 DOI: 10.34641/ctestem.2022.466

The TACTIDE EU STEM project : TeAching Computational Thinking with Digital dEvices

Marc Jansen[1,3], Nardie Fanchamps[2], Marcelo Milrad[3], Marcus Specht[4,] Ali Hamidi[3]

[1] University of Applied Sciences Ruhr West, Germany

[2] Open University Nederland, The Netherlands

[3] Linnaeus University, Sweden

[4]Delft University of Technology, The Netherlands

marc.jansen@hs-ruhrwest.de, nardie.fanchamps@ou.nl, marcelo.milrad@lnu.se, m.m.specht@tudelft.nl, ali.hamidi@lnu.se

ABSTRACT

One major challenge the educational community is facing relates to how to effectively integrate computational thinking (CT) concepts and ideas into a particular school curriculum. Acquiring CT-skills by means of STEM offers rich opportunities within students´ education which may lead to learning gains. Previous research has shown that, to maximize the appeal and potential of CT learning environments, a precondition must be set first. The materials used must invite problem-based, inquiry-based and self-discovery learning, must be used without creating misconceptions and, above all, must give students the opportunity to acquire knowledge that can be directly transferred to everyday practice in an accessible manner. All the above puts demands on teachers who carry out learning and teaching in these environments. The EU funded TACTIDE project has tried to incorporate relevant curricular components into a coherent task, implementing assignments and challenges across different subjects and curricula of three different European countries. Based on the analysis of each national curricula, common topics have been identified and sub-scenarios have been developed. These sub-scenarios have been conceived to promote the integration between the topics mediated by CT. To achieve this objective, a greenhouse scenario has been conceptualized and designed towards teaching CT, by the use of microcontrollers such as the BBC micro:bit and the Calliope Mini, as an overarching STEM-topic. Using available sub-scenarios, a Moodle-course for teachers was developed for daily school activities to which other subjects in the core curriculum were interconnected in order to integrate CT skills and abilities. Scalability across different school levels and heterogeneous groups of learners, especially focusing prior knowledge, have been considered important design elements.

KEYWORDS

Computational thinking, teachers, curriculum, STEM, learning scenarios

1. INTRODUCTION

The emergence and application of new technologies in everyday life requires specific knowledge and skills. Through STEM-education these skills could be acquired, as STEM educational scenarios offer opportunities for an integrated subject matter approach combined with the use of digital tools. Previous research has shown that for maximizing the attractiveness and possibilities of novel learning environments, a precondition must be set first. The used materials should invite problem-based, inquiry-based and self-discovery learning, should be used without creating misconceptions and, above all, should give students the opportunity to acquire knowledge that can be transferred directly to everyday practices in an easily accessible way. The latest sets demands on teachers providing these environments. New forms of STEM education also more and more stress the importance of students´ digital literacy and the development of computational thinking skills.

Computational thinking is a way of approaching and solving problems using concepts from computer science and primarily involves the ability to reason, plan and solve problems (Wing, 2006). It refers to operationalised concepts such as parallel thinking, pattern recognition, completion, debugging, sequencing, and abstract reasoning that are needed to systematically approach a problem (Basawapatna, Koh, Repenning, Webb, & Marshall, 2011; Lee et al., 2011). Computational thinking involves the process in which problem definition, solution expression and implementation with evaluation recur in the process of programming (Yadav, Hong, & Stephenson, 2016), and can contribute to understanding and solving complex programming problems (Voskoglou & Buckley, 2012). Computational thinking encompasses in general two main directions: computational concepts and computational practices (Grover & Pea, 2017). CT concepts include: logic & logical thinking, algorithms & algorithmic thinking, patterns & pattern recognition, abstraction & generalization, evaluation, and automation. CT practices refers to: problem decomposition, creating computational artefacts, testing & debugging, iterative refinement, collaboration, and creativity.

A challenging discussion for promoting computational thinking education is how the acquisition of these skills can be integrated in the curriculum and how other subjects in the core of the curriculum are linked to this. The TACTIDE project has explored how to integrate relevant curricular components into a coherent educational activity by linking them to the creation of a greenhouse which integrates tasks and challenges from different subjects and across the curricula of three different European countries.

2. STATE OF THE ART

The application of programmable tangibles and artefacts is a playful integration of developing problem-solving skills and computational thinking. The application of robotics in STEM-contexts requires students to apply logical reasoning

in programming environments. It also demands systematic thinking, for the right choice of sensors and actuators, to program a robot that can anticipate the physical environment (Fanchamps, et al., 2019). Programmable robots harbour the potential to develop computational thinking skills because the visually observable output makes the results of programming interventions concrete and tangible (Catlin & Woollard, 2014; Sapounidis, Demetriadis, & Stamelos, 2015; Slangen, 2016). When users can test the effect of programming actions in authentic situations, they are better able to critically review and assess their programming actions (Slangen, Keulen, & Gravemeijer, 2011). Because programmable robots can be used to obtain immediate feedback on the consequences of code, they function as direct manipulation environments (DMEs) (Jonassen, 2006; Rekimoto, 2000). Direct manipulation environments (DMEs) involve users in constructing mental models of phenomena. Users are challenged to directly manipulate parameters and variables in the environment. Many DMEs reinforce the sense of operating with concrete objects. DMEs allow users to reason, predict, and hypothesise, analyse and test through active participation (Jonassen, 2006; Slangen et al., 2011). Robots are concrete and physical DMEs and can be controlled by programming using actuators and sensors (Jonassen, 2006; Rekimoto, 2000). They provide a potentially rich context for learning, understanding and practising programming and robotics concepts and for developing (general) problem-solving and computational thinking skills (CT).

The ability to anticipate changing environmental conditions by means of sensor observations and the computer program to be constructed, is a strategy to obtain an increased proficiency in computational requirements (Kim & Kim, 2003). To anticipate changes in the environment by means of sensor use requires a different program than performing programming tasks in an unchanging, predictable environment. By making use of sense-reason-act (SRA) programming, a programmed artefact or simulation of reality can react to changes in its surroundings. SRA-programming can be described as the process in which external, sensor-based observations (sense) are entered into a microprocessor, so that these observations can be compared with internal pre-set conditions (reason) which infers executing desired programming actions (act) (Fanchamps et al., 2019). The ability to anticipate changing conditions in the task design by means of sensor-based observations requires a different programming approach in comparison to linear solutions (Dragone et al., 2005). SRA-programming involves the functional understanding and application of complex programming concepts such as nested loops, conditionals and functions (Jansen, Kohnen-Vacs, Otero & Milrad, 2018; Slangen, 2016). Being able to respond to changes in the task design by means of SRA-programming can ensure that users' computational thinking ability develops at a higher level (Riedmiller & Gabel, 2007).

To teach computational thinking, teachers and designers should develop curricula to prepare and further enhance children's computational thinking competencies (Fanchamps et al., 2020). This by reinforcing the application of CT concepts and practices in the classroom. For learners, practicing and applying computational thinking concepts and approaches in contexts both within and outside of programming is an important prerequisite for acquiring skills that are required and applicable in other school subjects. For teachers this demands an adapted and tuned pedagogy to be able to integrate the cognitive and affective dimensions of deeper learning underlying computational thinking. For a thorough implementation of STEM in education and curriculum integration, the methodology of subject integration is proposed (Kelley & Knowles, 2016).

3. TEACHING CT ACROSS DIFFERENT SUBJECTS

To develop an integrated approach across different subjects and the implementation in the different partner countries, the curricula (grades 6-9) from Germany, The Netherlands, and Sweden have been compared to see which potential subjects could be integrated into a multidisciplinary course in CT concepts in order to enforce learning. To achieve this, an identification of the different courses was created after which for each country a tick was placed to show the presence of these subjects. In different stages of education, the amount and selected subjects may differ. Therefore, the age group of 12-15 year-olds was chosen as the starting point. In the analysis of the learning outcomes for this age group the common subjects and objectives between the different countries have been identified. This led to common objectives in mathematics, biology and physics. From the courses analyzed languages, creative, and social studies did not meet the requirements for creating a CT course. These courses could not be selected as they are not widely supported within all three countries. This left the STEM (Science Technology Engineering and Mathematics) subjects as the final choice.

Despite the macro-level strategy adopted by the different countries which are involved in this study in terms of how to integrate CT into the curriculum, there are numerous possibilities to put CT into practice. One possible opportunity is the integration of several subject matters within the context of designing and implementing a greenhouse scenario in connection to STEM. Indeed, our designed scenario bonds CT and STEM in a context where physical and digital tools are integrating and interacting with each other. Designing, creating, and experimenting in areas that are interesting for students are three crucial elements for such integration (Brennan & Resnick, 2012; Zerega et al, 2021). The students involved in these learning activities will be encouraged to use their creativity to design a portable indoors greenhouse. They will conceptualize their ideas and then create their designing thoughts in practice. Moreover, they will learn and share their knowledge during both the design, implementation, and experimentation phases. Moreover, students will learn some central environmental facts by observing, thinking, experimenting, and testing them during the various activities.

This approach could also provide the opportunity for critical thinking and developing problem-solving skills in two ways: 1) in the design and building phase of the greenhouse from 3D design and modeling to physical construction, and 2) in programming the BBC micro:bit that is mounted in the

greenhouse together with different sensors. For those who are new to programming, it provides an opportunity to take the first steps into the field and to learn basic concepts of a programming language by using visual block coding in an authentic setting. The focus is not on learning coding only but also on developing computational thinking skills and physical computing through coding the microcontrollers and sensors. In general, by using electrical components and simple electronics in combination with environmental science and programming, different aspects of STEM are explored during this proposed activity. The learning activities and learning modules for the integrated course have been collected in an online course which can be used by teachers to run the course in the actual classroom.

4. IMPLEMENTATION

A greenhouse, which can be operated via a programmable micro controller, has been used as a concrete elaboration of a DME. The application of CT skills is requested to program and control this greenhouse in a functional way. This greenhouse allows for science education in the form of growing plants and what the plants require for their growth, fostering our precondition that a corresponding learning scenario should invite problem-based, inquiry-based and self-discovery learning. Using a microcontroller, a program can be created to measure and provide the needs of these plants. Engineering would be covered by allowing the students to create their own model for the greenhouse. Finally, mathematics could be covered when, for example, the student needs to make calculations for how long it would take to cool down the greenhouse using the fan.

The goal of this project is to provide an appealing and challenging learning scenario in which participants build an automated portable greenhouse, in which programmable microcontrollers like BBC micro:bit and/or the Calliope are used in order to monitor and control important parameters such as temperature, humidity, soil moisture to ensure a suitable environment for plants. The design and construction of the greenhouse was carried out in 2 steps:

- Sketching the mini greenhouse by using a browser-based 3D design and modeling tool (Tinkercad app), as shown in figure 1. This will help to imagine the final product.
- Building the greenhouse using reusable straws, clear vinyl/PVC, and glue as shown in figure 2.

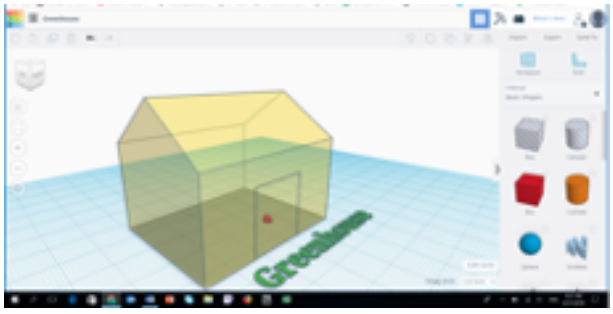

Figure 1. 3D Tinkercad design greenhouse.

Figure 2. Vinyl/PVC build greenhouse.

To control the above stated parameters physical computing devices and sensors are combined with visual programming. The coding part of the greenhouse project is done using the MakeCode editor (BBC micro:bit, 2019) as described in figure 3 depicted in the figure below:

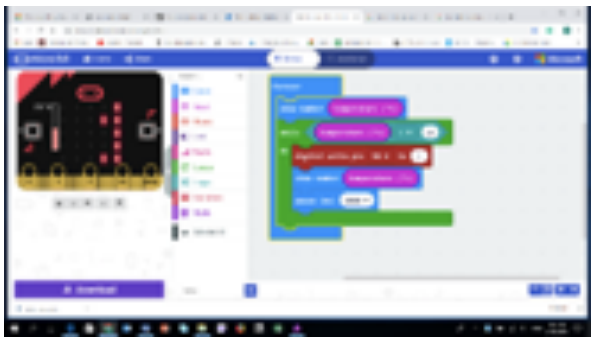

Figure 3. Coding BBC micro:bit.

After connecting a temperature/humidity sensor, a fan blade, a water pump, and a mini servo the following two scenarios could be realized:

Controlling temperature: According to the temperature measured the students can program the microcontroller to turn the fan when the temperature exceeds above a defined level. Similarly, the fan will be turned off automatically when the temperature comes below the defined level. This scenario includes specific tasks for reading sensor data from a thermometer, using conditionals to decide on different levels of temperature as also the use of loops for continuously measuring and controlling a fan until the temperature is in the target zone.

Controlling humidity: If the humidity comes either less than the defined limit or more than the optimum range, the fan, and the water pump will automatically turn on as well as rotating the mini servo to open the window on top of our greenhouse for better aeration.

In both scenarios, additional coding features and dimensions of computational thinking can be used such as creating functions to support working with patterns and problem decomposition.

5. SUMMARY AND OUTLOOK

In this paper we have presented the initial results of our efforts related to the conceptualization and development of a couple of educational scenarios that promote the integration between different school curriculum topics mediated by CT. The choice of content has been guided by

finding relevant subjects that emerged from the analysis of school curriculum in three European countries. Educational materials, including lessons plans, code examples, use of sensors and microcontrollers and video tutorials have been developed. Due to the Covid situation we have experimented since March 2020, the educational ideas and scenarios described above could not be validated with schools in these three countries.

Aspects that were planned to be assessed during the evaluation with students were related to conducting qualitative analyzes of the overall learning experience as well as the acceptance of this kind of learning scenarios in the schools.

It is important to emphasize that the TPACK framework (Mishra and Koehler, 2006) has been used to guide the development of the scenarios. Referring to the shortage in studies that focus on pedagogical aspects of teachers' CT development (Ottenbreit-Leftwich, et al., 2021), our work pays attention to aspects related to pedagogical content knowledge (PCK) as well as technological content knowledge (TCK) of the TPACK framework.

6. REFERENCES

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education.

Brennan, K. and Resnick, M., 2012, April. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).

Catlin, D., & Woollard, J. (2014). *Educational robots and computational thinking.* Paper presented at the Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education, Padova, Italy.

Dragone, M., O'Donoghue, R., Leonard, J. J., O'Hare, G., Duffy, B., Patrikalakis, A., & Leederkerken, J. (2005). Robot soccer anywhere: achieving persistent autonomous navigation, mapping, and object vision tracking in dynamic environments. Paper presented at the Opto-Ireland 2005: Photonic Engineering, Dublin, Ireland.

Fanchamps, N., Slangen, L., Hennissen, P., & Specht, M. (2019). The Influence of SRA Programming on Algorithmic Thinking and Self-Efficacy Using Lego Robotics in Two Types of Instruction. International *Journal of Technology and Design Education*, 1-20. doi:10.1007/s10798-019-09559-9

Fanchamps, N., Specht, M., Hennissen, P., & Slangen, L. (2020). The Effect of Teacher Interventions and SRA Robot Programming on the Development of Computational Thinking. Paper presented at the International Conference on Computational Thinking Education 2020, Hongkong.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, 19.

Jansen, M., Kohen-Vacs, D., Otero, N., Milrad, M. (2018). A Complementary View for Better Understanding the Term Computational Thinking. In: Proceedings of the International Conference on Computational Thinking Education. 2018.

Jonassen, D. H. (2006). *Modeling with technology: Mindtools for conceptual change*. Upper Saddle River, New Jersey, USA: Pearson Merrill Prentice Hall

Kelley, T. R., & Knowles, J. G. (2016). A conceptual framework for integrated STEM education. *International Journal of STEM Education*, *3*(1), 1-11. doi.org/10.1186/s40594-016-0046-z

Kim, D.-H., & Kim, J.-H. (2003). A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. *Robotics and Autonomous Systems*, 42(1), 17-30. doi:10.1016/S0921-8890(02)00311-1

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers' college record*, 108(6), 1017-1054.

Ottenbreit-Leftwich, A., Yadav, A., & Mouza, C. (2021). Preparing the Next Generation of Teachers: Revamping Teacher Education for the 21st Century. In *Computational Thinking in Education*, 151-171, Routledge.

Riedmiller, M., & Gabel, T. (2007). On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. Paper presented at the 2007 IEEE Symposium on Computational Intelligence and Games.

Rekimoto, J. (2000). *Multiple-computer user interfaces: beyond the desktop direct manipulation environments.* Paper presented at the Conference on Human Factors in Computing Systems, The Hague, Netherlands.

Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing, 19*(1), 225-237. doi:10.1007/s00779-014-0774-3

Voskoglou, M. G., & Buckley, S. (2012). Problem solving and computational thinking in a learning environment. *Egyptian Computer Science Journal*, 36(4), 18.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35. doi:10.1145/1118178.1118215

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends,* 60, 565-568. doi:10.1007/s11528-016-0087-7

Zerega, R., Hamidi. A., Tavajoh, S., & Milrad, M. (2021). A Co-design Approach for Developing Computational Thinking Skills in Connection to STEM Related Curriculum in Swedish Schools. In Proceedings of the *5th APSCE International Computational Thinking and STEM in Education Conference 2021*. Singapore: National Institute of Education, pp 144-147