International Journal of Advanced Smart Convergence Vol.6 No.2 24-33 (2017)
https://doi.org/10.7236/IJASC.2017.6.2.24

IJASC 17-2-4

Guided Instruction of Introducing Computational Thinking to Non-Computer Science Education Major Pre-Service Teachers

Ki-Sang Song

Dept. of Computer Education, Korea National University of Education, Korea
kssong@knue.ac.kr

Abstract

Since 'teaching coding' or 'programming' classes for computational thinking (CT) education in K-12 are renowned around the world, a pre-service teachers' readiness for integrating CT into their teaching subjects is important due to the fact that CT is considered to be another 'R' from algoRitm for 21st century literacy, in addition to the traditional 3R (Reading, Writhing, and Arithmetic) [2] and CT roles to other disciplines. With this rationale, we designed a guided instruction based CT course for pre-service teachers. We show the effectiveness of the program with respect to the teachers' attitude toward combining CT into their teaching subjects, and mindset changes of learning computing connected with the career development of the teacher themselves. The research focused on the instructional methodology of teaching programing for non-Computer Science Education (CSE) majors who are not familiar with computer science for alleviating the cognitive load of first exposure to programming course under the CT concepts.

Key words: *Computational thinking, non-CSE majors, Teacher training, Teaching method*

1. Introduction

Due to the recognition of the importance of computing's role relating with other disciplines and the intellectual framework of thinking founded on computer science, many countries have started to provide a Computational Thinking (CT) course as one of the basic courses not only in higher education but in K-12 computing education [1], [2] as well.

As Wing suggested CT is not only for computer scientists but a fundamental skill of analytical thinking for everyone [2]. It is believed that everyone can benefit from computationally [4] thinking, and there is ample evidence that science and engineering disciplines have produced great outcomes using computational concepts such as algorithms and data structure used in computer science. Today, it is notable that social studies also benefits from applying computational thinking concepts [4]. As a result, the concepts used in CT need to be taught in disciplines other than science [5], providing the opportunity for all disciplines to apply the computational thinking structure.

These trends affect the education sector creating an increased need for teaching CT to pre-service teachers

Manuscript Received: Mar. 17, 2017 / Revised: Mar. 28, 2017 / Accepted: Apr. 10, 2017
Corresponding Author: kssong@knue.ac.kr
Tel: +82-43-230-3751, Fax: +82-43-231-3790
Dept. of Computer Education, Korea National University of Education, Korea

who are non- Computer Science Education (CSE) majors [4]. Usually introducing CT to students means teaching how to create programs or teaching coding techniques based on CT concepts to solve given problems within a relatively easy programming environment.

Because the introductory computer science courses focus on teaching programming to first year CS majoring students, teaching programming to non-CS majors. The introduction of programming to first year CS majors is not an easy task and many controversies surround its teaching methodology [6]. The teaching method for pre-service teachers is important due to its transferring effect when they are assigned as K-12 teachers.

Some researchers like Guzidial argue that the direct introduction of programming to CS professionals does not consider the educational technology aspects [7] thus result in the method demands high cognitive load from the students. Jiang and Kirschner along with their colleagues [6], [8] support Guizidal's argument surrounding minimal guidance of instruction not being effective methodology.

This paper uses the literature review of teaching programming methodology to non-CS majors, applies the guided instruction methodology for non-CSE major pre-service teachers' CT course, and measures the effectiveness of this method in teaching programming. Unlike other papers introducing CT to non-CS majors with minimal guidance instruction, this paper reports the results of step by step appropriate guided instruction, increased independent tasks producing a better learning result and student satisfaction.

2. Theoretical Background

2.1 Literature review of introducing programming to non-CS majors

2.1.1 The necessity of teaching CT to non-CS majors as programming

As Wing [2] stated, CT displays systematic thought processes when one tries to solve problems by formulating and finding the solution using an information-processing agent. Despite Wing's claims that the key concept of CT is abstraction, the introduction of CT to non-CS majors teaches coding based on the developed algorithms according to subdomains of CT such as Table 1.

Table 1. Concepts of CT (Source: ISTE, CSTA, NSF [9])

Concept	Definition
Data Collection	The process of gathering appropriate information
Data Analysis	Making sense of data, finding patterns, and drawing conclusions
Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images
Problem Decomposition	Breaking down tasks into smaller, manageable parts
Abstraction	Reducing complexity to define main idea
Algorithms & Procedures	Series of ordered steps taken to solve a problem or achieve some end
Automation	Having computers or machines do repetitive or tedious tasks.
Simulation	Representation or model of a process. Simulation also involves running experiments using models.
Parallelization	Organize resources to simultaneously carry out tasks to reach a common goal.

(Source: Song. et. al. paper [10])

Among the 9 concepts of CT skills listed in Table 1, programming is emphasized in automation and simulation to see the effectiveness of the designed algorithm. Some papers suggest that the programming steps can be skipped because of the importance of developing algorithms, however we have observed that students often want to see the correctness of the designed algorithm by executing their own computer program. Therefore, it is a natural consequence that teaching CT includes teaching programming as a 'coding education' format.

Many papers report the results of different approaches to teaching programming to non-CS students. Chilana et. al. [11] argued that if someone is not becoming a professional programmer nor an end-user programmer what is the learning goal of programming for non-CS majors. The question of a learning goal can be addressed when we teach coding to not only pre-service teachers but also K-12 students. Although the fundamental question of the value of learning coding exists, it is widespread that many universities interdisciplinary undergraduate programs are introducing CT. Fincher [12] compared four approaches of teaching programming; "syntax-free," "literacy," "problem-solving," and computation as interaction approaches. Jiang and his colleagues designed a computer program for non-CS majors by combining the American model of giving the overall picture first and checking learning outcomes later, as well as the Chinese model of producing results after each step [6].

2.1.2 Reasons for teaching CT to non-CSE majors in teacher training courses

In addition to traditional reasons for teaching CT to pre-service teachers assuming most of them are non-CS majors, we need to focus on the professional development of teachers related with technology advancement as well as the teachers' role affecting students future career development. For example, the role of computing is changing and is now acknowledged as an innovative agent resulting in a need for introducing the importance of computing to generations to come. The knowledge of applying computing in a teachers' personal life is inevitable with today's technology enriched societies. Due to this rationale, the need for providing CT courses to every pre-service teacher including non-CSE major ones is an obvious.

2.2 Fully guided instruction and minimal guided instruction

Typically the programming courses in CS settings, first introduce syntax of specific computer language followed by task-driven programming practices. However, some researchers argue that the pedagogy for teaching programming needs to consider the cognitive load with respect to instructional aspect. Although few researches show the effectiveness of computing education methodology in regards to teaching programming at the introductory level of computer science relation to educational technology. It is worth noting the importance of Guzdial's arguments on minimal guidance of teaching programming not being effective [7].

In addition to traditional CS teaching issues, when we introduce a programing course to pre-service teachers it is necessary to consider the school environment. For example, many countries try to implement coding in K-12 education, the NSF in America created a goal for 10,000 teachers to be able to teach a computing subject by 2015 [13]. They attempted to secure highly qualified teachers for teaching CT in primary and secondary schools. Therefore, it is very important to call attention to the teaching methodology of programming at the K-12 level of education and the need for developing a good practice of teaching coding to non-CS major pre-service teachers in order to integrate CT into other subjects due to the transferring effect of pre-service learning.

2.2.1 Minimal instruction approach for teaching coding

Guzdial cited that Kirschner, Sweller, and Clark (KSC)'s paper [8] stated that minimal guidance during instruction does not work. As it is described in KSC's paper, this instructional methodology does not work, and it is believed that minimal guidance methodology often applied in Constructivist, Discovery, Problem-based, Experiential, and Inquiry-based teaching methods results in a failure in instruction.

It is not easy to define the level and amount of 'guidance' in minimal instruction for programming teaching. The minimal guidance during programing instruction includes first teaching syntaxes of certain programming language followed by students solving problems such as finding appropriate solutions, finding algorithms and coding to check the validity of a solution. Most activities require students to work independently regarding them as professional computer scientists. For pre-service teachers the process of problem solving in CS includes 'automation' and 'simulation' in computational thinking concepts, and proceeds to coding. Therefore simply applying traditional CS teaching methods may not work well for pre-service teacher programs.

2.2.2 Explicit instructional guidance approach for teaching coding

Unlike minimal guided instruction in programming courses where students are considered experts who work well under little guidance, the explicit instructional guidance provides full and explicit guidance for learning coding. As Clark and his colleagues [14] indicated, instead of requiring students to discover many aspects of what they must learn, teachers provide explicit guidance accompanied by practice and feedback in explicit instructional methodology. The main difference between minimal instruction and explicit instruction is asking students themselves to discover what to learn verses requiring students to practice recently learned content and skills.

When we introduce programming to pre-service teachers with specific instructional approaches we need to consider the transfer effect to schools. Around the world, coding education is implemented for nurturing students' capabilities of computational thinking. From coding education in primary schools, students may grasp the fundamental concepts of programming while developing computational thinking including logical thinking and problem solving skills.

Therefore, when we teach computational thinking to pre-service teachers, we need to equip them with the necessary tools to teach coding for either primary school or secondary school students. Furthermore, it is necessary to teach non-CSE pre-service teachers in order for them to integrate computational thinking with their teaching subject. Effective instructional methods of lowering cognitive load is essential for spreading coding into K-12 education.

3. Method

3.1 Designed guided instruction for introducing CT to pre-service teachers

3.1.1 Guided instruction in computational thinking courses

The purpose of this paper is to develop a better pre-service teacher training method that focuses on programming. Even though, many controversies surround teaching programming in CT teaching, we consider the worldwide trend of coding education [15]. Therefore, we combine both CT and the coding experience so that pre-service teachers may be exposed to knowledge and so that the acquired knowledge may be transferred to schools through their teaching.

The expected results of introducing CT to pre-service teachers are following; 1) they will show self-confidence in programming, 2) they will know the importance CT has on other subjects, 3) they will try to integrate CT in their potent teaching subject, and 4) they will assure the importance of computing for future jobs. To achieve such objectives we designed a CT course with step-by-step independent task execution, beginning with full instructor guidance and ending with independent student work, decreasing intervention gradually.

The term 'guided' in this paper is used because we provide step-by-step class management skills; lectures, a guided introduction on how to use and program with Scratch, showing how to analyze real-life system and develop working algorithms, group projects for developing programs, and finally assisting students on an individual coding problem. In addition to the instructor, we allocate one assistant. Therefore, even if students have been asked to formulate and develop a project combining their majoring subjects and CT knowledge, the coding load will not be too difficult or too intensive in the introduction to CT or an introductory programming course.

Also, we have applied the ‘use-modify-create’ framework suggested by Lee and his colleagues [16] considering the graphical programming environment such as Scratch, Alice, Arduino and Entry in an independent coding problem design stage. Scratch is widely used in schools and easy to learn, remix and recreate from existing programs. Although Scratch has many advantages in teaching, the simplicity of learning may cause a lack of training.

For example, it is not easy monitor what students are creating in Scratch and thus sometimes teachers may have no knowledge of a student’s work until they view the final Scratch project in the classroom. Those disadvantages may function as an opportunity for student’s unconstrained idea development but can make class management difficult. Therefore, we think providing guided instruction is a more desirable instructional methodology to teach at schools for pre-service teachers.

3.1.2 Designed computational thinking course

The course was designed for 3 hours/week and 3 credits for a 15 week liberal arts course aimed towards pre-service teachers of kindergarten, arts education, physical education and science education. However, any pre-service teachers can take the course even computer science education majors. 20 students registered for the course.

Considering most of the registered students were non-CSE pre-service teachers, we have applied the explicit instruction method dividing the course into five stages based on the use-modify-create model; *introduction of CT concepts, experience of Scratch-free visual programming language, experience of designing real-life system(algorithm design for functions of Automatic Teller Machine) as a mid-term take home examination, team project for program development, and individual programming* as shown in Table 2.

After introducing CT concepts, we provided twelve scenarios ranging from simple movement to complicated actions between actors for writing Scratch programs. Students derived algorithms based on their understanding of given scenarios applying CT concepts such as problem decomposition, and abstraction as shown in the Table 2. From Scratch experience to team project, a teaching assistant was assigned as a helper, and the ATM function development task was graded and given back to students for error correction.

In the ‘Experience of Scratch’ stage, students were given 12 problem sets as an introduction to programing ranging from easy to moderate. Most of the registered students do not have any experience in programing and they showed many mistakes even with very simple problems. For the students, the block commands of Scratch and their used cases were introduced from the ‘Projects’ uploaded in MIT Scratch site.

Table 2. Timeline of introduction of computational thinking and main topics

Week	Contents	Main topics
Week 1	Orientation / Introduction of Computational Thinking (CT)	Introduction of CT concepts
Week 2	History of computer / The principles of computing / Computer and program	
Week 3	Introduction of educational programing environment, LOGO, Scratch	

Week 4	Steps for problem solving with computers Expression of algorithms	Experience of Scratch
Week 5	Steps for problem solving with computers Expression of algorithms / CT concept 1	
Week 6	Steps for problem solving with computers Expression of algorithms / CT concept 2	
Week 7	Steps for problem solving with computers Expression of algorithms / CT concept 3	
Week 8	Mid-term: Design algorithm for ATM machine	Experience of real-life system
Week 9	Team project (Design scenario, Algorithm development and implementation 1)	Team project
Week 10	Team project II	
Week 11	Team presentation / Individual project design 1	
Week 12	Revise of individual project	Individual project
Week 13	Coding and testing 1	
Week 14	Coding and testing 2	
Week 15	Presentation of individual project	

For program development, students had been asked to design an algorithm first, and then do coding. Managing a classroom with 20 students as they complete a first ‘design algorithm’ and then coding is not an easy sequence because most students try to check whether their thinking is correct or not by programming. Even with a class size of 20 students the instructor may have difficulties observing students under a visual programming environment such as Scratch or Alice all at once.

The given 12 scenario based problems were used as a starting point for developing a new idea of applying CT to potential teaching subjects after graduation. Therefore, it is crucial to understand how to use the programming language itself in addition to the skills for developing ideas for solving given problems.

The most important part of instruction is asking students to develop their own projects connecting CT to their potential teaching subjects. We have provided some methods of reducing cognitive load to non-CSE major students by asking students to apply the 12 examples as templates for integrating their teaching subjects to CT concepts. From this perspective, the selection of 12 problems are important due to their function of a starting point to expand ideas. For reflections and helping each other, we provide group work sessions to secure a way of peer cooperation and sharing ideas.

4. Discussions and student feedback

Introducing CT to pre-service teacher training programs has various advantages; 1) exposing pre-service teachers to a new trend of emphasizing the need for computational thinking, 2) motivating pre-service teachers to integrate CT into their potential teaching subjects, and 3) realizing the importance of computing and guiding students for career development. After one semester of teaching we conducted a questionnaire survey for the effectiveness of introducing CT to pre-service teachers.

4.1 Methodology of teaching programming at CT Course and its transferability

Teaching programming in a CT course is often considered a fundamental element but various studies argue that teaching methods need to be improved for non-CSE major students. As pre-service teachers, the experience of learning how to program can potentially be passed on to their teaching methodology at schools. We have adopted the use-modify-create framework and step-by-step increase of student task difficulties so that even non experienced students will not be subject to solving tasks and programming alone without explicit instruction.

Some testimonials from students of non-CSE background after the individual projects:

I worried when I took this course because of not being skillful when working with computers and having no experience in writing computer programs at all. However, with the help of my professor and assistant I could learn the basics, and flowchart the algorithms to programming (Kindergarten education major student).

At first it was not clear how to integrate CT into physical education, my potential teaching subject, however I could apply the Scratch programming learned from the class. It was not clear at the beginning, but gradually it was concretized into my personal project. The final result gave me self-efficacy and I thought I could use it later in my teaching (Physical education student).

I never thought that I might integrate ICT (Information and Communication technology) into my teaching subject of music education before I took this course. It is a new experience step by step learning how to do programming, and it could be applied in my teaching (Music education student).

Most of the students' feedback on the teaching methodology commented on the step by step approach which proved helpful in understanding how to make programs with Scratch. They mentioned that provided materials and explicit teaching could be applied in their teaching at schools.

4.2 Understanding of the possibility of CT processes to other subjects

One of the successful results of teaching CT is the application of 'computational thinking' to problem solving. The capability is not simple doing programming but applying the thinking process to solve problems. Therefore, the pre-service teachers' understanding of the CT process to other subject teaching is important for future teachers.

We have surveyed this possibility with the following questions: 1) understanding the importance of computing with other subjects, 2) students' thinking of CT to be included in other subject teaching, and 3) CT can be integrated into other subject teaching. The student's feedback to the given questions are depicted in Fig 1. Students answered positively (>95%) to the possible applicability of CT to other subject teaching.

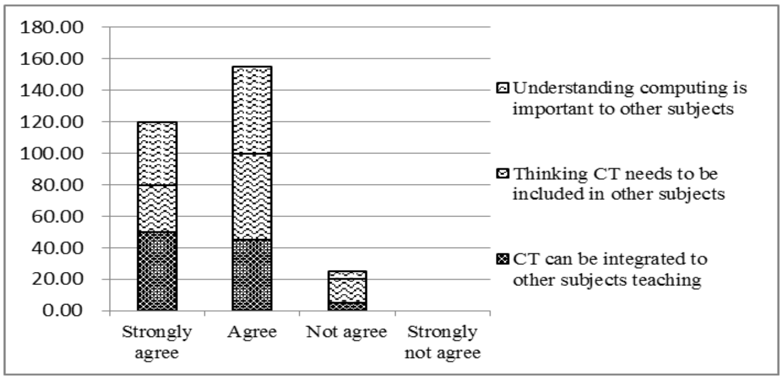

Figure 1. Accumulated responses of the possibility of CT processes to other subjects

4.3 CT learning and professional development

For teachers, professional development is an important concern to continue their teaching job. To be effective in CT pre-service teachers need to consider CT as useful knowledge in their professional development. To measure the effectiveness of this course we asked 1) expectation of computing knowledge for their job security, 2) expectation of computing concepts towards future jobs, 3) no need to learn computing for their future jobs.

Students answered positively (>90%) to the questions of computing roles to their future jobs, and negatively answered (>90%) to the question of no need of computing knowledge to their future teaching jobs. Such answers to contradict questions to check consistency of students' mind show that pre-service teachers have firm understanding of the necessity of computing knowledge to future job.

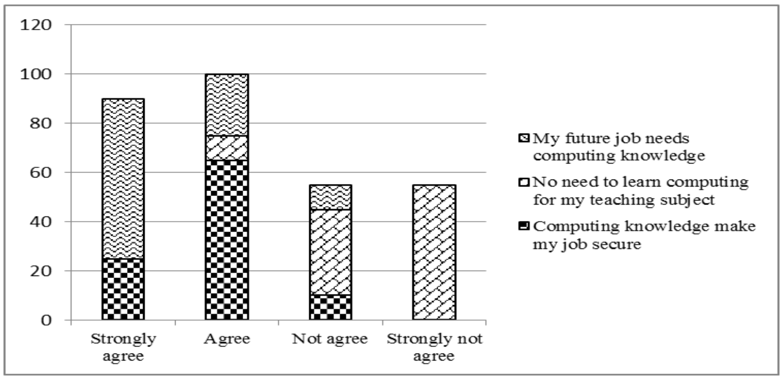

Figure 2. Accumulated responses of the pre-service teachers' opinion of CT to their professional development

4.4 Acknowledgement of the importance of computing for guidance counselling

As pre-service teachers, the recognition of the importance of computing related with future trends is enormously important because of their role of guidance counselling to students. Our survey results show that after CT learning pre-service teachers affirmatively response to 1) the expectation of computing concepts to future jobs, 2) assurance of daily use of computing knowledge, and 3) future jobs need of computing knowledge. From these results, pre-service teachers are expected to provide the importance of computing knowledge and computational thinking when they are assigned as in-service teachers and guidance counseling.

Their positive answers range from 85% to 90% show that pre-service teachers are not totally sure of their future job as teachers, but their understanding of computing to future job is firm. From these results, pre-service teachers are expected to provide the importance of computing knowledge and computational thinking when they are assigned as in-service teachers and guidance counseling.

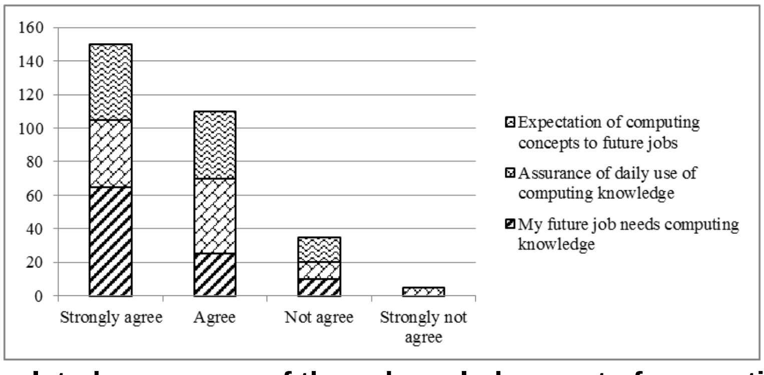

Figure 3. Accumulated responses of the acknowledgement of computing as guidance counselling

5. Conclusions

According to various reasons such as strengthening STEM in K-12 in the USA or up skilling digital competencies for the future in Europe, introducing CT in education as coding education then needs to develop effective computing educational strategies. Especially not only the computing education teachers but non-computing related subject teachers are also required to understand computational thinking as a thinking process of solving problems. Generally CT is introduced to non-computing majors as programing, when CT is introduced to pre-service teachers; it is natural to consider them as students of introductory computer science and programing courses.

Considering the school teaching coding environment, we applied the explicitly guided instruction method based on the use-modify-create framework, and learners showed that CT could be integrated in other disciplines and supports that CT needs to be included in other subjects. For pre-service teachers who may not directly teach CT in their majors, experiencing CT could change their perception on CT and its role as another 'R' for school education.

The explicit teaching method based on the use-modify-create framework with structured feedback resulted in effective learning outcomes especially since pre-service teachers aware of the importance of computational thinking for not only their CT knowledge but also the CT roles related with their professional development and potential possibility of guidance counselling after assigned as in-service teachers.

References

[1] A. Yadav, N. Zhou, and C. Mayfield, "Introducing Computational Thinking in Education Courses," in *Proc. of SIGCSE'11*, March 9–12, 2011.

[2] Wing, J.M., Computational Thinking, *Communication of the ACM*, Vol. 49, No. 3, pp.33-35, 2006.

[3] Wing, J.M., Computational Thinking Benefits Societies, *http://socialissues.cs.toronto.edu/index.html*.

[4] A. Yadav, C. Mayfield, N. Zohou, S. Hambrusch, and J. T. Korb, Computational Thinking in Elementary and Secondary Teacher Education, *ACM Transactions on Computing Education (TOCE)*, Vol 14, No. 1, pp. 5:1-5:16, March, 2014.

[5] S. Grover and R. Pea, Computational Thinking in K-12: A Review of the State of the Field, *Educational Researcher*, Vol. 422, No. 1, pp.38-43, 2013.

[6] Z. Jiang, E. B. Fernandez, and L. Cheng, "A Pedagogical Pattern for Teaching Computer Programming to Non-CS Majors," *in Proc. of the 18th Conference on Pattern Languages of Programs (PLoP). PLoP'11*, October 21-23, 2012.

[7] M. Guzdial, "How we teach introductory computer science is wrong," *https://cacm.acm.org/blogs/blog-cacm/45725 -how-we-teach-introductory- computer-science-is-wrong/fulltext.*

[8] P. A. Kirschner, J. Sweller, and R. E. Clark, "Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," *Educational Psychologist,* Vol. 41, No.2, 75-86, 2006.

[9] ISTE, CSTA, NSF, Computational Thinking Teacher Resources**,** *http://www.csta.acm.org/Curriculum/sub/CompThinking.html.*

[10] S.-Y. Park, K.-S. Song, and S.-H. Kim, Cognitive Load Changes in Pre-Service Teachers with Computational Thinking Education1, International Journal of Software Engineering and Its Applications Vol. 9, No. 10, pp. 169-178, 2015.

[11] P. K. Chilana, C. Alcock, S. Dembla, A. Ho, A. Hurst, and B. Armstrong, Perceptions of non-CS majors in intro programming: The rise of the conversational programmer, in *Proc. of IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2015.

[12] S. Fincher, What are We Doing When We Teach Programming? in *Proc. of the 29th ASEE/IEEE Frontiers in Education Conference*, Nov. 1-13, 1999.

[13] Computing Education Blog, What's the argument for becoming a computer science teacher? *https://computinged.wordpress.com/2011/02/07/whats-the-argument-for-becoming-a-computer-science-teacher/.*

[14] R. E. Clark, P. A. Kirschner, and J. Sweller, "Putting Students on the Path to Learning*," American Educator*, pp.6-11, Spring 2012.

[15] Computing Our Future, "Computer programming and coding Priorities, school curricula and initiatives across Europe," *European Schoolnet*, 2015.

[16] Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith, and L. Werner, "Computational thinking for youth in practice," *ACM Inroads*, Vol 2. No. 1, pp. 32-37, March 2011.