Title Towards using computational modeling in learning of physical computing: An observational study in Singapore schools
Authors Peter, Sen-Kee SEOW, Bimlesh WADHWA, Zhao-Xiong LIM and Chee-Kit LOOI
Source Proceedings of International Conference on Computational Thinking Education 2020 (pp. 21-26).
Organised by The Education University of Hong Kong

Copyright © 2020 The Education University of Hong Kong

This document may be used for private study or research purpose only. This document or any part of it may not be duplicated and/or distributed without permission of the copyright owner.

The Singapore Copyright Act applies to the use of this document.

This document was archived with permission from the copyright owner.
Towards using Computational Modeling in learning of Physical Computing – An Observational Study in Singapore Schools

Peter, Sen-Kee SEOW¹, Bimlesh WADHWA², Zhao-Xiong LIM³, Chee-Kit LOOI⁴
¹National University of Singapore, Singapore
²National Institute of Education, Nanyang Technological University, Singapore
³, National Institute of Education, Nanyang Technological University, Singapore
⁴Peter.seow@nie.edu.sg, bimlesh@nus.edu.sg, zhaoxiong.lim@nie.edu.sg, cheeakit.looi@nie.edu.sg

ABSTRACT
Coding for students is no longer just constrained to software and screen-based text and graphics. Students today use programmable sensors and microprocessors to solve the problems around them. The purpose of this research is to understand how students conceptualize problems and implement solutions with physical computing. Our study is driven by the following: 1) find out what Computational Thinking (CT) competencies, specifically abstraction, decomposition and algorithmic thinking, can be developed by students and 2) to what level students develop these competencies in carrying out physical computing projects. We closely observe how 41 Grade 7 students developed solutions for problems they identify in the physical world around them. Through doing so, we explore how powerful ideas of CT play a role in a project-approach to physical computing. We believe open-ended exploration through a project-approach in physical computing should reinforce practices where CT skills can grow and flourish. Our findings show that much of students’ interaction with sensors and devices is at pre-CT level, where students simply use pre-existing code fragments or templates. As students gain skills and confidence, they can be explicitly guided to develop CT skills with new projects of their own design justifying their choices. We strongly believe that Computational Modeling (CM) could help students develop their CT skills e.g. abstraction, decomposition, and algorithmic approach much more than the minimally guided syntax driven teaching approaches.

KEYWORDS
computational thinking, computational models, abstraction, physical computing, K-12

1. INTRODUCTION
Physical computing, an emerging approach to learning computing, teaches students about coding and computational thinking through hands-on activities with sensors using small computing boards like the micro:bit (Rogers, et al., 2017). In Singapore, primary school students are introduced to coding through the Digital Maker programme with the micro:bit (TNP, 2019).

The micro:bit is a pocket-sized physical computing device that can be input with various codes. The device is designed to be visually appealing and tactile, affordable, easy to use, interactive, and extensible. The board has a built-in display, buttons, motion detection, temperature and light sensing. It can be programmed using a desktop PC, laptop or tablet running one of several different operating system web-based programming environments: a visual block-based editor, Python or JavaScript.

Despite the ease in using the micro:bit to code, it is not certain that physical computing will actually improve students' understanding of computational thinking (CT). It is therefore important for educators to explore the question on “what and how do students develop CT competencies when they use physical computing devices to interact with the physical world?” With this as an over-arching research question, we set out to design our observational study in Singapore schools. Our aim here is to 1) find out what CT competencies, specifically abstraction, decomposition and algorithmic thinking, should be developed by students and 2) to what level, do students develop these in carrying out physical computing projects. Furthermore, we want to find out if students used any conceptual or computational modeling before or while carrying out their physical computing projects. We believe our observations would help us partially answer our over-arching research question.

The project-approach to physical computing, an often used pedagogy in schools, serves as an open-ended exploratory approach to examine the computational thinking competencies that students should learn. We observed that among many other factors that inhibit the development of CT skills, the inherent complexity of problem and solution space could overwhelm students. Additionally, the cognitive load in designing and developing their solutions could also hinder them in their learning. Therefore, we propose gentle scaffolds for developing a sound conceptual model, followed by guided Computational Modeling (CM) for overall CT skills development.

2. RELATED WORK
Our study is informed by the ideas of Computational Thinking (Papert, 1972; Wing, 2008), Computational Models (Aho, 2012; Denning, 2017; Calder, et. al., 2018), learning computing models (Sentence, Waite & Kallia, 2019) and Physical Computing (O’Sullivan & Igoe, 2004). Seymour Papert (1972) described CT as a mental skill children develop from practicing programming. In a 2006 paper, Jeannette Wing (2006) catalyzed a ‘CT for all’ (p.33) movement. It has been debated since then if CT makes better problem solvers or if practice of coding can help develop CT skills, with claims that everyone can benefit by CT not yet being fully substantiated by studies (Guzdial, Kay, Norris, Soloway, 2019). Many definitions and role of CT in computing as well as in other fields, and overlap of CT and computing have followed.

Denning (2017), in his viewpoint on CT, finds the absence of computational models in the post-2006 CT definitions as a mistake. He points that key ingredients of CT e.g. abstraction, data representations and decomposition are used, in order to get a model to accomplish certain task. He encourages teachers to take note of Aho’s reflection about
computation as “a process defined in terms of an underlying model of computation” (p. 832) and CT as “the thought processes involved in formulating problems so that their solutions can be represented as computational steps” (p. 832). Aho suggests the use of the term ‘computation’ in conjunction with a well-defined model whose semantics is clear and which matches the problem being investigated. He added that one could use CT skills to devise computation models.

There is a growing emphasis on teaching computing since the idea of CT was proposed by Wing. Countries such as the United Kingdom have mandated the integration of Computing and Computational Thinking into the National curriculum at all levels. Japan is making learning computing compulsory for elementary and higher education. Introducing computing has expanded from using screen-based tools such as Scratch to physical computing such as the micro:bit. In physical computing, students interact with the world through the use of sensors as input and controllers as output of computing devices. Computation is done on the data from the input sensors like buttons or temperature sensor to drive the controllers such as motors or LED lights.

For students learning to code, they need to deal with the complexity of knowing what data they need from the environment, how to use the sensors to collect the data, how the data is used for computation, what output needs to be created and how it should be used. This complexity could overwhelm students in designing and developing their solutions. In many approaches to introducing coding, students are taught using physical computing without introducing CT skills. The assumption is that students would learn CT skills as a result of the learning coding through physical computing. In the pre-CT stage, students may encounter difficulties in implementing physical computing solutions without using CT skills, such as automating machine to interact with the physical world (Fig 1). For example, students need to know how to acquire data from the environment, process to compute the data, and output to the world. Teaching CT skills explicitly to students can help students to implement their solutions better as shown in Fig 2. The teaching of CT skills can bridge some of the difficulties students face in learning coding and implementing solutions with Physical computing.

Figure 1. Illustration of the difficulties of implementing Physical Computing in the classroom without involvement of CT

PRIMM (Sentence, Waite, & Kallia, 2019), a framework for teaching programming, focuses on students talking about how and why programs work before they tackle writing their own programs. The first element of PRIMM i.e. Predict, is about students discussing and predicting what a program might do, drawing and writing out what they think will be the output in order to develop the vocabulary they need to talk about the program. We believe such vocabulary development should extend to CM, which is an important step in the process of formulating problems to be solved.

3. CONTEXT OF STUDY
To understand how CT is applied in learning of physical computing through the use of micro:bit in coding, the research team observed the micro:bit training sessions and interdisciplinary project work lessons of Grade 7 comprising of 41 students divided into 10 groups, in a local neighbourhood school, over a period of 4 ½ months. The purpose of the study was to understand how students conceptualised the problems and implemented the solutions with physical computing. Additionally, the research aimed to 1) find out what CT competencies, specifically abstraction, decomposition and algorithmic thinking, students should develop and 2) to what level, do students develop these in carrying out physical computing projects. Students find it exciting when they see their projects come to life. Physical computing is therefore very engaging that helps them understand how things work in the real world.

The students followed a project-approach to develop physical computing solutions using micro:bit. During the micro:bit training sessions, the students are first introduced to both the basic and intermediate technical aspects of the micro:bit board and makecode editor, where they carry out block coding. Thereafter, they are introduced to the development environment of the ‘makecode editor’, an online visual block-based coding programme, where they could develop their codes. Their solutions were expected to incorporate sensors to capture data occurring from everyday phenomena such as surrounding temperature or sound. The process that students undertake in designing their projects using the below-mentioned flowchart.

The entire training sessions and interdisciplinary project work lessons seek to complement the Applied Learning Programme (ALP) in Robotics and Programming run by the school, which aims to empower students with the technological and thinking skills that will enable them to be innovative and empathetic members of the community. (MOE, 2019)
They sought to teach students real-life curriculum skills, namely Abstraction, Decomposition, and Algorithmic Thinking. Sessions were filmed and recorded in both audio and video format.

The researchers observed the 10 hours training sessions, order to accustom students to the micro:bit coding to make the solution more effective.

Additionally, we observed the students as they design, code and implement their physical computing projects in Lessons 5 to 7 (see Table 1). We looked out for CT skills application in specific milestones of problem formulation, initial design, implementation, and demonstration as they carry out their projects. The purpose of the activity was to understand how students use abstraction, decomposition and algorithmic thinking, while conceptualising the problems and implementing the solutions with physical computing.

For data collection, we selected two groups for more detailed observation (See Table 2). We followed these two groups as they developed their projects and enquired them on their actions and decisions. These two groups were selected based on the complexity of their projects and recommendations by their teachers because they were able to articulate their ideas clearly compared to their peers. We recorded the presentations they made to classmates on their ideas and solutions. After their presentation, we interviewed the group members and archived their codes for analysis.

Table 2. Projects of the two groups of students observed

5. ANALYSIS AND FINDINGS

To evaluate students on their application of their CT skills, we developed a set of rubrics for CT skills. The rubrics were developed from our literature review of the CT skills and we also obtained feedback from practitioners on the levels of the rubrics and the skills. For this work, we focused in observing three CT skills, namely Abstraction, Decomposition, and Algorithmic Thinking (See Table 3).

In our analysis, we observed the two groups closely as they developed their projects with physical computing. Our observations of two groups and the projects that they worked on in the following paragraphs.

Group A worked on a problem of a sensor-operated doorlock that would open upon motion detection of a contactless card. The students were queried about the algorithmic steps and meaning of the codes in the micro:bit block. The research team hinted to students about thinking logically about the codes found in the block and figure out which blocks can be matched together to form the required codes. We observed that the students lacked the necessary knowledge on the type of data and sensor to read the contactless card. As a result, the students realized that they had to change the sensor from a card scanner to a digital keypad with numbers connected to the classroom door, as the use of a card scanner had been deemed unfeasible. Even with the change, they could not implement the use a digital keypad with the micro:bit.

Table 3. Rubrics for Classroom Observation

| ABSTRACTION – to choose the right amount of detail for the problem to be modeled |
|----------------------------------|-------------------------------------------------------------------|
| Beginner: | Able to identify and choose relevant data and information for the model and solving the problem |
| Intermediate: | Beginner + identify relevant data and from multiple sources to integrate for developing possible CMs |
| Advanced: | Intermediate + physically represent through modelling and interact with relevant data and information for the |
Group B worked on the problem of noise detection in the classroom, which would send an alert to the teacher once the noise threshold is reached. The team had difficulty in conceptualizing their solution with the micro:bit. They recognized that they needed a sound sensor to detect sound from the physical environment. They were able to test the sensor input and simulating an alert to the teacher by pairing two micro:bits. They however lacked the systematic knowledge. For example, they did not think deep enough on where to best place the sensor to capture the noise accurately or how sound travels could affect the coding and prototyping of their project. It showed an inadequate mental model, and therefore an inadequate conceptual model of the problem and solution i.e. how the sensors interact with the physical world, and e.g. sound travels by waves and where they place sensors matters.

We analysed the transcripts of the interview made with the students to evaluate the decisions the made with regard to the implementation of their solution. We identified how their abstraction, decomposition and algorithmic thinking are demonstrated from the interview data, as explained by the students (See Table 4). This was made in reference to the rubrics we developed.

Table 4. Interview Transcript of students in Group A demonstrating the skill of Algorithmic Thinking

<table>
<thead>
<tr>
<th>R/Researcher</th>
<th>Dialogue</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S/Student</td>
<td>How did you solve the problem?</td>
<td>Students decide to give remote micro:bit to teacher indication of real time noise level</td>
</tr>
<tr>
<td>S/Student [06:38]</td>
<td>We decided to place, instead of alerting the teacher when it hits the second level, we decided to show the teacher the level all the time</td>
<td>Abstraction – choose the relevant data</td>
</tr>
<tr>
<td>S/Student</td>
<td>Oh show, show the teacher the level all the time</td>
<td>Decomposition – break down the problem into smaller part to show the information to the teacher</td>
</tr>
</tbody>
</table>

The codes developed for the solution and the created artefact comprising of the micro:bit board with sensors were analysed for students algorithmic thinking, decomposition and abstraction competencies (See Figure 4). For example, the group with the sound sensor used one micro:bit to read the sound level from Pin 1 and control a LED at Pin 0. The micro:bit will send the value of the sound level to another micro:bit through radio. The students designed such that the remote micro:bit will be carried by the teacher and will notify the teacher if the level exceeds the noise. The students calibrated and tested the actual noise level in their classroom that they deemed as noisy. This was the level they chose as the trigger to notify the teacher. During testing, the students noticed that there was a delay in sending the value to the teacher’s micro:bit and the noise level reading was not accurate. They did not have time to solve these issues.

Figure 4. Example of code (Sound Sensor) done by students using makecode editor in the project design

We analysed the groups’ work process in developing the solution, their completed artefact solution and codes, and the interview transcripts. The analysis from the sources were triangulated and compared to the rubrics we developed. Results showed that most were at best able to achieve only the beginner level of the CT skills (see Table 5). However, we noted that the acquiring of these skills progressed over time and towards the end, most students became more competent in them as they engaged in more programming.

From our observations, we posit that students have difficulties in designing a computing solution due to their rudimentary CT skills. At the end, the students managed to build a prototype of their solution but experienced challenges in abstracting the vital data required for the solution in the initial stages. This affected their choice of sensors to use as input to their solution and difficulties in
thinking algorithmically on the computation to obtain the automated output. Referring to Table 4, we believe both groups worked at pre-CT stage (see Figure 1). The above observations are specific to a few projects and students, and generalizations would require more studies with more students and diverse settings.

Table 5. Results of skills demonstrated by the two groups during the observation

<table>
<thead>
<tr>
<th>CT skills</th>
<th>Group A (Level achieved)</th>
<th>Group B (Level achieved)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>Beginner</td>
<td>Intermediate</td>
</tr>
<tr>
<td>Decomposition</td>
<td>Beginner</td>
<td>Beginner</td>
</tr>
<tr>
<td>Algorithmic Thinking</td>
<td>Beginner (students who programmed)</td>
<td>Beginner (students who programmed)</td>
</tr>
</tbody>
</table>

6. DISCUSSION AND RECOMMENDATIONS

Computational Modeling (CM) as per one of the established definitions (Calder, et. al., 2018) can help us “translate observations into an anticipation of future events, act as a testbed for ideas, extract value from data and ask questions about behaviours. The answers are then used to understand, design, manage and predict the workings of complex systems and processes, from public policy to autonomous systems.” (p. 2)

Computational Thinking (CT), on the other hand is a generalized problem-solving process that can be applied to a wide variety of problems. The steps of CT include formulating a problem in a way that enables us to use a machine to solve it. The machine here refers to computer and other devices. In the process, data and concepts are abstracted and analysed and algorithms are developed for automating a solution.

We believe, that CT definition is not explicit about modelling, i.e. representation of abstracted data and concepts before algorithms are developed. Here in this study, we first observe if indeed classrooms have modelling incorporated in the CT lessons. Our findings show that students attempt to directly code or develop algorithm once they have understood the problem. They do not develop any models or use any tools to represent data or concepts.

Today, visual block-based visual programming platforms such as Scratch, Blockly are popular vehicles for programming sensors and delivering CT. Even though students are quick to pick up the programming constructs, conceptual difficulties pertaining to problem and solution space, and developing CT skills e.g. abstraction, decomposition and algorithmic skills, are often evident.

From our observation of the work of both groups, we surmise that students face difficulties in designing a computing solution due to the missing explicit CT exposure and almost non-existent CM. They have challenges in abstracting the vital data required for the solution and thinking algorithmically on the computation to obtain the desired output.

For students, owning an idea serves as a motivation to learn. We observed that students identified a problem or innovation they wanted to pursue. We found that though students were engaged in the maker-rich environments, they did not move to thinking computationally and solving problems. Much of their interaction with sensors and devices is superficial. This is inferred through our interactions with students. When we discussed with students about for example how sensors were working or how transmission of data or signal was from one to another device, we did not find them confident of their knowledge of hardware beyond what they were using it for.

However, when we, for example, introduced input-process-output model, their understanding of the project seemed better. They could explain the project better to another researcher from our own group as well as to their teachers later. We would want them to develop higher order design skills through physical computing, not just coding. They should understand the iterative nature of finding a solution and testing. Open-ended exploration through the project-approach in physical computing should reinforce practices where CT skills grow and flourish.

Additionally, based on our observations, students have difficulty in starting the implementation due to their lack of CT and CM in the pre-CT stage. Reasons for such gaps are mainly due to the inherent complexity of the problem that they are trying to solve, as well as integrating different components of the solution involving use of sensors, collection of data, computation of data and automating the solution. To scaffold their learning, it is important that they are guided through developing a conceptual model, such as CT (abstraction and algorithmic skills) and CM (See Figure 4), leading them eventually towards the CT+CM stage.

We propose that a CM phase could act as a glue from understanding problem to the coding activity (see Figure 4) for students to formulate their problems in computational steps (Denning, 2017). Execution of computational models could be seen as controlling and automating the machine to solve the problem computationally. We believe focused modeling activities could help students develop their CT skills e.g. abstraction, decomposition, and algorithmic approach much more than the minimally guided syntax driven teaching approaches.

Based on our observations we suggest concrete steps that can be taken to support the development of computational thinking. We believe a project-approach through physical computing provides an excellent maker-platform, in which students are provided with the opportunity to evaluate and manipulate underlying abstractions and mechanisms. It gives ample scope of developing CT skills namely abstraction, decomposition, and algorithmic thinking.
We propose a triad-model for effective and systematic development of CT skills. This model describes a pattern of engagement (see Figure 5). It is based on the premise that computational modeling promotes the acquisition and development of CT among students. At the 'pre-CT' level, students are simply coders. For example, they code using pre-existing code fragments or templates, and acquire coding skills through a series of iterative refinements. New skills and understandings are developed over time and they begin to code with increasing levels of sophistication. This does develop an understanding of at least a subset of the abstraction contained within the problem and solution. We observed that most of the students in our study operate at the pre-CT level. As students gain skills and confidence, they can be explicitly guided to develop CT skills with new projects of their own design justifying their choices. At this CT level, all three key aspects of computational thinking: abstraction, decomposition and algorithmic thinking, come into play. We observed two groups of students partially acquiring CT skills when we explicitly made them think about specific issues about their problem or solution. We strongly believe that Computational Modeling (CM) could act as an effective medium to develop computational thinking skills especially in the context of physical computing. We propose another level in our framework labelled 'CT+CM' i.e. using Computational Modeling (CM) as a medium to develop CT skills.

Here are our recommendations for effective delivery of CT skills with CM based on this study:

- Design of a thinking style workshop that could help students to develop and strengthen their mental model about the problem. It is an important and essential that students have the relevant vocabulary of the problem domain, and systemic thinking before attempting to formulate a solution.

- Guided team brainstorming sessions could help students develop conceptual models for the solution they propose. Developing a sound conceptual model at the team level could help each individual member to strengthen his/her mental model, and sync well with team before implementing the solution.

- Gentle scaffolds could be introduced, e.g. graphic organisers for the above, to ease students into developing Computation Models. CM could be the glue that connects a conceptual solution and actual code.

7. REFERENCES


Rogers, Y. et al. (2017). From the BBC micro to micro:bit and Beyond. interactions 24(2), 74–77.


