Towards a computer-assisted Computational Thinking (CT) assessment system in higher education

Xiaoling Zhang, Marcus Specht

Delft University of Technology, Van Moori Broekmanweg 6, Delft, 2628XE, The Netherlands

Abstract

With the vision to promote CT to a wider group of audiences, this PhD project explores the formative assessment of CT skills in Programming Education to support students to learn CT skills in Higher Education. In this project, we plan to investigate the importance of CT in the context of Higher Education, explore the relationship between CT skills and programming skills, build a model to assess learners' CT skills and develop a computer-assisted assessment system with automated components to enhance students' CT competences in Higher Education. Mixed-method research methodologies will be employed in distinct phases of the project accordingly. A system which allows formative assessment of CT skills will be iteratively designed and constructed throughout the project. The outcome of the project should support the CT learning process, make CT more visible for people from diverse backgrounds and empower them with a CT mindset to embrace the digitalization of society.

Keywords 1

Computational Thinking, Computer-Assisted Assessment, Higher Education, Educational Technology

1. Introduction

1.1. Digitalisation and Computational Thinking

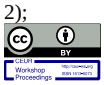
Living in an era of digitalisation, digital elements is everywhere. For instance, education, healthcare and governance, fundamentals to a modern society, are developing towards a digital direction [1-3]. This has a huge influence on employment and skills, such as the increasing unemployment rate, and the increasing demand for digital skills in the labour market [4]. To empower people the capability of living and working in such a digitalized society, governments, and education institutions from distinct levels world-wide have been striving to promote education of computer-based technologies and skills varying from academy to industry. Among skills being mentioned, digital skills, problem-solving skills, and computational thinking (CT) are the top few most mentioned skills and are regarded as fundamental skills in workplaces [5-7, 28].

Computational Thinking is closely related to the development of digitalisation in different domains and changes the professional competencies need for these professions. First proposed by Papert as procedural thinking [8] and then being promoted by Wing [9], a considerable amount of research has been conducted to define CT in the past few decades. Though there is no agreed-upon theoretical or operational definition so far, existing works share main components of CT, which are problem decomposition, abstraction, pattern recognition and algorithm [9-15]. Besides studying the operational and theoretical definition of CT, massive amounts of studies have been conducted globally to investigate topics around CT education, such as

Proceedings of the Doctoral Consortium of Seventeenth European Conference on Technology Enhanced Learning, September 12-16, 2022, Toulouse, France

EMAIL: x.zhang-14@tudelft.nl (A. 1); m.m.specht@tudelft.nl (A. 2);

ORCID: 0000-0003-0951-0771 (A. 1); 0000-0002-6086-8480 (A. 2);



pedagogical contents, didactic strategies, integration of CT into other disciplines [16-26].

People of almost all ages can be participants in these studies, however, most of the existing research focuses specifically on K-12 settings, with an increasing number of studies conducted in Higher Education over the last decade. Existing work in K-12 settings has explored a considerable range of topics regarding learning and teaching CT in both science, technology, engineering, and mathematics (STEM) and non-STEM disciplines, results in a flourishment of development in tools and activities for teaching and learning CT, both CS-unplugged such as bebras challenge and Lego construction and CS oriented such as programmable robotics, micro-bits, code.org, Scratch, Alice [20]. While being regarded as crucial competence for learners in higher education, the development of CT, compared to CT in K-12 setting, is still in its infancy. Increased attention has been paid to CT in Higher Education in recent years, most of which are related to Computer Science (CS) major, and few are in non-CS major disciplines [26]. In their literature review, Lyon and Magana identified several issues existing in current CT education which makes it difficult for students to understand CT, including unclear definition, lack of assessment methods, unclear use of CT in classrooms [26]. They also stressed the necessity of a clearer definition of CT and called for more implementation of CT in Higher Education and studies.

With current insights into existing literature, it is obvious to conclude that CT is closely related to developments of digitalisation in different domains and changes the professional competencies needed for these professions. However, it is still unclear how to embed CT in different curricula and how to develop transdisciplinary CT skills. Therefore, researchers need to conduct studies to establish a comprehensive and more complete system for the purpose of enhancing people's CT competencies.

1.2. Computational Thinking and Programming Education in Higher Education

Learners of diverse backgrounds learn CT with various purposes and learners' target objectives considering the proficiency level also differ accordingly on learner's level of proficiency. Therefore, it is important to know what the necessary skills are to be developed in higher education, what proficiency level of CT is expected for people from distinct domains and in what way should CT be incorporated in different domains in Higher Education. Programming education is frequently used for fostering CT in higher education; visual programming in Scratch and Alice as well as text programming in Python, C, C++, Java have been used for teaching CT in K-12 settings as well as in Higher Education settings [39-40]. However, it remains a controversial topic whether everyone should learn to code. For example, Shein acclaimed that "Not everyone needs coding skills but learning how to think like a programmer can be useful in many disciplines" [35]. Therefore, it would be important to study the role of Programming Education.

CT and programming skills are closely interlinked and are both challenging for novice learners [29, 30]. However, a significant drop-out rate can be found in programming education on novice learners due to distinct difficulties students meet during their learning process [31]. Pane et al. [32] found that the ability to solve problems using programming skills so that the solution can be transformed and executed by computing agents does not come naturally for learners in CS studies. Additionally, studies also suggest that the absence of strategic tools can lead to deficient performance in learning to program [33-34].

To overcome these challenges, it is necessary to conduct research in both programming skills and CT skills and the relationship between them, which has been seldom researched.

Through qualitative and quantitative analyses, Selby [38] built a preliminary model to reveal connections between CT skills and programming activities using Bloom's taxonomy. However, it does not demonstrate in detail how CT can be measured in programming. Thus, it is necessary to carry out studies on how to empower students to use CT as a strategic tool for programming and gain CT knowledge through learning to program.

In brief, the following questions should be studied regarding CT and Programming Education in Higher Education:

- What skills are necessary for students in different domains in Higher Education?
- What is the role of Programming Education for students from different domains in Higher Education?
- How are programming skills and CT skills related and how to foster CT skills via programming?

1.3. Formative Assessment and Feedback Generation

Novice programmers who are new to programming are faced with challenges such as misunderstanding the programming concepts, misusing the language syntax, and understanding poorly the feedback generated from the interpreter or compiler [31]. Alternative approaches to overcome these issues can be, for instance, enhancing teachers' pedagogical content knowledge, developing more effective didactic strategies, using formative assessment to provide feedback.
Assessment and feedback are essential elements in different learning theories which are used to assist students in the learning process [41]. Assessment is presented in two categories in general, formative assessment and summative assessment. Formative assessment is defined as assessment for learning while summative assessment as assessment of learning [42]. Formative assessment generally consists of teacher observation, conventional assessment, oral presentation and so on. According to Paul Black & Dylan Wiliam [43], formative assessment remains incomplete until it has resulted in feedback and action on the part of the instructor and/or learner. Therefore, a formative assessment is all about feedback. According to Hattie and Timperley [45], feedback is one of the most crucial factors for efficient learning.

The development of formative assessment in Programming Education is still at an early age though there has been lots of research on intelligent tutoring systems which assess students' solutions in recent years. Computer-assisted learning environments provide the opportunity to automate the assessment and considerable work has been conducted to assess works in STEM disciplines automatically [44]. In terms of Programming Education, Grover [42], in the Raspberry Pi Foundation Computing Education Research Seminar, strived to promote the concept of formative assessment in CS for K-12. In contrast, no existing study explicitly facilitates formative assessment either in computing education or in Programming Education specifically in Higher Education.

While most of the assessments being conducted on CT and Programming Education are summative, there is some work that applies formative assessment measures in their implementations. These implementations focused on merely part of programming education and none of these works incorporated CT into programming education, making them infeasible for assessing CT in Programming Education. Meanwhile, some studies aimed at supporting students in learning to program, mostly in the form of automated assessment systems and intelligent tutoring systems for programming exercises. In their literature review, Keuning et al. [47] reported that most of the elaborate feedback provided by the systems reviewed focus on the identification of mistakes and no further suggestions on how to proceed and fix the problem. This, however, can impede students from enhancing their performance according to the feedback model defined by Hattie and Timperley [45]. Therefore, it is necessary to conduct research to explore formative assessment of CT in Programming Education in order to assist students in the learning process to enhance their CT in Programming Education.

With the vision to make CT skills more accessible and tangible in the context of Programming Education for learners from different domains, this project aims to develop formative assessment components to improve students' performance in learning to program and gaining CT skills.

2. Theoretical Background

To address the questions mentioned in the last section, theories on formative assessment and theoretical models of CT and Programming Education are crucial. Therefore, they are being investigated to ensure the reliability of the conduction of the project. CT and Programming Education will be first introduced with a focus on Brennan and Resnick's operational framework [16] and Bloom's taxonomy on Programming Education. Then follows theory for formative assessment and feedback models

with a focus on Hattie's feedback model and the theory of formative assessment from Paul Black & Dylan Wiliam [43]. The theories are identified as the backbone in the implementation of this project.

2.1. Computational thinking and programming education (Bloom's Taxonomy)

Although there are no agreed-upon operational and theoretical definitions, definitions given by researchers and educators share the same elements in their definition. Wing defined CT operationally with the concepts of abstraction and automation [9]. Having components used in Wing's definition, Barr and Stephenson [46] included also problem decomposition, algorithmic thinking, data collection, analysis and representation and simulation to define CT. Similarly, Selby's definition of CT consists of abstraction, decomposition, generalization, evaluation and algorithmic design [38]. Four main components of CT can be identified from existing definitions: problem decomposition, pattern recognition, abstraction and algorithmic design.

Deriving from the main CT components, Brennan and Resnick [38] proposed an operational framework of CT which is frequently used in CT studies and the framework relates quite close to programming concepts and skills. Three dimensions constitute the framework: computational concepts, computational practices and computational perspectives. These components are recognizable in other disciplines and practices as well, which is consistent with Denning's description CT: it is nothing new, it is the way of thinking about the world shaped by the current technologies [50]. This framework considers elements comprehensively from both a knowledge perspective and a psychology perspective and it is a framework that can be practically used for setting learning objectives, designing pedagogical contents, and assessing students' performance [48].

CT concepts and CT practices involved in this framework [48] are some of the indicators that measure CT competences through programming concepts and practices. Studies have been conducted to map programming skills and CT skills as well as using Bloom's taxonomy and SOLO taxonomy to differentiate various levels of cognition for both CT and programming skills [36, 37]. Assessment of CT through assessing Scratch codes in Dr. Scratch with the framework presented by Brennan [38] is an example of how CT can be matched in Programming Education [49]. Selby [39] developed a model which discovers the relationship between CT skills and programming activities by using Bloom's taxonomy. This model can serve as the backbone in fostering CT via programming and vice versa.

2.2. Formative assessment and feedback generation

Having a CT framework and a model which maps CT to programming using cognitive levels in Bloom's taxonomy is insufficient for this project as the aim of this project is to enhance students' CT skills via formative assessment. Therefore, this subsection will introduce theories on formative assessment and models for generating feedback as formative assessment is said to be all about feedback [42].

Assessment is identified as one of the fundamental elements in all learning theories in education [41]. Formative assessment is defined as assessment for learning, and it is expected to result in feedback and action on the part of the instructor and/or learner if formative assessment is implemented. Thus, feedback is crucial in formative assessment, which is consistent with "Feedback plays a crucial role in learning" [27].

The efficiency of the feedback is influenced by the kind of formative feedback provided and the learner characteristics. Under the definition given by Boud and Molloy [51], feedback is formative, and it can be used to improve learners' performance. Another type of feedback is summative feedback, typically consists of grades or percentage of evaluation, which informs the learner about the performance. However, this type of feedback is usually too superficial to be useful for learners. Therefore, formative feedback is of more importance for the purpose of improving learning.

Different definitions and models have been investigated regarding feedback generation both in general and for studies in specific

domains. Boud and Molloy define feedback as a process in which the learners improve their work with the given information which presents the discrepancy and similarities between learners' work and the expected standards [51]. Hattie and Timperley [45] described a model for feedback which is also in a formative way. The model aims to answer learners' questions about where they are, how they should proceed and where they should arrive. In this model, feedback is categorized into "task level", "process level", "self-regulation level" and "self-level", with findings indicating self-level the most ineffective one.

Having a model of feedback is insufficient for generating the most effective feedback for learners, extra facets should be considered when generating feedback. In Le and Pinkwart's work [52], programming exercises supported in learning environments were categorized into three classes according to the level of ill-definedness of the programming problem. As Hattie and Timperley [45] pointed out that feedback should target students at appropriate levels, it would be necessary to also consider Narciss's [53] categorization of feedback in computer-assisted learning environments according to the aspects of the instructional context. Narciss [53] has identified eight types of feedback components, five of them are elaborated feedback component and are intended to "improve learner's performance": knowledge about task constraints (KTC), knowledge about concepts (KC), knowledge about mistakes (KM), knowledge about how to proceed (KH) and knowledge about Meta-cognition (KMC). Combining the context to be assessed, the type of exercises to be assessed and the feedback level to provide, a strategy for generating feedback can be devised.

In sum, this project will first focus on identification of the need for CT and the role of Programming Education in different disciplines. Then, the focus will be shifted to the measurement of CT skills and programming skills and the relationship between these two sets of skills. Based on studies conducted, this project will then explore feedback generation and develop feedback generation strategies to promote CT for students from different domains and enhance their performance in CT skills and programming skills. The following definitions will be used for the remainder of the proposal:

- CT competencies: according to Brennan's framework, CT competencies refer to CT concepts, CT practices and CT perspectives.
- Programming skills: including conceptual knowledge, syntactic knowledge and strategic knowledge and programming style.
- Indicators for CT skills and programming skills: Any features, instruments that provide a sign or a signal of CT competence and programming skills.
- Formative assessment: A kind of assessment which provides feedback to the learner and it is an assessment for learning.

3. Research Questions

The research will be guided by the following research questions:

RQ1. How are CT skills and programming skills being conceptualised and measured?

1. What are indicators and assessment methods for CT competence and programming skills?
2. What systems and domains are using the indicators and assessments for CT competence and programming skills?
3. How to evaluate the validity of the indicators/assessment?

After collecting the indicators for CT competencies and assessment methods, techniques used for formative assessment and feedback generation and the effect of feedback should be investigated to provide the basis for design feedback generation strategies. Therefore, the second research question is:

RQ2. How should feedback be provided to support developing CT skills and programming skills, and how should formative assessment be implemented in this process?

1. What formative assessment and feedback generation strategies are used for the development of programming skills and CT competence?
2. What are the effects of different types of feedback on motivation, learning gain, and CT performance?
3. What empirical knowledge has been established regarding the effect of providing feedback on the development of CT competence and programming skills??

4. How to use formative assessment and generate feedback to support the development of CT and programming skills?

Based on the results obtained by answering the questions above, the next step is to contextualize the feedback and thus employ formative assessments for learners from different educational backgrounds. To achieve the goal, the following questions should be studied:

RQ3. How can Programming Education and learning of CT be contextualised and embedded in different educational domains?

1. How important are links between curricular tasks and CT skills?
2. What role can transfer learning play in the contextualisation of CT?
3. What are the means to contextualise and embed CT learning in different domains?
4. What is the impact of contextualised teaching of CT skills on student motivation and understanding?

4. Design and Methods

The research is organized in four phases. In the first phase a desktop research/systematic literature review will be used to identify relevant works to get an overview of state-of-the-art regarding the topic being studied in this project - formative assessment for supporting students from different disciplines in the process of learning CT in the context of Programming Education in Higher Education. The following factors will be identified in this phase: indicators used for assessment and assessment methods for CT in Programming Education; formative assessment and feedback generation; empirical experiences of CT in different domains. The indicators identified in the first phase can then be used to develop an assessment model for CT in the context of Programming Education and a CT dashboard to present learners' progress and CT level. Exploratory research in the form of formative studies will be employed in this phase. Phase three will focus on the development of strategies for feedback generation and formative assessment based on the assessment model and the CT dashboard built in phase two. In the last phase, an integrated study will be conducted to evaluate the tool developed and refine the system according to different needs from people of different backgrounds. In parallel, design and development of the formative assessment tool for CT in the context of Programming Education will be carried out throughout the lifecycle of the project. In addition to that, the design, development and testing of the prototype will be iteratively proceeded. The plan for the workflow is provided in the diagram shown in Figure 1 (in the Appendix.

Phase 1 Desktop research - Literature review

In this phase, a systematic literature review will be conducted to get a holistic overview of formative assessments for supporting learners in different disciplines to learn CT in the context of Programming Education. This process will follow the PRISMA statements and the PRISMA diagram, including defining research questions, collecting literature, screening, checking eligibility of the literature, data extraction and analysis of extracted results. RQ1.1, RQ1.2, RQ2.1 and RQ3.1 will be addressed in this phase. The outcome of this phase will be indicators used for assessment and assessment methods for CT in Programming Education; a comprehensive overview of formative assessment and feedback generation; empirical experiences of CT in different domains.

Phase 2 Exploratory research/ Formative studies - Build up the assessment model and a CT Dashboard

This phase begins with interviews with different target groups. The aim of the interview is to identify the necessity of CT skills and the role of Programming Education for learners with diverse backgrounds. In combination with the indicators and assessment methods identified in Phase 1, assessment models can then be prototyped according to the result from a qualitative analysis of the interviews. The interviews should also clarify the embedding of the CT skills in the different study contexts and the relevance for student and educators' goals in the different curricula. According to the goals and models a CT dashboard will be developed. To ensure the usability of the models and the CT dashboard, a usability study will be conducted in a programming course for students and the models and CT dashboard will be refined accordingly. Once the usability of the model is verified, quasi experimental studies will then come into play to examine the effect of using the assessment model and CT dashboard.

In this phase, RQ1.3, RQ2.2 and RQ2.3 will be studied, and an assessment model based on the indicators and assessment methods found in Phase 2 will be developed. This will include a participatory design and prototype of a CT dashboard. The design and the development of the models and the CT dashboard will proceed iteratively.

Phase 3 Develop feedback and formative assessment based on assessment model and CT Dashboard

This phase will focus on addressing RQ2.4, which is about developing proper feedback generation strategy to present to students their CT competencies and programming skills based on the strategies for feedback generation and formative assessment identified in Phase 1 and the CT assessment prototype and CT dashboard developed in Phase 2. Formative studies will be conducted to iteratively develop the feedback generation model. Student models will be identified in this phase by using data such as analysis of students' code, student's competence profile and analysis of students' performance. At the end of this phase, strategies for providing feedback and formative assessment should be identified.

Phase 4 Evaluation - Integrated study on the developed formative assessment tool

The result from Phase 3 will provide a basis to address RQ3.2 to RQ3.4 in this phase. Considering the factors which are important in adapting feedback for learners from different domains identified in phase 1, RQ3.2 to RQ 3.4 will be addressed by conducting an integrated study which includes both case studies and an evaluation study to contextualise the model developed and embed it into different educational domains and verify the validity and the effectiveness of the designed system. This integrated study aims to evaluate the tool developed and refine the system according to diverse needs from people of different backgrounds such that CT can be promoted further to a wider audience.

5. Acknowledgements

Xiaoling Zhang: Conceptualization, Methodology, Data Collection, Analysis, Writing - Original Draft, Writing – Review & Edit, Visualization, Resources

Marcus Specht: Conceptualization, Methodology, Writing – Review & Edit

This work is a part of a PhD project funded by Center for Education and Learning at Leiden-Erasmus-Delft Universities (LDE-CEL).

6. References

[1] Dillenbourg, P. (2016). The Evolution of Research on Digital Education. Int J Artif Intell Educ 26, 544–560 (2016). https://doi.org/10.1007/s40593-016-0106-z.

[2] Duggal, R., Brindle, I., Bagenal, J. (2018, January 15). Digital healthcare: regulating the revolution. doi: https://doi.org/10.1136/bmj.k6.

[3] Holzer, M., Kim, Seang-Tae. (2006) Digital Governance in Municipalities Worldwide (2005) : A Longitudinal Assessment of Municipal Websites Throughout the World. United Nations Public Administration Network. http://unpan1.un.org/intradoc/groups/public/documents/aspa/unpan022839.pdf.

[4] Schwab,K., Sala-i-Martín, X. (2013).The Global Competitiveness Report 2013–2014: Full Data Edition. URI: http://hdl.handle.net/11146/223.

[5] D Barr, J Harrison, L Conery. (2011). Computational thinking: A digital age skill for everyone. Learning & Leading with Technology, 2011 - ERIC.

[6] Francisco José García-Peñalvo, Antònio José Mendes,Exploring the computational thinking effects in pre-university education,Computers in Human Behavior,Volume 80,2018, Pages 407-411, ISSN 0747-5632, https://doi.org/10.1016/j.chb.2017.12.005.

[7] Anita JUŠKEVIČIENĖ, Valentina DAGIENĖ. (2018). Computational Thinking Relationship with Digital Competence.

[8] Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, 1980.

[9] Wing, J. Computational thinking, Commun. ACM 49, 3 (Mar. 2006), 33–35.

[10] Wing, J. 2008. Computational thinking and thinking about computing. Philosophical Transactions of The Royal Society A, 366, 3717-3725.

[11] Wing, J. 2011. Research Notebook: Computational Thinking - What and Why?

The Link. Pittsburgh, PA: Carneige Mellon.
[12] Computer Science Teachers Association Task Force. 2011. K–12 Computer Science Standards, New York, ACM.
[13] Hu, C. 2011. Computational thinking: what it might mean and what we might do about it. Proceedings of the 16th annual joint conference on Innovation and technology in computer science education. Darmstadt, Germany: ACM.
[14] Guzdial, M. 2011. A Definition of Computational Thinking from Jeannette Wing. Computing Education Blog [Online]. Available from: http://computinged.wordpress.com/2011/03/22/a-definition-of-computational-thinking-from-jeanette-wing/ [Accessed 30-11-2020].
[15] Guzdial, M. 2012. A nice definition of computational thinking, including risks and cyber-security. Computing Education Blog [Online]. Available from: http://computinged.wordpress.com/2012/04/06/a-nice-definition-of-computational-thinking-including-risks-and- cyber-security/ [Accessed 30-11-2020].
[16] Brennan, K., & Resnick, M. 2012, New frameworks for studying and assessing the development of computational thinking. Paper presented at the Annual Meeting of the American Educational Research Association, Vancouver, BC.
[17] Z. Berkaliev et al., Initiating a programmatic assessment report, Primus 24 (2014), 403–420. https://doi.org/10.1080/10511970.2014.893939.
[18] P. Curzon et al., Developing computational thinking in the classroom: a framework, Comput. Sch. (2014), http://eprints.soton.ac.uk/369594/10/DevelopingComputationalThinkingInTheClassroomaFramework.pdf.
[19] C. Evia, M. R. Sharp, and M. A. Perez-Quinones, Teaching structured authoring and DITA through rhetorical and computational thinking, IEEE Trans. Prof. Commun. 58 (2015), 328–343. https://doi.org/10.1109/TPC.2016.2516639.
[20] S. Grover and R. Pea, Computational thinking in K-12: A review of the state of the field, Educ. Res. 42 (2013), 38–43. https://doi.org/10.3102/0013189X12463051.
[21] K. Jaipal-jamani and C. Angeli, Effect of robotics on elementary preservice teachers' self-efficacy, Sci. Learn. Comput. Think. (2017), 175–192. https://doi.org/10.1007/s10956-016-9663-z.
[22] Y. Jeon and T. Kim.The effects of the computational thinking-based programming class on the computer learning attitude of non-major students in the teacher training college, J. Theor.Appl. Inf. Technol. 95 (2017), 4330–4339.
[23] B. Kim, T. Kim, and J. Kim, Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution, J. Educ. Comput. Res. 49 (2013), 437–459. https://doi.org/10.2190/EC.49.4.b
[24] Y. Lan, Exploration on database teaching based on computational thinking, Bol. Tec. Bull. 55 (2017), 363–370. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85038935540&partnerID=40&md5=010e1ae2c24dbf9eaa18b000844ece22.
[25] C. Mouza et al., Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK), Australas. J. Educ. Technol. 33 (2017). https://doi.org/10.14742/ajet.3521.
[26] Lyon, J. & Magana, A. (2020). Computational thinking in higher education: A review of literature. Computer Applications in Engineering Education. 28. 10.1002/cae.22295.
[27] A. L. S. O. de Araujo, W. L. Andrade, and D. D. S. Guerrero, “A systematic mapping study on assessing computational thinking abilities,” in Frontiers in Education Conference (FIE), 2016 IEEE. IEEE, 2016, pp. 1–9.
[28] Francisco José García-Peñalvo, Antònio José Mendes. Exploring computational thinking effects in pre-university education. Computers in Human Behavior. Volume 80. 2018. Pages 407-411. ISSN 0747-5632. https://doi.org/10.1016/j.chb.2017.12.005.
[29] M. Tedre. Many paths to computational thinking. Paper presented at the TACCLE

3 final conference, Brussels, Belgium (2017).

[30] P. Denning, M Tedre, P Yongpradit. Misconceptions about computer science. Communications of the ACM. Volume 60, Number 3 (2017), Pages 31-33.

[31] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. ACM Trans. Comput. Educ. 18, 1, Article 1 (December 2017), 24 pages. DOI :https://doi.org/10.1145/3077618.

[32] Pane, J. F., Ratanamahatana, C. A. & MYERS, B. A. 2001. Studying the language and structure in non-programmers' solutions to programming problems. International Journal of Human-Computer Studies, 54, 237-264.

[33] Robins, A., Rountree, J. & Rountree, N. 2003. Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13, 137 - 172.

[34] Saknini, V. & Hazzan, O. 2008. Reducing Abstraction in High School Computer Science Education: The Case of Definition, Implementation, and Use of Abstract Data Types. J. Educ. Resour. Comput., 8, 1-13.

[35] Esther Shein. 2014. Should everybody learn to code? Commun. ACM 57, 2 (February 2014), 16–18. DOI:https://doi.org/10.1145/2557447.

[36] Susana Masapanta-Carrión and J. Ángel Velázquez-Iturbide. 2018. A Systematic Review of the Use of Bloom's Taxonomy in Computer Science Education. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). Association for Computing Machinery, New York, NY, USA, 441–446. DOI:https://doi.org/10.1145/3159450.3159491.

[37] David Ginat and Eti Menashe. 2015. SOLO Taxonomy for Assessing Novices' Algorithmic Design. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). Association for Computing Machinery, New York, NY, USA, 452–457. DOI:https://doi.org/10.1145/2676723.2677311.

[38] Cynthia C. Selby. 2012. Promoting computational thinking with programming. In Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE '12). Association for Computing Machinery, New York, NY, USA, 74–77. DOI:https://doi.org/10.1145/2481449.2481466.

[39] Cynthia C. Selby. 2014. How Can Teaching of Programming Be Used to Enhance Computational Thinking Skills? University of Southampton, Faculty of Social and Human Sciences, PhD Thesis, pagination.

[40] Valerie J. Shute, Chen Sun, Jodi Asbell-Clarke. Demystifying computational thinking. Educational Research Review. Volume 22. 2017. Pages 142-158. ISSN 1747-938X. https://doi.org/10.1016/j.edurev.2017.09.003.

[41] Dale H. Schunk. (2012). Learning theories an educational perspective sixth edition.

[42] Shuchi Grover. (2020). Formative Assessment for Students in CS Classrooms, https://www.youtube.com/watch?v=0ZuSqsJQRFg&feature=emb_title. [last access: 2020-11-16]

[43] Black, P., Wiliam, D. Developing the theory of formative assessment. Educ Asse Eval Acc 21, 5 (2009). https://doi.org/10.1007/s11092-008-9068-5.

[44] Barana, A., Conte, A., Fioravera, M., Marchisio, M., Rabellino, S.; A model of formative automatic assessment and interactive feedback for STEM. In: Proceedings of 2018 IEEE 42nd Annual Computer Software and Applications Conference, pp. 1016–1025. IEEE Computer Society Conference Publishing Services (CPS), Tokyo, Japan (2018).

[45] Hattie, J., & Timperley, H. (2007). The power of feedback. Review of educational research, 77(1), 81-112.

[46] V. Barr and C. Stephenson, Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? ACM Inroads 2 (2011), 48–54.

[47] Keuning, H., Jeuring, J. T., & Heeren, B. J. (2019). A Systematic Literature Review of Automated Feedback Generation for

Programming Exercises. ACM Transactions on Computing Education, 19(1), [3]. https://doi.org/10.1145/3231711

[48] Yeni, S., & Hermans, F. (2019). Design of CoTAS: Automated computational thinking assessment system. In TACKLE 2019: 2nd Systems of Assessments for Computational Thinking Learning workshop: Proceedings of the 2nd Systems of Assessments for Computational Thinking Learning workshop (TACKLE 2019) (Vol. 2434). (CEUR Workshop Proceedings).

[49] Jesús Moreno-León and Gregorio Robles. 2015. Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15). Association for Computing Machinery, New York, NY, USA, 132–133. DOI:https://doi.org/10.1145/2818314.2818338.

[50] Peter J. Denning and Matti Tedre. (2019). Computational Thinking.

[51] Boud, D., & Molloy, E. K. (2013). Feedback in higher and professional education: Understanding it and doing it well. Routledge. https://doi.org/10.4324/9780203074336.

[52] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. 2015. Learning Feedback in Intelligent Tutoring Systems. Künstliche Intelligenz 29, 4 (2015), 413–418.

[53] Susanne Narciss. 2008. Feedback strategies for interactive learning tasks. Handbook ofresearch on educational communications and technology (2008), 125–144.

7. Appendix

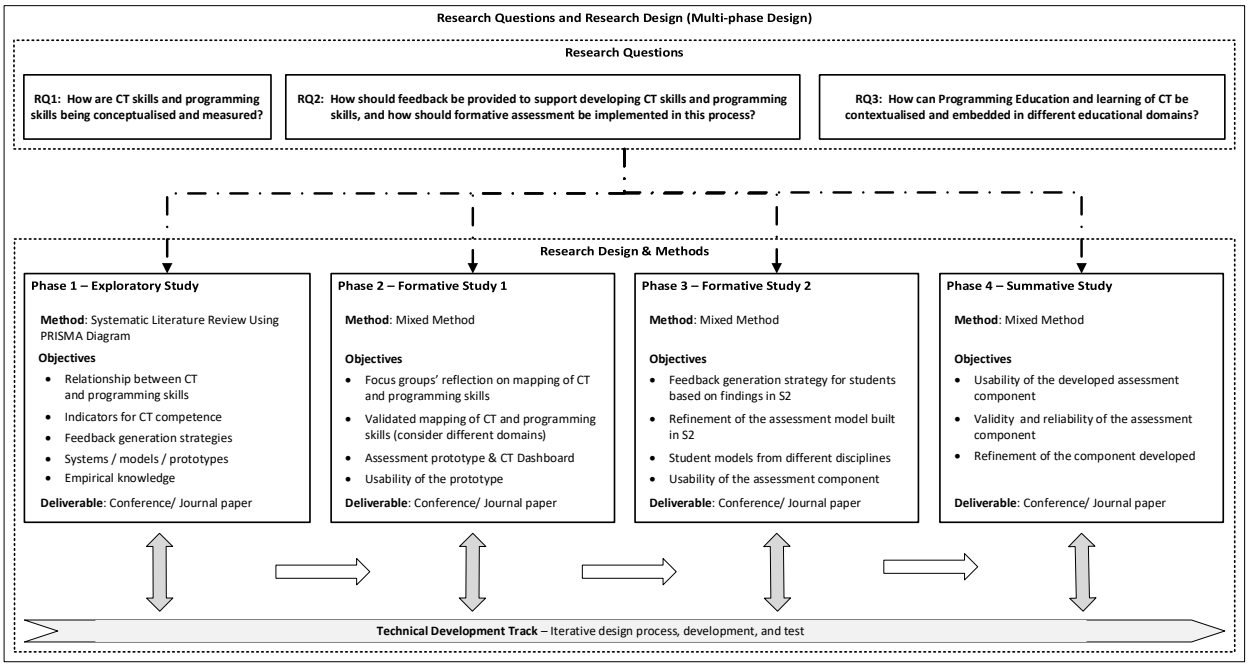


Figure 1. The whole PhD research plan with the main goals presented for each year. The system for providing feedback will be iteratively designed and developed throughout the project lifecycle.