Title	Researching and developing pedagogies using unplugged and computational thinking approaches for teaching computing in the schools
Author(s)	Looi Chee Kit, Wu Longkai, Peter Seow and Wendy Huang

OER FINAL REPORT SERIES

The OER Final Report series includes final reports from funds managed by Office of Education Research, National Institute of Education, Nanyang Technological University.

Reports are submitted as part of the funding review process and intended for the funding agency, local schools and educators, teacher educators, policymakers, and education scholars. They do not take the place of scholarly, peer-reviewed articles but report on the background, procedures, and major findings of the project.*

This study was funded by Singapore Ministry of Education (MOE) under the Education Research Funding Programme (OER 04/16 LCK) and administered by National Institute of Education (NIE), Nanyang Technological University, Singapore. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Singapore MOE and NIE.

*In some cases reports show coloured font or highlights. These are an artifact of the review process and not intended to have any special weight or meaning within the report itself.

Version: 17 Jul 2020, NIE OER

EDUCATION RESEARCH FUNDING PROGRAMME

PROJECT CLOSURE REPORT

Researching and developing pedagogies using unplugged and computational thinking approaches for teaching computing in the schools

By

Looi Chee Kit, Wu Longkai, Peter Seow, Wendy Huang

National Institute of Education
Singapore

EXECUTIVE SUMMARY

INTRODUCTION/BACKGROUND

In 2017, Singapore's Ministry of Education implemented a new GCE 'O' Level Computing curriculum. The new curriculum is a distinct shift from the teaching students on the use of software technology to the development of Computational Thinking skills and programming competencies. Computing thinking skills are associated with problem solving, reasoning and logic skills that all students should develop. As Singapore moves to implement a new curriculum with a greater emphasis on the development of computational thinking and programming, the following are some of the challenges that must be addressed:

1. Teachers' Pedagogical Knowledge in teaching Computing
2. Teachers' Competency and Knowledge on Computational Thinking

STATEMENT OF PROBLEMS

This project has a focus on using and integrating the unplugged approach as introductory activities for teaching computing as a pedagogy. It focuses on helping students to understand concepts in Computational Thinking. The approach also fits very well to the teaching and learning environment in a typical secondary school classroom. We worked with the teachers from collaborating schools to design and co-design unplugged activities, observed how they enacted the lessons in the classroom. This would help us to understand how teachers interpret computational thinking and adapt the unplugged approaches with their teaching practice. Also, we would like to study students' learning outcomes as a result of the teaching.

The existing practice and research of unplugged teaching has the following problems:
1. There is no systematic integration. Among the many topics in computing, there are not many topics that match unplugged activities.
2. For the first-line teachers, the available public accessible resources do not help much. It can only be used when they encounter related topics. Even if there are corresponding resources on the Internet, many teachers are not keen on adopting unplugged teaching methods, due to the time and effort needed to prepare and to enact the lessons.
3. The existing unplugged teaching resources are designed with the goal of mobilizing students' interest and engagement, and more in-depth practice and research in transiting from teaching with unplugged methods to programming is needed.

PURPOSE OF STUDY

The purposes of this proposed research study are the following:

- Develop and evaluate pedagogies linked to teaching CT. We introduce teaching unplugged as an effective student-centered approach to introducing computing concepts without the use of computers, and then we design follow-up activities and pedagogies that move students forward in the crucial computational experiences.
- Assess the effect on teachers. Teachers' pedagogical content knowledge will be assessed to understand the level they started with, and the level they would have attained after the workshops and teaching in class. Classroom observations will be held to study the teachers' enactment of computing lessons. We want to understand the territory of teachers' dispositions for, attitudes toward and stereotypes concerning CT and Computing.
- Assess the effect on students. Students' work will be analysed to assess their level of comprehension and application of computing concepts, and this will be done through prior experience surveys, pre-post computing perceptions survey, pre-post computing tests,

quizzes and computing assignments. These are steps towards developing an assessment framework for CT.

PARTICIPANTS

Four secondary schools, about eight teachers and a hundred and thirty students are involved in the study.

METHODOLOGY / DESIGN

The research methodology is design-based research (DBR) (Brown, 1992) using mixed methods to gather and study information on how teachers teach the learning activities, and how and what students learn. The research objectives stated above imply the central theme of the methodology to result in greater understanding of a learning ecology by designing its elements and by anticipating how these elements function together to support learning. The research methods include: Survey/Questionnaires; Interviews; Achievement Tests; Classroom Observations; Video recordings; Teaching/Learning Artefacts; Document analysis.

FINDINGS / RESULTS

There is a range of unplugged activities which can be introduced as part of computing lessons to firstly, engage and motivate students, and secondly, with good teacher facilitation, help students develop appropriate mental models or what is called notional machines of computing concepts and algorithms. New activities can be designed by teachers that fit well with the computing curriculum. When students engage in these activities, they develop computational skills such as decomposition, pattern recognition, abstraction and algorithmic thinking.

Implications for practice: Unplugged pedagogies add to the repertoire of teaching methods used by teachers to engage students cognitively in the computing tasks.

Implications for policy and research: That some of computational thinking can be taught by unplugged pedagogies suggests that we can reach out to a bigger base of students without the distractions of getting the resources and infrastructure for computer programming.

Learning gains (for studies involving intervention): Students learning an algorithm like sorting using an unplugged approach demonstrate the learning of computational thinking skills that are beyond students learning the algorithm using a more conventional approach (Looi et al, 2018).

Proposed Follow-up Activities: More workshops and courses to introduce unplugged pedagogies to computing teachers; increase the uptake of these pedagogies.

CONTRIBUTIONS

In terms of contributions to NIE Progammes and Practice, we have developed some curricula and activities for teaching teachers about computational thinking education. In terms of contributions to practice, teachers of computing have access to resources prepared by us on the unplugged pedagogies.

CONCLUSION

There is a range of unplugged activities which can be introduced as part of computing lessons to firstly, engage and motivate students, and secondly, with good teacher facilitation, help students develop appropriate mental models or what is called notional machines of computing concepts and algorithms. New activities can be designed by teachers that fit well with the computing curriculum. When students engage in these activities, they develop computational skills such as decomposition, pattern recognition, abstraction and algorithmic thinking.

ACKNOWLEDGEMENTS

This study was funded by the Education Research Funding Programme, National Institute of Education (NIE), Nanyang Technological University, Singapore, project no. OER 04/16 LCK. The views expressed in this paper are the author's and do not necessarily represent the views of NIE.

KEYWORDS

Computational Thinking; Computing, Unplugged Activities; Unplugged Pedagogies

Researching and developing pedagogies using unplugged and computational thinking approaches for teaching computing in the schools

Looi Chee Kit, Wu Longkai, Peter Seow, Wendy Huang

National Institute of Education

INTRODUCTION/BACKGROUND

In 2017, Singapore's Ministry of Education implemented a new GCE 'O' Level Computing curriculum. The new curriculum is a distinct shift from the teaching students on the use of software technology to the development of Computational Thinking skills and programming competencies. Computing thinking skills are associated with problem solving, reasoning and logic skills that all students should develop. As Singapore moves to implement a new curriculum with a greater emphasis on the development of computational thinking and programming, the following are some of the challenges that must be addressed:

1. Teachers' Pedagogical Knowledge in teaching Computing
2. Teachers' Competency and Knowledge on Computational Thinking

STATEMENT OF PROBLEMS (I.E., JUSTIFICATION FOR THE STUDY)

This project has a focus on using the unplugged approach as introductory activities for teaching computing as a pedagogy. These activities help students to understand the concepts in Computational Thinking. The approach also fits very well to the teaching and learning environment in a typical secondary school classroom. We worked closely with the teachers to design or co-design activities using the plugged approaches. We also want to understand how teachers interpret computational thinking and adapt the unplugged approaches with their teaching practice. Also, we would like to study students' learning outcomes as a result of the teaching. Through a series of iterations studying the linkages

between the PD workshops, classroom enactment and student outcomes, we hope to refine the professional development model to develop teachers' pedagogical knowledge and capacity to teach computational thinking in the project.

The existing practice and research of unplugged teaching has the following problems:

1. There is no systematic integration. Among the many topics in computing, there are not many topics that match unplugged activities.
2. For the first-line teachers, the available public accessible resources do not help much. It can only be used when they encounter related topics. Even if there are corresponding resources on the Internet, many teachers are not keen on adopting unplugged teaching methods, due to the time and effort needed to prepare and to enact the lessons.
3. The existing unplugged teaching resources are designed with the goal of mobilizing students' interest and engagement, and more in-depth practice and research in transiting from teaching with unplugged methods to programming is needed.

PURPOSE OF STUDY (INCLUDING RESEARCH QUESTIONS AND/OR OBJECTIVES)

The purposes of this proposed research study are the following:

- Develop and evaluate pedagogies linked to teaching CT. We introduce teaching unplugged as an effective student-centered approach to introducing computing concepts without the use of computers, and then we design follow-up activities and pedagogies that move students forward in the crucial computational experiences.
- Assess the effect on teachers. Teachers' pedagogical content knowledge will be assessed to understand the level they started with, and the level they would have attained after the workshops and teaching in class. Classroom observations will be held to study the teachers' enactment of computing lessons. We want to understand the territory of teachers' dispositions for, attitudes toward and stereotypes concerning CT and Computing.

- Assess the effect on students. Students' work will be analysed to assess their level of comprehension and application of computing concepts, and this will be done through prior experience surveys, pre-post computing perceptions survey, pre-post computing tests, quizzes and computing assignments. These are steps towards developing an assessment framework for CT.

PARTICIPANTS

Four secondary schools, about eight teachers and a hundred and thirty students are involved in the study.

METHODOLOGY/DESIGN

The research methodology is design-based research (DBR) (Brown, 1992) using mixed methods to gather and study information on how teachers teach the learning activities, and how and what students learn. The research objectives stated above imply the central theme of the methodology to result in greater understanding of a learning ecology by designing its elements and by anticipating how these elements function together to support learning. The research methods include: Survey/Questionnaires; Interviews; Achievement Tests; Classroom Observations; Video recordings; Teaching/Learning Artefacts; Document analysis.

FINDINGS / RESULTS

DESIGN OF UNPLUGGED ACTIVITIES AND ASSOCIATED RESEARCH

We designed as well as co-designed with the teachers a number of unplugged activities. Below summarizes the findings of the enactment of some of these activities. In our study, we worked with computing teachers at four secondary schools. They were open to trying new

ways of teaching, especially using activities that encourage active learning. Initially, we identified possible matches between the topics in the computing syllabus and CS Unplugged activities. This was not a trivial process as we realized that there were few authentic connections. The ones that we selected to test in classrooms included:

1. Mouse Programming: give directional instructions to a physical device shaped like a mouse to reach certain locations on an orthogonal grid; used to practice algorithmic thinking and decomposition and learn basic programming constructs including sequencing and loops

2. The Box Variable and swapping: an unplugged activity selected for addressing a difficult programming concept for novices

3. Binary numbers: directly aligned to syllabus learning goals, but needed to be adapted to be more challenging for secondary school level students

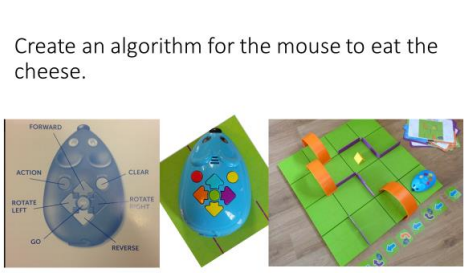
4. Sorting: not in the syllabus but used to practice algorithmic thinking and writing flowcharts and pseudocode

5. Potato Pirates: using a card game developed by a local start-up to teach loop constructs in programming

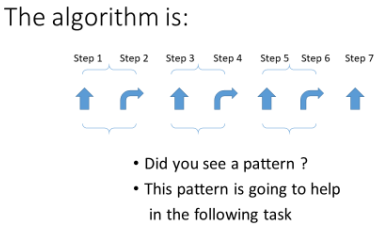
In this report, we briefly shared findings from the first three unplugged activities.

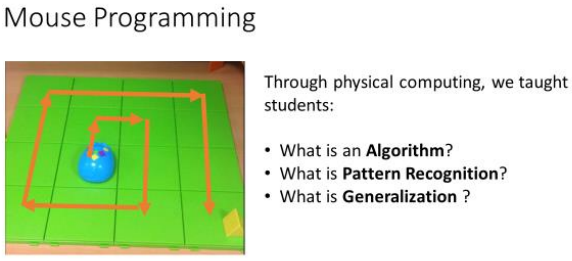
1. Mouse programming

What is an Algorithm?: In the very early stage in Sec 3, a programmable mouse on a mat is used to introduce students to the concept of an algorithm as a series of instructions for navigating to carry out a task.



Students expressed their enjoyment of the activity after the class. We also used this activity to start to introduce the concept of patterns to the students, such as:





2. Box Variables and Swapping

Knowing the variable concept through unplugged activity in swapping of variables: students worked in groups to design their solution for swapping the values of two variables. We found out in the early stages of their understanding of the variable in computing, they develop ingenious and yet inaccurate or incomplete solutions to the swapping, indicating that their mental model of the variable is still developing.

We administered a pre-test and a post-test. On the question of "Is x = x+y a valid statement in Python?", in the Pre-test: 15 out of 23 says NO whereas in the Post-test: 3 out of 23 says NO, indicating a learning gain. On the question of "A variable still holds its original value after an assignment", in the Pre-test, 11 out of 23 says YES while in the Post-test: 2 out of 23 says YES.

Students need to have appropriate mental models of the variable with generalization, meaning they need to be able to write statements that swap any two variables, not two values. The range of student misconceptions include: Value assignment to variable is like an equation in math; "Value" is a limited commodity as exemplified by this example of a group's work: Colour 1 = "red"; Colour 2 = "green"; Colour 3 = "red" (create one more copy of read, before putting Colour 2 = Colour 1, as in the first group that presented its solution during the class). The following table shows the full set of results on a class of 24 Sec 3 students.

Correct Rate	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Pre-Test	34.80%	17.40%	17.40%	78.30%	69.60%	13%	8.70%	52.20%	65.20%	26.10%	21.70%	30.40%
Post-Test	87%	78.30%	87%	95.70%	82.60%	69.60%	47.80%	91.30%	34.80%	30.40%	43.50%	30.40%

2. Binary Numbers

School	Level	First formal lesson on binary numbers?	Programming task?
School #1	Sec 4	Yes	Yes
School #2	Sec 3	Yes	No
School #3	Sec 4	No	Yes

The pre-test has 3 questions:

1. What is the next number in the sequence below?

   00001 00010 00011 00100 _____

2. What decimal number is represented by 01011? How do you know?
3. How would you write the number 20 in binary? How do you know?

The post-test has 3 similar questions:

1. What is the next number in the sequence below?

   00001 00010 00011 00100 _____

2. What decimal number is represented by 01110? How do you know?
3. How would you write the number 25 in binary? How do you know?

The results of the pre-test are:

School	Pre
School #1	1/33 tested students got all 3 questions correct
School #2	1/24 tested students got all 3 questions correct

School #3	11/13 tested students got all 3 questions correct

The results of the post-test are:

School	Post
School #1	100% of students got all 3 questions correct
School #2	• 21/24 students had completely correct answers for the binary-to-denary conversion Q • 24/24 students had completely correct answers for the denary-to-binary conversion Q
School #3	N/A

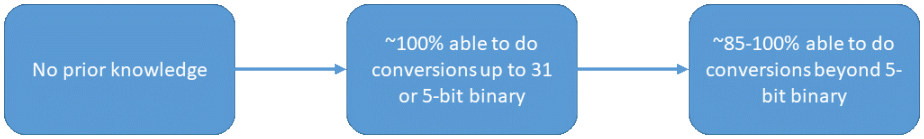
A follow-up test after the post-test was also designed:

1. How do you convert an 8 bit binary number like 10010111 to a denary (decimal) number? (show workings or explain your thinking)
2. How do you convert a denary number like 1350 to a binary number? First consider, what is the minimum number of bits you need to represent this number? (show workings or explain your thinking)

Here are the results:

School	Follow-Up
School #1	• 37/38 students had correct or partially correct answers for the binary-to-denary conversion Q • 33/38 students had correct or partially answers for the denary-to-binary conversion Q
School #2	• 23/23 students had correct or partially correct answers for the binary-to-denary conversion Q • 20/23 students had correct or partially correct answers for the denary-to-binary conversion Q
School #3	100% of students had correct or partially correct answers for the conversion problems.

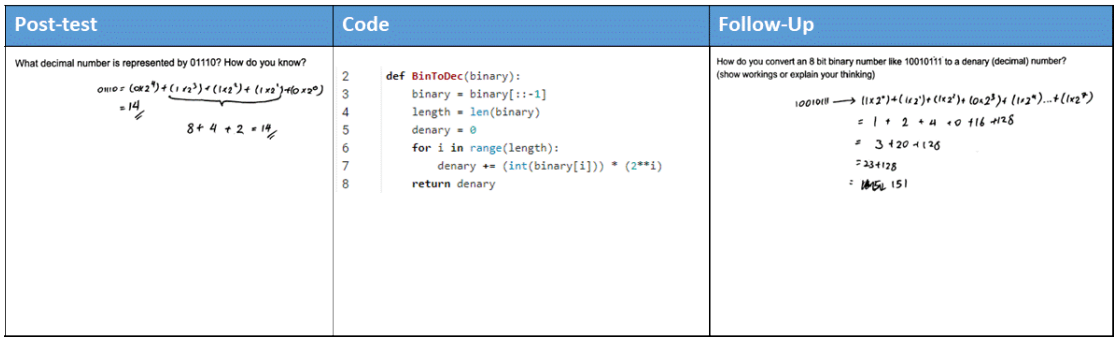
The point here is not to prove that unplugged is better but that it is at least no worse than traditional methods. For the class that already had prior knowledge, students certainly did no worse, and in fact, one student who scored low on the pre-test got all the questions right on the follow-up test.



The unplugged activity introduces to students the concrete mental model that anchored students' understanding of how binary numbers are represented; each digit corresponds with a number of dots on a card. The dots cards start with 1 dot to represent the rightmost digits and increases by a *2 pattern from right to left. The transfer test shows that students can convert binary numbers greater than 5 digits. They can generate the place values for more bits if the binary number contains more than 5 digits. Thus, the mental model can guide students to develop algorithms. There is evidence in their programming – students developed a rule – "if digits == 1, add its place value to a cumulative sum; else ignore. Go to the next digit."

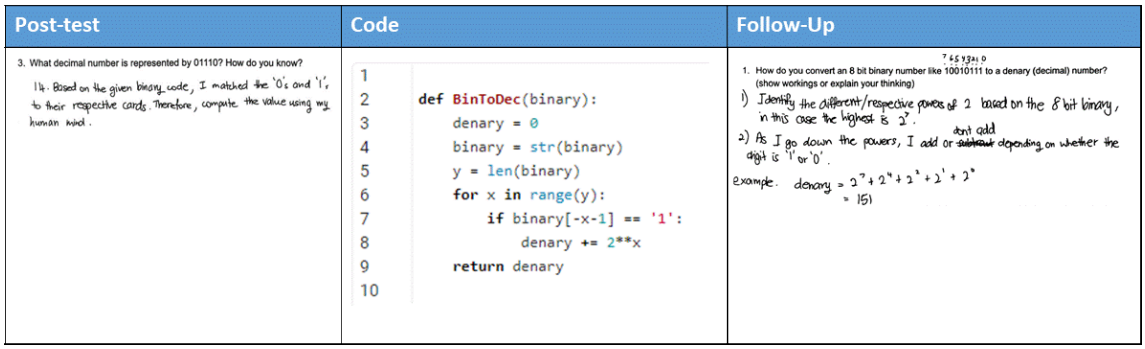
Our unique spin on this activity – where we deviated from the original unplugged published in the literature. We added a programming component. We put the "plug" back in, so to speak. in one class, we noticed that over 90% of students were able to correctly perform the algorithm on paper but only about half were able to program it. What accounts for this gap? Doesn't it seem unusual that if a student has a good grasp of an algorithm, they should be able to translate that to code? This finding raised questions about this gap and how to bridge it.

Here is an example of a student's work:



Post-test (was given at the end of the unplugged activity) - note the use of exponential expansion and evaluating the expression (came from explicit connection taught). In the student's code, note the core structure in line 7 (digit times 2 to the power); since index variable i starts at 0, the student flipped the string in line 3. In the follow-up task, we want to see if students can convert a binary number greater than 5 bits (limit of the unplugged activity) – note the flipped string. Thus one implication is that the students' mental model of an algorithm will influence how they code and in turn, how they perform the algorithm on paper. Thus, it is important to get the algorithm right in order to code it.

Here is an example of another student's work:



Post-test	Code	Follow-Up
3. What decimal number is represented by 01110? How do you know? 14. Based on the given binary code, I matched the '0's and '1', to their respective cards. Therefore, compute the value using my human mind.	1 2 def BinToDec(binary): 3 denary = 0 4 binary = str(binary) 5 y = len(binary) 6 for x in range(y): 7 if binary[-x-1] == '1': 8 denary += 2**x 9 return denary 10	1. How do you convert an 8 bit binary number like 10010111 to a denary (decimal) number? (show workings or explain your thinking) 1) Identify the different/respective powers of 2 based on the 8 bit binary, in this case the highest is 2^7. 2) As I go down the powers, I add or ~~subtract~~ don't add depending on whether the digit is '1' or '0'. example. denary = $2^7 + 2^4 + 2^2 + 2^1 + 2^0$ = 151

Post-test (was given at the end of the unplugged activity) - note the difference with Student A – this student "matched" the 0s and 1s to the cards rather than writing an expression. Wrt the code, note the core structure in line 7 (selection statement); also, this student didn't reverse the string but figured out how to use the iteration variable to traverse the string from right to left. Wrt the follow-up, compare this to language used in post-test; the process is a computational solution, which shows the influence that coding had on his understanding of the algorithm. Again the take-away is that students' mental model will influence how they code the algorithm.

4. General Summary of Findings

One of our findings is that there is a potential mismatch between the constraints of adapting an unplugged activity for the formal goals of schooling and its exploratory characteristics. We attempted to strike a middle ground by aligning the activities to learning objectives in the computing syllabus and creating assessments similar to classroom exams, yet still designing the activity to be as student-centred as possible. For instance, for the adapted binary activity, half the total time was spent on lecture, whole class discussions and interactive demonstrations, while the other half was spent on group work. The worksheet provided printed instructions for self-paced group activities and discussions to guide exploration. To encourage student interaction, we provided each group with only 1 worksheet, so that they need to discuss and reach consensus before committing to responses to the questions. As the binary numbers activity was originally designed for primary age students and we were adapting it for upper secondary students, we took advantage of the low threshold to make it possible for all students to achieve the basic learning objectives, and raised the ceiling by

challenging students to figure out how a particular magic "trick" worked, which was based on an understanding of binary numbers.

Although we developed lesson plans for other activities, they were loosely aligned with the syllabus and more seen as enrichment rather than being directly relevant to learning required concepts. To address this disconnect, we set out to work with teachers to design unplugged activities that would directly target particular learning objectives. We also wanted to continue exploring the connection between unplugged activities and programming. The first activity was based on the game Boggle, where players form words by connecting letters that are arranged in a 3x3 grid. This game activity was designed by the teachers of one of the secondary schools.

The second activity was designed by us researchers, and involved learning about lists, list commands, and list sorting. A list is visually represented as train cars, and physically represented as linked pockets with pieces of paper with the "data" written on them that go inside the pockets. Students are encouraged to use the manipulative to walk through programming code that uses list commands. They are then given scaffolds to develop a list sorting algorithm, starting with a list of 5 particular values, then a list of 5 unknown values, and finally a list of unknown length of unknown values. The study investigates the premise that students struggle with parsing and creating code with lists because of difficulty with visualizing states as the list is being referenced and modified. We want to know if students would use the manipulatives or if they develop other strategies, such as sketching the list as it changes states. Also, we want to see how far students progress in solving the list sorting problems, what kinds of algorithms they develop (expressed in code), and whether the manipulatives were helpful.

Both of these unplugged activities integrate programming code. Students are writing and reading code, and executing them with the aid of physical manipulatives. Teachers find

these designed activities to be more directly relevant to the learning goals of the syllabus, and they help to address certain concepts that students are struggling with. Using unplugged this way is more like how college lecturers use kinesthetic learning activities (KLAs) to make certain ideas more visible and concrete to students. These newly designed activities can then be evaluated using the design pattern developed by Nishida et al (2009) to see how well they fit with general characteristics of CS Unplugged activities. The one limitation is that the pattern assumes that students without specific CS knowledge should be able to participate in the activity. This part of the pattern would need to be adjusted to better fit a formal learning environment where unplugged activities are part of an instructional sequence. More work needs to be done to determine the effectiveness of using an unplugged approach in conjunction with a programming language, how the unplugged approach might support the learning of a programming language, and how unplugged activities might be followed up with relevant programming exercises.

TEACHING RESOURCES AND PACKAGES

In this research project, we have produced a set of enriched SOW/teaching packages that show how to embed unplugged and other pedagogies into the Computing subject, with our research data and analysis to support their use.

Unplugged Activities	MOE Computing Syllabus Scheme of Work
Searching Algorithm	1.1 Data Management: Data Lookup Function
Binary Number	1.2 Data Representation: Number Systems
Guess My Number	
Parity Bit Checking	2.2 Data Communications: Parity and Checksums
Deadlock Avoidance	2.2 Data Communications: Introduction to network
Mouse Programming	3.1 Problem Analysis: Modularity and Generalization
Tower of Hanoi	3.1 Problem Analysis: Modularity and Generalization

Concept of Variables	4.1 Program Development: Numeric Data Type and String Data Type
Swapping Variables	
Sorting Algorithm	4.1 Program Development: Lists and Loops
Sorting Network	
Selection Statement with Cards	4.1 Program Development: Selection statement

Some of our observations about unplugged methods are:

1. Students found them interesting – from our survey results and focus group discussions. Like many other studies, we have found that unplugged pedagogy greatly enhances students' interest in learning. In one school, we did an experimental design of teaching data sorting, students were allowed to experience the plug-in and unplug-in activities. From the survey results, we noted the whole class that prefers the unplugged teaching method. However, some students with higher learning ability preferred to study problems directly and write code. Some teachers told us that in the questionnaire after the end of a semester, the students still remembered the unplugged class at the beginning of the semester and mentioned that they hoped to have more such teaching activities. It can be seen that the impression of unplugged left on the students is more deep and lasting than traditional classroom teaching.

2. The visibility, operability, and contextualized teaching associated with unplugged activities made it easier for students to understand abstract computer science content. Through the practical operation of unplugged learning materials, students intuitively felt or experienced the calculation process that actually takes place in the internal world of computers. In one study on the sorting algorithm, we found that students doing the unplugged can more intuitively grasp the details and steps in the computer program, and express it in some

representation, when compared with the students in the control group. It is as if the students in the control group know that the stool must be composed of four legs, and the students in the experimental group know how to build the stool from one leg.

In summary, the reasons for using unplugged methods are:

- Brings in the social and kinaesthetic;
- Anchors understanding to concrete experiences;
- Focus on big ideas; Accessible to all;
- Avoids difficulties and distractions of using computers.

Indeed, one of the teachers shared with us: “Unplugged force the students to get off their chairs and move around. And also force them to think very carefully how things work. They learn to apply it into real world. By doing so they can internalize the conceptual concepts.”

Consistent with the literature, students’ learning difficulties (from observations and teachers’ feedback) in computing include:

- Students have undeveloped mental models of key concepts such as variables and iterative constructs, and yet they started coding
- Students do not yet have robust models of such
- Students do not know how to represent algorithms even in pseudo-code or English or flowchart
- Students can tweak or edit written code but have problems in planning and writing code from scratch

COMPUTING TEACHERS’ PERCEIVED READINESS TOWARDS IMPLEMENTING COMPUTING CURRICULUM

In Dec 2016, prior to the formal implementation of computing curriculum, we conducted a survey on computing teachers to seek their degrees of understanding, interest levels, capacities and challenges regarding the teaching of computing. 36 computing teachers (27 male and 9 female) from 19 schools participated in our survey.

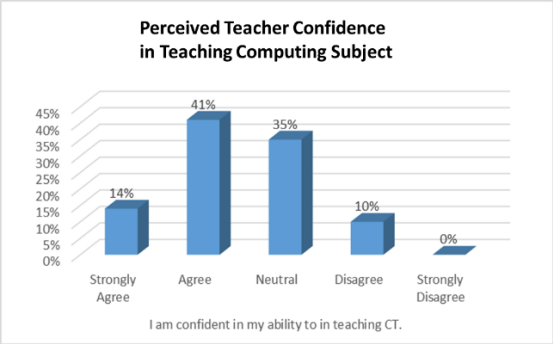


Figure. Perceived Confidence in Teaching Computing Subject

As to perceived teacher confidence to teach in computing subject (see above Figure), 55% teachers agreed that (14% strongly agree) they are confident to teach and implement CT in their classes. 35% are neutral while 10% consider that they are not ready. For teachers who lack confidence in teaching CT, they may need help on how to bring out expected learning outcomes.

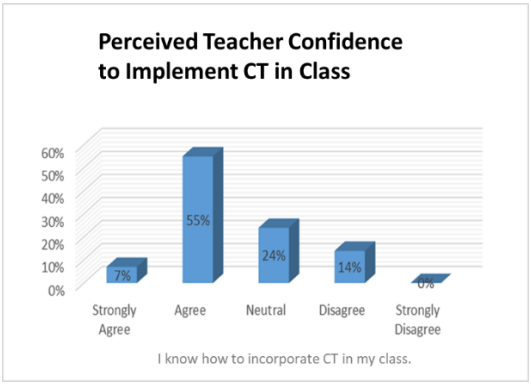


Figure. Perceived Readiness to Implement CT in Class

As to perceived teacher readiness to implement CT in Class (see Figure), 62% teachers agree (7% strongly agree) that they have been ready to incorporate and implement CT in their classes. 24% are neutral while 14% consider they are not ready.

As to perceived student readiness to learn computing in Class (Figure 3), 52% teachers consider (4% strongly consider) their students have been ready to learn computing. 28% are neutral while 21% consider they are not ready considering that CT is too complex to learn at the level of their students.

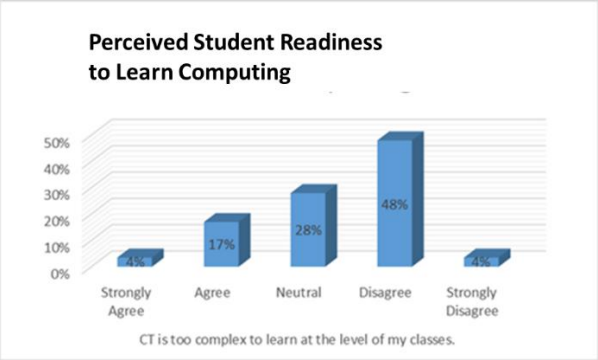


Figure. Perceived Student Readiness to Learn Computing

Thus, the confidence and readiness in teaching the computing subject and in incorporating CT into the teaching and learning can be further enhanced among the computing teachers. All teacher participants answer the question on the challenges in teaching computing, indicating that the lack of teaching resources (94%) ranks the first and lack of pedagogical knowledge (83%) ranks the second among the seven options (Table 1). When responding to open-ended questions, they also mention that they would need shared lesson plans and best practices by other schools to help their teaching. It is obvious that teachers are much more concerned about the resources for teaching and how to teach rather than what to teach, i.e. content knowledge.

Table 1. Perceived Challenges in Teaching Computing.

	Percentage	Count	Ranking
Teaching Resources	94%	34	1
Pedagogical Knowledge	83%	30	2
Ways to Motivate and engage students	69%	25	3
Instructional Skills	67%	24	4

Community Support	64%	23	5
Content Knowledge	61%	22	6
Computing Infrastructure in school	42%	15	7

MESO AND MACRO ISSUES IN UNPLUGGED APPROACHES

We study the introduction of unplugged approaches at the micro, meso and macro levels with their strong interdependencies on each other levels.

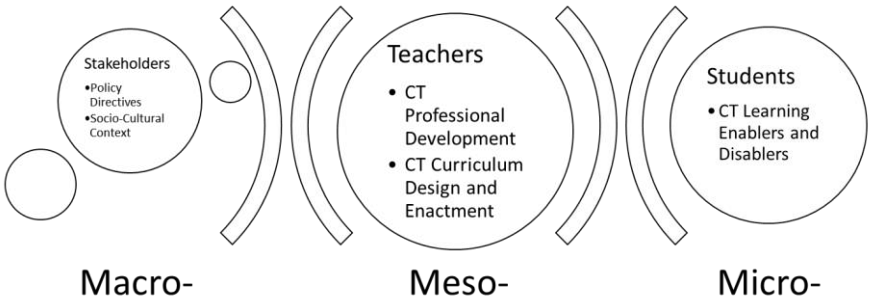


Figure. A DBR View of Formalization of Unplugged Approaches in CT Education

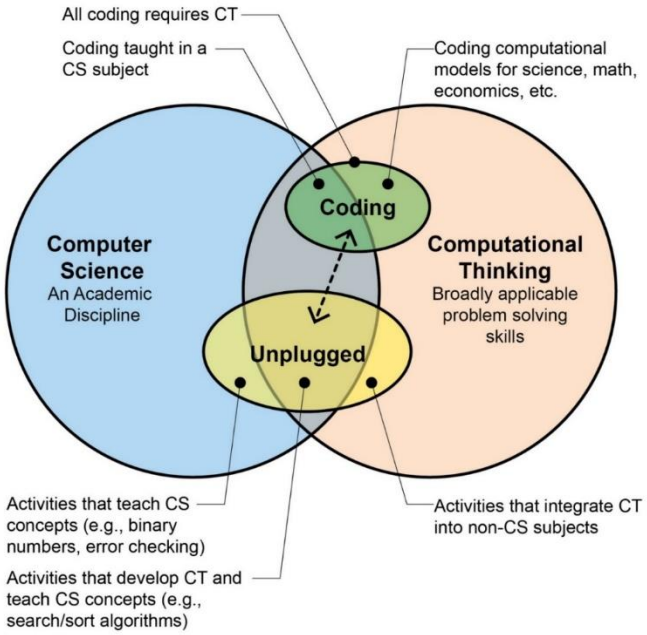
At the micro-level, the class of unplugged activities is not a homogenous entity, as the range of such activities spans a wide spectrum and includes different types at different levels of abstraction and fidelity to the intended CT concept or application. Unplugged activities can lead to some CT mastery but may not be complete for computing education. For example, a student can do the tracing to simulate the sorting of a list of numbers, e.g. demonstrating understanding of bubble sort, but still cannot represent the algorithm because of several reasons. If the curricular objectives is for the students to learn to write the algorithms, then

unplugged activities fit into a larger curricular flow in which the engagement and conceptual understanding is supposed to flow or lead into the expression of the algorithm into more structured representations like pseudo-code, flowchart or programming code. Thus design principles from the learning sciences are critical in designing the learning flow, including introduction of the notional machine model (of how a computer works) as a mechanism to realize the problem-solving steps.

At the meso-level, the effective enactment of an unplugged activity requires the facilitation by a teacher in order to lead to understanding. Teachers may be challenged on how these activities can be integrated into day-to-day classroom lessons and how they can assess their students' learning of the computing concepts. At the macro-level, an important aspect is to understanding the rationales and directives the policy makers have set for the current and future CT development.

A notable constraint in employing the DBR approach in CT education has been identifying or establishing the coordination or interdependence of multiple stakeholders. How are the roles of unplugged activities in computing education? How do teachers orchestrate and students do the unplugged? How different perspectives and efforts of stakeholders can be optimized for an eco-system view?

Using unplugged approach to develop CT contrasts with the more common approach of developing CT by learning to code. The figure below which is adapted from a similar figure from a report by Digital Promise (2017) shows how we relate CS and CT, with coding and unplugged representing two distinct teaching approaches that are often used together.



CONTRIBUTIONS OF STUDY

The following are some of the outcomes of this study:

1. Lesson plans for selected unplugged methods available to all O-level Computing teachers
2. Sharing of unplugged pedagogies to O-level Computing CPDD-facilitated NLC in March 2018
3. Sharing of unplugged pedagogies to A level teachers in May 2018
4. Sharing with NIE graduating teachers (NIE's BTOP) in Nov 2017, May 2018, Nov 2018, and May 2019
5. Symposium at RPC in June 2017

In terms of implications for practice, unplugged pedagogies add to the repertoire of teaching methods used by teachers to engage students cognitively in the computing tasks. Here is

our list of design principles for incorporating unplugged activities into the computing classroom:

1. Identify the learning objectives and communicate this to the students at some point during the lesson. It doesn't need to be at the very beginning if you have a better hook to grab the students' interest.

2. If you need the activity to replace a lesson in your syllabus, make sure that the unplugged activity's learning objectives align with the lesson being replaced.

3. Find the appropriate place in your curriculum. For the learning objectives that are more focused on CT (e.g., Algorithm Design or Program Development) and less on specific content (e.g., Computer Architecture, Logic Gates), you can use a variety of unplugged activities.

4. Find connections with other topics / concepts / skills in the curriculum.

5. Refer back to the unplugged activity later on in your teaching. Unplugged activities are memorable because of their hands-on nature and focus on concrete representations. Take advantage of this to help students recall and build upon what they learned.

6. If there aren't assessments in the activity, create them.

7. Design the lesson to be student centered. Don't let it become a lecture. Unplugged activities are by definition social and hands-on, using concrete materials (moving to abstract representation should happen, but don't push this too quickly).

8. Stretch the activity so that it's appropriately challenging for your students. The original unplugged activities were designed to be accessible by a wide range of school age kids (and adults). You may need to design additional worksheets or harder problem sets. One way to stretch is to look for opportunities for students to engage in CT (e.g.,

design an algorithm using flowchart, pseudocode, computer language; look for a pattern to generalize; build higher levels of abstraction). Another way to stretch the activity is to look for interesting applications in the real world or how the knowledge or skill they gained can be used to solve other problems.

9. The updated unplugged activities website has some good examples of CT connections for each of its activities. You can view the binary unplugged lesson here: https://csunplugged.org/en/topics/binary-numbers/unit-plan/how-binary-digits-work/

In terms of implications for policy and research, that some of computational thinking can be taught by unplugged pedagogies suggests that we can reach out to a bigger base of students without the distractions of getting the resources and infrastructure for computer programming.

In terms of earning gains (for studies involving intervention), we published a paper on students doing the unplugged sorting algorithm. Students learning an algorithm like sorting using an unplugged approach demonstrate the learning of computational thinking skills that are beyond students learning the algorithm using a more conventional approach (Looi et al, 2018).

In terms of follow-up activities, more workshops and courses to introduce unplugged pedagogies to computing teachers; increase the uptake of these pedagogies. In March 2019, an in-service course on unplugged pedagogies was offered at AST for computing and computer science teachers by the research team, and the course was favourably received.

CONCLUSION

There is a range of unplugged activities which can be introduced as part of computing lessons to firstly, engage and motivate students, and secondly, with good teacher facilitation,

help students develop appropriate mental models or what is called notional machines of computing concepts and algorithms. New activities can be designed by teachers that fit well with the computing curriculum. When students engage in these activities, they develop computational skills such as decomposition, pattern recognition, abstraction and algorithmic thinking.

ACKNOWLEDGEMENTS

We like to thank the participating teachers of our four collaborating schools. We also thank Chee Meng Teck, Patricia Lye and other colleagues from CPDD, MOE, in providing MOE's inputs to this project and their valuable suggestion. Lastly, we thank Bimlesh Wadhwa of School of Computing, NUS, and Liu Liu and How Meng Leong, for their contributions to this project.

REFERENCES

Curzon, P., McOwan, P. W., Cutts, Q. I., & Bell, T. (2009). Enthusing & inspiring with reusable kinaesthetic activities. In ACM SIGCSE Bulletin.

Huang, W. & Looi, C.K. (submitted for publication), Computational Thinking without Coding: Democratizing K-12 Computer Science Education Through "Unplugged" Teaching Approaches.

Looi, C.K., So, H-J., Toh, Y. & Chen W. (2011), The Singapore experience: Synergy of national policy, classroom practice and design research, International Journal of CSCL, Volume 6, Number 1, March 2011, pp. 9-37.

Penuel, W. R., Fishman, B. J., Cheng, B. H., & Sabelli, N. (2011). Organizing research and development at the intersection of learning, implementation, and design. Educational Researcher, 40(7), 331–337.

Russell, J. L., Jackson, K., Krumm, A. E., & Frank, K. A. (2013). Theories and Research Methodologies for Design-Based Implementation Research: Examples from Four Cases. Yearbook of the National Society for the Study of Education, 112(n2), 157-191.

Seow, P., Looi C-K., Wadhwa, B., Wu, L. & Liu, L., Computational Thinking and Coding Initiatives in Singapore, Proceedings of the International Conference on Computational Thinking Education, conference, Hong Kong, 13-15 July, 2017.

APPENDIX

If you have more than one appendix, please label them using Roman letters (e.g., Appendix A, Appendix B). Previously published research/technical reports can be appended here as well.