. Kazimoglu, M. Kiernan, L. Bacon and L. MacKinnon, “Learning Programming at the Computational Thinking Level via Digital Game-Play,” in *International Conference on Computational Science, ICCS 2012,* 2012.

[48] Ö. Korkmaz, R. Cakir and Ö. M.Y., “A validity and reliability study of the computational thinking scales,” *Computer in human behavior,* vol. 72, pp. 558-569, 2017.

[49] M. Romero, A. Lepage and B. Lille, “Computational thinking development through creative programming in higher education,” *International Journal of Educational Technology in Higher Education,* vol. 14, no. 42, pp. 2-15, 2017.

[50] J. Cai, H. Yang, D. Gong, J. MacLeod and Y. Jin, “A Case Study to Promote Computational Thinking: The Lab Rotation Approach,” in *International Conference on Blended Learning*, Switzerland, 2018.

[51] M. Tsai and C. Tsai, “Applying online externally-facilitated regulated learning and computational thinking to improve students’ learning,” in *Univ Access Inf So*, Berlin Heidelberg , 2018.

[52] A. Saad, G. Loewen, T. Shuff and K. Burton, “Supporting Undergraduate Computer Science Education Using Educational Robots,” in *Proceedings of the ACMSE 2018 Conference*, 2012.

[53] A. Settle, “Computational Thinking in a Game Design Course,” in *In proceedings of SIGITE’11, ACM*, 2011.

[54] E. Freudenthal, M. Roy, A. Ogrey, T. Magoc and A. Siegel, “MPCT – Media Propelled Computational Thinking,” in *Proceedings of Special Interest Group on Computer Science Education (SIGCSE)* , 2010.

[55] W. Weintrop and U. Wilensky, “Comparing Block-Basedand Text-Based Programming in High School Computer Science Classrooms,” *ACM Transactions on ComputingEducation,* vol. 18, no. 1, 2017.

[56] K. Yeh, Y. Xie and K. F., “Teaching Computational Thinking to Non-computing Majors Using Spreadsheet Functions,” in *ASEE/IEEE Frontiers in Education Conference* , 2011.

[57] R. Almeida and T. Pessoa, “Learning to think like a trainer: bringing Scratch for Educational Sciences professional’s formation,” in *IEEE,* 2018.

[58] Vinayakumar R, K. Soman and P. Menon, “CT-Blocks: learning computational thinking by snapping blocks,” in *IEEE,* Bengaluru, India, 2018.

[59] R. Vinayakumar, K. Soman and P. Menon, “Fractal Geometry: Enhancing Computational thinking with MIT Scratch,” in *IEEE,* Bengaluru, India, 2018b.

[60] H. Y. Durak, “The Effects of Using Different Tools in Programming Teaching of Secondary School Students on Engagement, Computational Thinking and Reflective Thinking Skills for Problem Solving,” *Technology, Knowledge and Learning,* 2018.

[61] F. Agbo, S. Oyelere, J. Suhonen and M. Tukiainen, “Identifying potential design features of a smart learning environment for programming education in Nigeria,” *Manuscript submitted to a journal for review,* 2018.