10

INTEGRATING DESIGN THINKING INTO K–12 COMPUTATIONAL THINKING CLASSROOMS IN TAIWAN: PRACTICES OF COLLABORATIVE ROBOTIC PROJECTS

Ju-Ling Shih

INTRODUCTION

Computational thinking (CT) is generally regarded as problem-solving skills with logical thinking ability and programming techniques. It is mostly treated as an integral part of interdisciplinary curricula instead of a stand-alone subject and is therefore often associated with STEM- and project-based activities in education.

This chapter begins with a broad overview of the existing standards and curriculum efforts in computational thinking education in the Taiwanese K–12 sector and the initiation of innovative instructions in the classrooms.

To illustrate how instructional innovation is formed, a model of the synergy of design thinking and computational thinking is introduced. The two thinking models may be paralleled, twisted, or aligned, and that should be considered with the instructional needs. This is an iterative process working between divergent and convergent thinking, taking the participants through the "discover," "define," "develop," and "deliver" stages. In this process, students solve problems and use programming to tackle situational challenges.

An instructional example of classroom implementation is given, showing how the teacher guided the students through the design thinking

and computational thinking process, working in groups to carry out an interdisciplinary collaborative drawing board project in training their problem-solving skills. The primary results of the classroom implementation are reported along with the revelation of the possible subsequent instructional challenges.

At the end of the chapter, the teacher training process—"to see, to feel, to change"—is presented to illustrate how the teachers shall be trained from knowing to doing, from now to new, with insight and goals.

K–12 CT EDUCATION IN TAIWAN

In this section, the Darmstadt model (Hubwieser 2013) is used to describe Taiwan's CT education in terms of educational system, curriculum, knowledge, teacher qualification, and teaching methods.

EDUCATIONAL SYSTEM

In Taiwan, students attend elementary school at age six, going through grades 1 to 6 in primary school, grades 7 to 9 in junior high school, and grades 10 to 12 in senior high school, to complete the twelve-year compulsory education. Taiwan's Ministry of Education (MOE) initiated a twelve-year curriculum for basic education in 2014 with the theme of "Spontaneity, Interaction, Common Good" (Tsai et al. 2011) and published the detailed guidelines for the 2018 academic year. The three themes encourage students (1) to do self-learning, think systematically to solve problems, and respond to changing situations for spontaneity; (2) to communicate with others and be aware of the world for interaction; and (3) to care about the health of society and people around them for the common good.

CURRICULUM

The rapid current of information technology development and innovative educational trends have driven the Ministry of Education in Taiwan to continue with educational reform, and it finally categorized "systematic thinking and problem solving" into the autonomous learning realm in the curriculum in 2014 (Ministry of Education 2014). Followed by several iterations of the guidelines and contents, the 2016 version of the curriculum

guidelines has added the technology learning area that includes two curriculum tracks: "information technology" and "life and technology."

The "information technology" curriculum is based on traditional computer courses that teach students the use of software such as word processing, slide presentations, graphical presentations, and so on. It uses computational thinking as its core, which nurtures abilities such as "computational thinking and problem-solving," "information technology and co-creation," "information technology and communication," and "attitude toward the use of information technology." The learning content includes six aspects: "algorithms," "coding," "system platforms," "information representation, management, and analysis," "information technology application," and "information technology and human society" (National Academy for Educational Research 2015). The main goal of CT is not to train students to become programmers but to promote logical thinking through the structure of programming language, intuitive cognition of machine processing, the ability to solve problems in a procedural way (Wing 2008), and the ability to explore wider learning approaches (Resnick 2013). Programming guides students to dissect a big problem into small problems to more clearly, accurately, and demonstratively explain the process of solving problems (Fernaeus, Kindborg, and Scholz, 2006 and Scholz 2006; Angeli and Giannakos 2020). These skills can be used in interdisciplinary learning and applied to real-world situations.

On the other hand, the "life and technology" curriculum offers students general technological knowledge, tools, and skills for the purpose of computational thinking, while at the same time enhancing students' cross-disciplinary knowledge integration abilities with maker practices (Ministry of Education 2016). It is a STEM-based curriculum that provides students with the ability to solve real-life complex problems in the realm of science, technology, engineering, and mathematics and trains them to have twenty-first-century key competencies (Kucuk and Sisman 2020).

KNOWLEDGE

In short, the computational thinking curricula are divided into two parts—(1) programming, which involves computers, and (2) making, which emphasizes on hands-on practices—that are practiced in the aforementioned two

curricula, respectively. Both of them are mostly done in the cross-disciplinary approach that integrates theme-based learning contents, but neither has predetermined teaching materials or a recommended course structure. From personal experience working with dozens of elementary and secondary schools, all teachers responsible for these curricula are searching for materials that are appropriate to the students they teach and are designing materials specifically for their own school environment and educational goals. For example, some elementary schools are teaching Scratch from the third grade up and teaching App Inventor for advanced levels; some schools emphasize STEM-based or IoT (Internet of Things) applications. The STEM education in Taiwan, defined as cross-disciplines of science, technology, engineering, and mathematics, is mostly done with robots and microcontrollers such as mBot, Lego EV3, Arduino, and Micro:bit. Since these technological tools require programming to make them work, the STEM curriculum is often cross-related to the programming curriculum. In most instances, these are inseparable. Thus, CT is regarded more as a problem-solving process with logistical thinking and done through programming. Psycharis (2013) mentions that the teaching of STEM requires students to explore and process data using computer language, so computing sits in a central position of STEM (Chi and Jain 2011; Henderson, Cortina, and Wing 2007).

In addition to the somewhat blurry line between the two curricula, some schools integrate them into their own school-based curriculum. If the school prioritizes teaching about the nearby temple, community fishery, or forest guardian, then the CT-related skills are integrated into that curriculum, bringing the skills into school-based real-life scenarios. Therefore, CT in Taiwan is no longer a stand-alone subject but is practiced in various forms with one or many teachers' efforts. In some schools, CT is extended into after-school clubs, which often train students to be high performers who participate in all kinds of national and international competitions. The whole practice matches the global educational spirit stated by Taiwan's MOE: cross-field integration, real-life scenarios, school-based curriculum, and personalized learning.

TEACHER QUALIFICATION

These courses are taught by qualified teachers with or without technology or information professional training due to uneven teacher allocations in rural and urban schools. Since the needs in different schools vary, teachers form their own groups to reach updated information and self-train to teach. They either join groups formed through the educational bureau in every city, which appoints head teachers to take the lead, or individual teachers initiate interest groups, study groups, work groups, exchange groups, or workshops and conferences for the whole country. Therefore, the teachings are of various innovative ways and are mostly bottom-up.

TEACHING METHODS

Nevertheless, Frymier, Shulman, and Houser (1996) also observe in the classrooms that teachers usually control the class content while students are simply content followers. Similar situations can be seen in the STEM and robotics education in Taiwan where students follow the manuals provided by the teachers, digital or paper, to assemble robots and code for actions to perform uniform tasks. The completion of tasks is deemed "learning" regardless of its implications.

General problems perceived in today's classroom practices include:

1. In the classroom, teachers normally provide self-made teaching materials and manuals to the students to follow the instructions and imitate the sample projects. Learning evaluation is based on the completion rate of the projects and sometimes the speed of completion.
2. The students often complete their projects knowing how without knowing why.
3. The students have few opportunities to be creative and make their own creations.
4. The project scenario is predesigned by the teachers, real or fictional, with either tasks or problems. Sometimes the scenario is dismissed in the teaching, which leaves only the robotic tasks to be done.

Donovan and Bransford (2005) describe the manuals as recipes that give lock steps to the process, which shortchanges the imagination. Questions, problems, and situations should be more unclear so that students are required to search for clues and innovative solutions. Thus, this chapter

proposes a pedagogical framework to provide another type of instructional model that would address the current needs of CT education.

INTEGRATE COMPUTATIONAL THINKING WITH DESIGN THINKING

Computational thinking was defined as using computer logic to solve problems. It draws on the concepts fundamental to computer science and outlines five cornerstones of computational thinking (Selby and Woollard 2013; Wing 2006).

(CT 1) decomposition: breaking a big problem down into smaller parts.
(CT 2) pattern recognition: looking for similarities within and between problems.
(CT 3) abstraction: taking the details out of a problem and ignoring irrelevant information.
(CT 4) algorithms: create the simple step-by-step rules to follow in order to solve the problem.
(CT 5) generalization: adapting solutions to current problems to solve new ones.

However, CT is generally recognized in two ways: specific CT refers to computer programming skills, while generic CT refers to overall problem-solving skills. But Wing (2006) more precisely said that CT involves solving problems, designing systems, and understanding human behavior, so it is necessary to perceive CT as computer programming, scenario connection, and interdisciplinary learning, which is a comprehensive view of both specific CT and generic CT.

DESIGN THINKING AS PROBLEM-SOLVING PROCESS

While CT describes the techniques of each production or problem-solving stage, design thinking (DT) guides one through the production process, providing insight to approach the problem. The major step of DT that is distinct from other teaching models is to start from the "empathy" stage before diagnosing the problems and ideating for solutions. For many classroom practices, students deal with targeted projects without having to think about how and why the projects would be made in a certain way.

Design thinking is a thinking model that places creating a humanistic environment as top priority and treats convenience and problem-solving with creativity as the key goals. This requires students to have good insights and to generate creative solutions. Therefore, DT can help

students think like designers, confront difficulties, and solve complex problems in schools, companies, and daily life (Brown and Wyatt 2010). The five steps of design thinking are defined by David Kelley in IDEO (2001):

(DT 1) empathize: identify the target users and collect information about the users to solve their problems from their views.
(DT 2) define needs: recognize the needs of your users and define the scope of the problems.
(DT 3) ideate: think about how to solve problems and brainstorm for creative solutions.
(DT 4) prototype: use tools to build representations of your idea and make prototypes.
(DT 5) test: test the idea to the user, discuss whether the problem is solved, and get feedback.

The designers should go through repeated testing and iterations so that the final product can be closer to the needs of users.

SITUATIVE COMPUTATIONAL THINKING

In classroom practices, there are two major types of instructional designs for CT projects but not limited to their variations. One type is to assign small project tasks to students to design creative products with the given resources; the other type is to provide big problem scenarios so the students have to come up with creative solutions. The earlier works more like project-based or task-based learning, such as creating a Scratch game or creating a functional robot; while the latter is more like theme-based learning, situated learning, or even game-based learning wherein a larger learning scenario is presented with a more complex problem. The first type is practiced more in schools since the latter takes more designing effort and time for both teachers and students.

Based on the aforementioned statements of either case, we see a need for a learning model where CT is applied in conjunction with interdisciplinary learning. We defined this as "situative CT" (Shih 2020). As Wing (2008) states, CT is conceptualizing, not programming; and CT is a way that humans think, not computers. Therefore, situative CT is further described as having the following four features that are described as distinct from specific and generic CT (Shih 2020): a) it is a combination of specific CT and generic CT that leads students to write programming and solve problems at the same time; b) it is practiced in the contextual

situation, normally a theme-based scenario that relies on related domain knowledge correspondence; c) CT skills are no longer the learning goals but the tools; and d) problems are situated in the scenario in which students have to respond to problems in context instead of conducting uniform tasks.

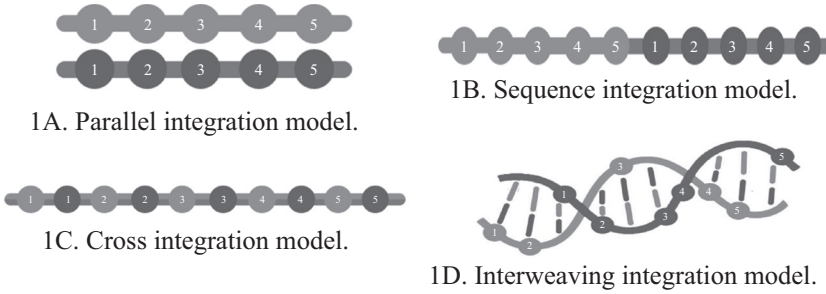
Overall, situative CT learning scenarios emphasize the individualistic production in real-life scenarios. In the confined and problematic condition, students solve contextual problems by accessing the appropriate resources from limited options and generating learning strategies to be used for solving problems. The class should give students space for individualistic creation rather than uniform production. The advanced-level learning scenario can be fluid and dynamic rather than static; sometimes presented with social dilemmas, students would have to make up strategic and feasible plans spontaneously to confront the complex problems.

To sufficiently address the instructional practice of situative CT, a framework with more flexible guidelines would work better than traditional manual-oriented step-by-step instructions. Design thinking is a process that helps designers systematically observe, analyze, extract, and apply these techniques to solve problems in innovative ways with human concerns (Brown 2009). It uses an interdisciplinary approach (Barak and Assal 2018) by connecting the disciplinary systems to the context of the real world (Breiner, Harkness, Johnson, and Koehler 2012; Honey, Pearson, and Schweingruber 2014). In the leading cases and most worked examples of DT, it is used in producing products that can be mass-produced in the market or widely used by a group of people (e.g., a supermarket shopping cart, a handbag, or a low-cost air-conditioner for classrooms); or it is used to transform organizations or living conditions, working toward solving daily life problems, (e.g., team morale, energy efficiency, or learning effects).

A CLASSROOM IMPLEMENTATION EXAMPLE

INTEGRATION OF DESIGN THINKING AND COMPUTATIONAL THINKING

The integration of any two systems would come in various ways (figure 10.1), such as paralleled (1A), sequenced (1B), intersected (1C), and interwoven (1D) integration models. When two systems are carried out in parallel, they



10.1 Integration models of two systems.

have to perform two works at the same time in a synchronous way (1A). It requires two systems to have similar work patterns and procedures so that they can be coordinated nicely. Nevertheless, it is rare to have two working procedures or conceptual frameworks that can be done together side by side. One of the easiest ways to convey two systems is to do them in consecutive order (1B). However, the sequence integration model is simply performing two things in two different times without interacting with each other. With that said, a cross-integration model is to take turns performing single steps of each system (1C). Attention must be paid to make sure that every step of each system can follow the previous step of the other system so that the complete sequence makes sense. Other than that, the interweaving integration model seems to be the most logical and reasonable way of merging two systems (1D). Some steps from either system would be distinct from each other and have to be performed separately, while some steps are similar so they cross over each other and can be performed together.

COURSE DESIGN AND PRACTICES

With the possible instructional models in mind, an instructional implementation example was conducted in a first grade information technology class in a secondary school in Taiwan (Huang 2020). This example is closer to the practice of project-based learning with the integration of STEM than the dynamic game-based learning scenario. A total of twenty-three students aged thirteen participated this course, with eleven boys (47.8 percent) and twelve girls (52.2 percent). About half of the students

had experience in coding with Scratch, and all students had computer information education at the elementary school level.

The goal of this instructional practice aimed to teach students to use technology to work on a collaborative robotics project (situative CT). They were guided to create a combinational project wherein each group devoted its own piece of the creation to the class and all pieces of creations were assembled into one complete functional project (generic CT). Students used computer programming (specific CT) to mobilize the automated mechanism with the connections of robots, sensors, and IoT.

The course was conducted once a week for one class hour with forty-five minutes each. Other than pre- and posttest questionnaires, the teacher used learning sheets to guide students' learning, which were analyzed to ascertain the students' design thinking process. Meanwhile, observations were done to monitor students' learning conditions. The complete research results can be viewed in Huang's study report (2020).

Six groups of students with about four students per group had respective tasks for completing different parts of a simulated automated container terminal (situative CT: theme-based scenario): security gate, container ship, shipping route, crane, container, and cargo truck. In the CT class time, students learned basic programming using Scratch and micro:bit to control movements and functions of the IoT sensors such as ultrasound, motors, buzzers, LED lights, and ultrared light sensors (situative CT: CT skills as tools, not goals).

This attempt treated CT as problem-solving skills and DT as a problem-solving process, therefore conceptually aligning DT in the beginning and the end of the course, with CT in the middle to teach students how to deploy the technologies. The teacher first presented the problem tasks to the students, describing the real-life scenario in which a large amount of cargo ships and trucks come and go and large amounts of cargo are to be transported in the harbor (situative CT: contextual situation and problems). Therefore, an automated container terminal is needed to guide the ships and trucks to go on the designated routes, and the cargo is to be transported automatedly by detections of approaches, weights, directions, and movements of related transportation. At this time, the students searched for information and attempted to not only dissect the problems but also to find creative solutions to the problems with mechanical skills.

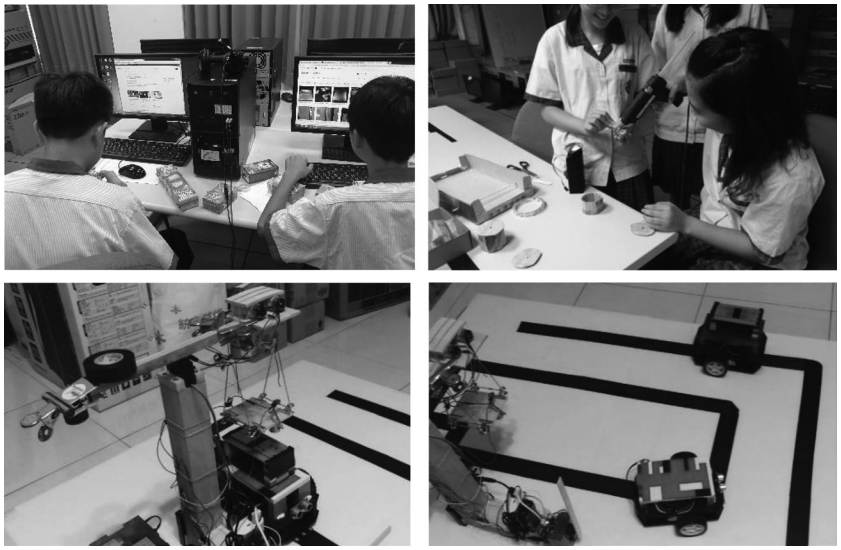
The whole teaching model is more similar to the interwoven integration model (figure 10.1: 1D). The teaching concept is to teach students the design thinking process and computational thinking skills at the same time. CT and DT start in a parallel model. The teaching started with the design thinking process of empathizing with the users and defining needs (DT 1 and 2) so that students understood how to approach a problem with a goal in mind. Practice examples of DT used in the course were "how to turn an empty space of classroom into a coffee shop and make a plan about the space design, equipment, menu, etc." Then, decomposition, abstraction, algorithm, as well as pattern recognition abilities (CT 1, 2, 3, and 4) were taught so that the students learn how to dissect the problems and turn the small tasks into programming languages.

After that, three Scratch activities were chosen, such as robot movements, music playing, and interactive sensors, so that the students not only learn how to do programming but also approach tasks in the logic of computer language (CT 5). With the skills at hand, the teacher assigned the class project "Automated Container Terminal" and guided the students to approach their respective tasks in groups. The students then started to go through the design of prototype and development stages of design thinking (DT 3 and 4). Though in the teaching process it looks like CT and DT are sequenced, the students are actually learning two things at the same time since they have to learn CT skills while thinking how the skills can be used in the design of their projects. This is the intersecting point of the two models.

What followed was the process of hands-on work, which was similar to that in many other STEM classes. The only difference was that each group was responsible for a part of the final project, so their design was not only to solve their own problem but also to communicate with other groups and find out the collaborative connections between the groups.

Figure 10.2 shows how the students were working on the collaborative robotics project with computer programming and hands-on maker practices. Their final integrated project showed the efforts of each group where the ship, cargo terminals, crane, and cargo truck can work in the proper sequence and flow well.

From this class project, the students learned from trial and error induced by the problems that failed the mechanism (DT 5). For example,



10.2 Students work on the collaborative robotics project.

the first version of the crane used the L-shaped boom to lift the cargo, but the balsa wood was too thin to handle the heavy cargo, so they changed it into plywood and used an MG996R servo motor to swing the cargo from the ship to the truck. With several iterations, the whole class collaboratively made respective adjustments to fit their pieces into the large project. That was the design thinking spirit this course attempted to deliver.

STUDENT EVALUATIONS

Since the instructional practice focused on both CT and DT, the evaluation also covered both parts. Valid pre- and posttests results were twenty-two copies. The design thinking questionnaire and computational thinking questionnaire each has five corresponding aspects with a total of twenty-three and twenty-five questions, respectively. Both questionnaires are on a five-point Likert scale ranging from 5 as strongly agree to 1 as strongly disagree.

The pretest and posttest of design thinking questionnaire results are shown in table 10.1. The prototype aspect ($t=2.951$, $p=.008$) and test aspect ($t=2.731$, $p=.013$) both reached significant differences. It shows that after the students had the learning experience, their protype and test

Table 10.1 T-test results of design thinking questionnaire

		N	Mean	SD	t value	p value
Empathy	pretest	22	19.14	3.427	1.656	.112
	posttest	22	20.50	3.751		
Define needs	pretest	22	13.82	3.581	1.238	.230
	posttest	22	15.00	3.281		
Ideate	pretest	22	17.91	3.558	1.019	.320
	posttest	22	19.00	3.147		
Prototype	pretest	22	18.00	4.059	2.951	.008**
	posttest	22	20.50	2.972		
Test	pretest	22	14.50	1.896	2.731	.013*
	posttest	22	16.41	2.462		

*p<.05, **p<.01, ***p<.001

abilities improved. But the empathy, define needs, and ideate aspects did not have significant differences. From the observation and after-course reflection with the teacher, there were several in-depth findings and suggestions for conducting this course. First of all, students, especially those in Taiwan, had little training on design projects in a systematic and creative way. The teacher had to provide a lot of guidance in the creation process, so the ideate stage was not totally self-proceeded but required a large amount of teacher assistance. Furthermore, since this Automated Container Terminal project was done from the third-person point of view, its users were difficult to define and the students had a hard time relating, empathizing, and defining needs from a targeted perspective. Thus, the first three aspects did not see significant self-perceived improvements in the questionnaire. On the other hand, since the students were mostly focused on the hands-on projects, their prototype and test aspects had significant improvements.

The pretest and posttest of the computational thinking questionnaire results are shown in table 10.2. The decomposition aspect ($t=2.452$, $p=.023$), algorithm aspect ($t=-2.149$, $p=.043$), and pattern recognition aspect ($t=2.917$, $p=.008$) reached significant differences. This shows that

Table 10.2 T-test results of computational thinking questionnaire

		N	Mean	SD	t value	p value
Decomposition	pretest	22	18.55	3.961	2.452	.023*
	posttest	22	20.64	2.735		
Abstraction	pretest	22	18.14	2.077	1.767	.092
	posttest	22	19.68	2.950		
Algorithm	pretest	22	18.91	2.245	2.149	.043*
	posttest	22	20.64	2.854		
Pattern recognition	pretest	22	18.41	1.681	2.917	.008**
	posttest	22	20.68	2.679		
Generalization	pretest	22	19.86	2.513	1.929	.067
	posttest	22	21.36	2.381		

*p<.05, **p<.01, ***p<.001

after the students had the learning experience, their decomposition, algorithm, and pattern recognition abilities improved. But the abstraction and generalization abilities did not reach significant differences. From the observation and after-course reflection with the teacher, there were several in-depth findings and suggestions for conducting this course. In the teaching process, the abstraction skill was less emphasized since the example given of turning the classroom into a coffee shop did not illustrate the abstraction process and neither did the programming activities with Scratch. Therefore, the students had little idea about what abstraction actually is and could not apply it in their practices. Since the course time is limited and rather short in terms of the semester-long project, the course ended with the completion of the project and summative evaluation without having the chance to approach another similar case for the students to transfer onto. Thus, the generalization skill was not practically used or demonstrated, thus showing insignificant improvement.

From this worked example, several teaching issues were revealed. For example, there is little class time for conducting a large-scale project. The time needed for the students to solve problems is relative to the scale of complexity of the problems. With limited class time, the teachers tend

to sacrifice deep thinking for efficiency. Meanwhile, teachers should change their teaching habits to provide guidance instead of instructions. The teaching model should be student-centered rather than teacher-centered and process-based rather than product-based. The evaluation methods should also be transformed into criteria-based instead of test-based. Aspects such as learning behaviors, effectiveness, emotions, and sense of achievement shall be considered as important factors of learning. Evaluation should be more practice-oriented with rubrics, use dynamic observations, and emphasize more the various aspects of creation in the formative way than standardized evaluation in the summative manner.

REVISED INTEGRATION MODEL

As an instruction procedure, it would certainly limit the possibilities of model applications. After this teaching practice of integrating CT and DT, we found that the integration models of type A to type D are not enough to support the work of situative CT. The natures of the concepts of the two systems are not the same. The two models of CT and DT do not correspond to each other step by step, nor do they take care of only the beginning or the end of the production process. They work in a flexible sequence, mostly iterative and sometimes overlapping. Also, since one or more CT skills might be used in every stage of the production, the two models should interweave with each other in more reasonable steps but in a more interchangeable manner. Therefore, type E (figure 10.3) is revealed to be the most feasible integration model.

Since teaching is linear and progresses with time, it is necessary to treat the conceptual framework in the procedural way. The aligned integration model has the benefit of the interwoven integration model (figure 10.1: 1D) while eliminating the rigidity of the cross-integration model (figure 10.1: 1C). It is also important to note that the steps are not locked into what is proposed but should be flexible to accommodate various instructional conditions.

10.3 Aligned integration models of two systems.

Computational thinking encompasses five steps, namely decomposition, abstraction, algorithms, pattern recognition, and generalization (Wing 2008). On the other hand, design thinking involves five stages, namely empathize, define needs, ideate, prototype, and test (d.School 2010). While the former describes more about the techniques used in one or many stages, the later provides a procedural paradigm of carrying out an innovative design.

To give students an overall concept of DT and sufficient understanding of technological techniques related to CT, it is important to schedule DT0 and CT0 stages. In the DT0 stage, the teacher introduces the history, basic concept, practice model, and a worked example practice of design thinking; in the CT0 stage, the teacher teaches various related technologies that might be used in the assigned project. Without going through the CT0 stage, students might generate wild ideas for problem solutions. Their creative design might not be feasible, the difficulty level would be out of control, and resources would be insufficient. Teachers need to choose relevant technological techniques that are extendable and with variations so that students can design their creative products based on the skill foundations taught.

Therefore, guiding students through the design thinking process with the concepts of computational thinking, the suggested aligned integration model offers a procedural flow of the instruction. The teacher presents the problem scenario. Then the students empathize with the people who are involved in the problem, either as active users or passive receivers (DT 1: empathize), and define their needs to focus on the essence of the targeted problems (DT 2: define needs). At this time, both the teacher and the students often blindly jump right into the trial-and-error hands-on work; however, when errors happened too often, students get frustrated and the teacher has to come back to this step to dissect problems. Therefore, it is important to remind the teachers to carefully design the activity in this stage so that problems can be seen in a more constructive way. While facing complex problems, the students should be guided to decompose the big problems into smaller ones (CT 1: decomposition). Seek problems that are comparable to those in previous experiences and search for similar patterns that can be solved with known solutions (CT 2: pattern recognition). At this time, the teachers should find sufficient previous

experiences that can provide important guidance and directions. If there aren't enough familiar experiences, a few examples would help to stimulate innovation. The next step is to generate creative solutions that can integrate possible solutions into one (DT 3: ideate). In this step, the question is often about how detailed the idea should be. Suggested principles are to assess whether the design is already functional, concrete, and workable. If it is too conceptual, vague, and imaginative, it will be hard to realize it at the next step. Once the design concept is drafted, the students need to simplify the complexity and try to use more obvious terms or rules to describe the solutions (CT 3: abstraction). This step works more like setting up concrete goals eliciting the main themes of the project. At that point, students attempt to transform the solution into computer logic or using computer languages (CT 4: algorithms). After creating the prototype (DT 4: prototype), students repeat to test the results and adjust the solutions through iterations (DT 5: test). Once the problem is solved, students can use this experience for problems encountered in the future (CT 5: generalization). In class, there might not be a chance to make evident the learning transfer, and it can take longer to reveal the impact of the learning to the students.

This proposed model emphasizes the design thinking process, encourages students to freely design for individualistic creation, and focuses all student teams within and between group projects. It is also important for the teachers to avoid providing direct answers to the students when they encounter difficulties, but to give clues, hints, and updated information and resources so they can solve the problems by themselves. Guiding them to periodically reflect on the DT process will increase their self-awareness and metacognition of every step.

TEACHER TRAINING PROCESS

In Taiwan, there is also a digital divide between urban and rural areas, mostly caused by geographical differences. In the cities, Internet and hardware resources as well as governmental and administrative supports are much more abundant than in the countryside and mountain areas. This also contributes to the insufficiency of teachers in those areas since most teachers prefer to stay in the cities closer to home. The distant

schools thus have fewer resources for professional development. The curriculum, consequently, is weaker than those in the cities.

To provide more in-service training to the teachers who work in distant schools, the Taiwanese government invites and assigns professors from the universities as well as leading school teachers to collaborate with K–12 schools to give professional guidance in terms of designing the curriculum and implementing new technologies. By enhancing school principals' leadership, and creating teachers' community of practices, the city schools and rural schools exchange teaching experiences and resources in many ways. Schools, both rural and urban, establish their own school-based curriculum, form their communities of practices, foster teams for STEM, maker, and CT clubs, and sometimes aim to participate in national and even international competitions. With activities that are different from the traditional lecture classrooms, students' learning motivation gets higher, which influences the teachers to move forward with the students. The co-learning cycle becomes more positive.

The whole process is to give teachers a paradigm shift knowing how important CT is, how to integrate CT into their classes, and how to guide students for meaningful practice. There are some rules of thumb for teacher development.

The traditional process of teacher training is to gather the teachers from everywhere and give lectures to them expecting them to put the theory into practice right away. However, it has been found that without constructive guidance and examples to follow, the teachers still do not know how to put conceptual frameworks into practice. Therefore, providing opportunities to experience should be the first step. Just as in the classical saying "to see, to feel, to change," it is better to let the teachers experience the modeled teaching and the effects of change so they have more incentive to learn new things.

In the second step, the key point is to form teacher communities so they can share resources, exchange ideas, and support each other going forward. The goal is to encourage communication between teachers. Meanwhile, conducting needs assessments and providing individualized support can help the teachers do a better job.

The third stage is to build common ground and understanding of the core curriculum concept with the teachers and guide them to customize the design for their own classrooms.

Last, walk together with the teachers. As they put these ideas into practice, be with them and give advice on what they can do. Avoid expecting perfection or unified goals. All schools, teachers, students, and classes are different. Help them through the individualized process.

The double diamond model of design thinking is an iterative process working between divergent and convergent thinking, taking the participants through the discover, define, develop, and deliver stages. It is very similar to the instructional design process while the teachers produce lesson plans. In the same way, it brings the teachers from knowing to doing, from now to new, with insight and goals.

This can provide a model for the teachers when thinking about creating a nurturing scenario. First, emphasize individual creation as well as group collaboration. Avoid one-size-fits-all creation goals, providing assembling manuals, and giving standard prototypes. Then, if possible, extend to a more complex situation and encourage a more creative and dynamic problem-solving process. The teachers can establish a narrative scenario with embedded issues. They can implement clues to get, conflicts to solve, and goals to achieve in order to provide an interaction, creation, and resolution process for the students.

It is an extended goal that through the teacher training process, teachers' motivation and belief can be enhanced and their concerns can be addressed.

CONCLUSIONS

In the era of advanced technology and rapid social change, students are going to face complex problems in the near future. They should have the ability to adapt to the environment in which unpredictable situations might occur. Students should learn to observe, detect, and formulate their own problems along with more collaborative, creative, and critical thinking practices.

From the teachers' perspective, there is the need to integrate existing learning models and form a new one that can be adopted more flexibly. In this chapter, we have proposed that design thinking be integrated into computational thinking classrooms. The proposed teaching model in this chapter should be iterative since the process might vary from case to case due to the various natures of projects.

ACKNOWLEDGMENTS

This study is supported in part by the Ministry of Science and Technology of Taiwan under MOST 104–2628-S-008–002-MY4. I would like to show my appreciation for my research team of several previous studies.

REFERENCES

Angeli, Charoula, and Michail Giannakos. 2020. "Computational Thinking Education: Issues and Challenges." *Computers in Human Behavior* 105: 106185.

Barak, Moshe, and Muhammad Assal. 2018. "Robotics and STEM Learning: Students' Achievements in Assignments According to the P3 Task Taxonomy—Practice, Problem Solving, and Projects." *International Journal of Technology and Design Education* 28, no. 1: 121–144.

Breiner, Jonathan M., Shelly Sheats Harkness, Carla C. Johnson, and Catherine M. Koehler. 2012. "What Is STEM? A Discussion about Conceptions of STEM in Education and Partnerships." *School Science and Mathematics* 112, no. 1: 3–11.

Brown, Tim. 2009. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. New York: HarperCollins.

Brown, Tim, and Jocelyn Wyatt. 2010. "Design Thinking for Social Innovation." *Development Outreach* 12, no. 1: 29–43.

Chi, Hongmei, and Harsh Jain. 2011. "Teaching Computing to STEM Students via Visualization Tools." *Procedia Computer Science* 4: 1937–1943.

Donovan, M. Suzanne, and John D. Bransford. 2005. *How Students Learn—Science in the Classroom*. Washington, DC: National Academy Press.

d.School. 2010. "An Introduction to Design Thinking Process Guide." https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/36873/attachments/74.

Fernaeus, Ylva, Mikael Kindborg, and Robert Scholz. 2006. "Rethinking Children's Programming with Contextual Signs." In *Proceedings of the 2006 Conference on Interaction Design and Children*, edited by Kari-Jouko Räihä and Johanna Höysniemi, 121–128. New York: Association for Computing Machinery. https://doi.org/10.1145/1139073.1139105.

Frymier, Ann Bainbridge, Gary M. Shulman, and Marian Houser. 1996. "The Development of a Learner Empowerment Measure." *Communication Education* 45: 181–199.

Henderson, Peter B., Thomas J. Cortina, and Jeannette M. Wing. 2007. "Computational Thinking." In *Proceedings of the 38th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*, 195–196. New York: ACM.

Honey, Margaret, Greg Pearson, and Heidi Schweingruber. 2014. *STEM Integration in K-12 Education: Status, Prospects, and an Agenda for Research*. National Academy of

Engineering; National Research Council. Washington, DC: The National Academies Press.

Huang, Pau-Ching. 2020. *STEM-Based Computational Thinking Course with Design Thinking Method: A Collaborative Project of Automated Container Terminal (ACT)*. Unpublished master's thesis, Tainan City, Taiwan. https://hdl.handle.net/11296/nhmjyv.

Hubwieser, Peter. 2013. The Darmstadt Model: A First Step towards a Research Framework for Computer Science Education in Schools. In *Proceedings of International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, 1–14. Berlin, Heidelberg: Springer.

Kucuk, Sevda, and Burak Sisman. 2020. "Students' Attitudes towards Robotics and STEM: Differences Based on Gender and Robotics Experience." *International Journal of Child-Computer Interaction* 23–24: 100167.

Ministry of Education. 2014. *Curriculum Guidelines of 12-Year Basic Education General Guidelines*. Taipei: Ministry of Education.

Ministry of Education. 2016. *Draft Curriculum Guidelines of 12-Year Basic Education Guidelines for Science and Technology Courses*. Taipei: Ministry of Education.

National Academy for Educational Research. 2015. *Draft Curriculum Guidelines of 12-Year Basic Education Guidelines*. Taipei: Ministry of Education.

Psycharis, Sarantos. 2013. "Examining the Effect of the Computational Models on Learning Performance, Scientific Reasoning, Epistemic Beliefs and Argumentation: An Implication for the STE Agenda." *Computers & Education* 68: 253–265.

Resnick, Mitchel. 2013. "Learn to Code, Code to Learn." Accessed December 25, 2016. https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn.

Selby, Cynthia, and John Woollard. 2013. "Computational Thinking: The Developing Definition." Accessed April 1, 2014. http://eprints.soton.ac.uk/356481/.

Shih, Ju-Ling. (2022). "Computational Thinking in the Interdisciplinary Robotic Game: The CHARM of STEAM." In *Computational Thinking Education in K-12: Artificial Intelligence Literacy and Physical Computing*, edited by Siu-Cheung Kong and Harold Abelson, 245–269. Cambridge, MA: MIT Press.

Tsai, Ching-Tian, Yen-Hsin Chen, Ming-Lieh Wu, Mei-Kuei Lu, Sheng-Mo Chen, De-Lung Fang, and Yung-Feng Lin. 2011. "A Study Regarding the System Connected the Key Competencies with Individual Academic Field in the Curriculum of K to 12." *National Academy for Educational Research Report* (NAER-99–12-A-1-05-00-2-11). Chiayi County, Taiwan: National Chung Cheng University.

Wing, Jeannette M. 2006. "Computational Thinking." *Communications of the ACM* 49, no. 3: 33–35.

Wing, Jeannette M. 2008. "Computational Thinking and Thinking about Computing." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 366, no. 1881: 3717–3725.

This is a section of doi:10.7551/mitpress/14041.001.0001

Computational Thinking Curricula in K–12

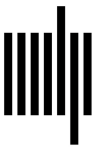
International Implementations

Edited by: Harold Abelson, Siu-Cheung Kong

Citation:

Computational Thinking Curricula in K–12: International Implementations
Edited by: Harold Abelson, Siu-Cheung Kong
DOI: 10.7551/mitpress/14041.001.0001
ISBN (electronic): 9780262378642
Publisher: The MIT Press
Published: 2024

The open access edition of this book was made possible by generous funding and support from MIT Press Direct to Open



The MIT Press

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Stone Serif by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Abelson, Harold, editor. | Kong, Siu Cheung, editor.
Title: Computational thinking curricula in K-12 : international implementations / edited by Harold Abelson and Siu-Cheung Kong.
Description: Cambridge, Massachusetts : The MIT Press, [2024] | Includes bibliographical references and index.
Identifiers: LCCN 2023027971 (print) | LCCN 2023027972 (ebook) | ISBN 9780262548052 (paperback) | ISBN 9780262378659 (epub) | ISBN 9780262378642 (pdf)
Subjects: LCSH: Computer science—Study and teaching—Case studies. | Problem solving—Study and teaching—Case studies.
Classification: LCC QA76.27 .C478 2024 (print) | LCC QA76.27 (ebook) | DDC 004.071—dc23/eng/20230905
LC record available at https://lccn.loc.gov/2023027971
LC ebook record available at https://lccn.loc.gov/2023027972

ISBN: 978-0-262-54805-2