Bringing Computational Thinking to Nonengineering Students through a Capstone Course

Keeheon Lee
Underwood International College, Yonsei University
keeheon@yonsei.ac.kr

Younah Kang
Underwood International College, Yonsei University
yakang@yonsei.ac.kr

Abstract

Although the concept of computational thinking has flourished, little research has explored how to integrate various elements of computational thinking into an undergraduate classroom setting. Clarifying core concepts of computational thinking and providing empirical cases that apply computational thinking practices into a real-world educational setting is crucial to the success of software engineering education. In this article, we describe the development of a curriculum for a social innovation capstone course, using core concepts and elements of computational thinking. The course was designed for undergraduate students of a liberal arts college at a university in Korea. Students were asked to define a social problem and introduced to the core concepts and processes of computational thinking aided by Arduino and Raspberry Pi programming environments. After building a business model, they implemented a working prototype for their proposed solution. We document class project outcomes and student feedback to demonstrate the effectiveness of the approach.

1. Introduction

Cuny et al. [1] defined computational thinking (CT) as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." Recently, CT has emerged as a key skill set that can support problem solving in many areas. Perlis [2] strongly stated that algorithmizing would eventually happen in all fields, emphasizing the importance of the CT approach. Particularly in the time of increasing computing power, researchers have argued that CT can benefit almost everyone [3–5].

Particularly, researchers have focused on how CT could benefit students in educational settings [6–8]. Many studies have explored how CT education benefits K-12 students and what kinds of educational approaches can best serve them, whereas several other studies have focused on college students and explored how to effectively convey CT concepts to engineering students [9,10].

Despite the educational benefit of CT for engineering students, little research has explored whether the approach is effective for nonengineering students as well, who are the broader audience of CT education. Empirical studies that investigate the advantages of CT for that population are even scarcer. To prove that CT education can benefit problem solving in diverse areas, we believe that the field needs a better understanding of how CT education can influence nonengineering students and how best to teach them the core elements of CT.

In this paper, we describe the development of a curriculum for a capstone course, using core concepts and elements of CT. Specifically, we propose a business model-oriented CT framework in which students develop a business model to solve a real-world problem and clarify their goal through CT. We assumed that developing a business model would help students elucidate a specific goal and the procedure to achieve the goal. We then describe how we designed the capstone course in detail, based on the framework, followed by the project outcomes of the course. After interviewing students, we report on how they perceived their learning experience.

Our contributions include the following:

- We provide an empirical case study of a CT-based capstone course for nonengineering students
- We provide a detailed description of the course with several elements that ensured successful course outcomes
- We provide qualitative evidence of perceived learning outcomes related to CT education

This document is organized as follows: Section 2 outlines a literature review; Section 3 provides details about the methods we developed and applied; Section 4 summarizes the course description; Section 5 provides class project examples; and Section 6 gives the results of student feedback on the learning outcomes.

URI: https://hdl.handle.net/10125/60206
ISBN: 978-0-9981331-2-6

2. Literature review

2.1. CT frameworks as problem solving

Recently, CT has been considered a fundamental ability that is necessary for everyone in the era of data—a skill as important as reading, writing, and arithmetic. This ability refers to the thought processes involved in formulating a problem and expressing its solutions in computational steps that a computer can solve effectively [11]. Namely, CT involves problem formulation and problem solving [12]. Its benefits are not only effective software and hardware artifacts, but also an intellectual framework of thinking [3]. Thus, the mental process of structuring a problem to draw a computational solution is fairly important in CT [12]. In the original source of CT, Papert [13] explicitly mentioned CT as something that should apply to our daily lives without elaboration.

Despite the universal benefits of CT in problem solving, no single framework to examine CT yet exists [11,14]. Brennan and Resnick's framework [14] for investigating CT has three dimensions: concepts, practices, and perspectives. CT concepts provide the building parts required to express a problem and its solution computationally. CT concepts are common in many programming languages and include sequences, loops, parallelism, events, conditionals, operators, and data. CT practices concentrate on the process of thinking and learning, incorporating information, being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing. CT practices enable people to draw an adaptive, robust, but practical solution to complex problems based on CT concepts. This solution is the implementation of a solution expressed in CT perspectives. In CT perspectives, an individual can express an idea through CT and during this expression, make a creative artifact by social practices.

The Computer Science Teachers Association [15] argues that CT is composed of problem formulation for computational solutions, logical organization and analysis of data, abstractions, algorithmic thinking, assessment of efficiency and correctness, and generalization to other domains. The Computing at School subdivision of the British Computer Society [16] cites logical reasoning, algorithmic thinking, decomposition, generalization, patterns, abstraction, representation, and evaluation as CT components. The International Society for Technology in Education's Standards for Students [17] states that CT consists of leveraging the power of technological methods to develop and test solutions, collecting and analyzing data, representing data, decomposition, abstraction, algorithms, automation, testing, parallelization, and simulation.

Google [18] states that CT is "a problem solving process that includes a number of characteristics and dispositions." Google defines decomposition, pattern recognition, abstraction, and algorithm as cornerstones of CT. Decomposition is breaking down data, processes, or problems into smaller and manageable parts. Pattern recognition involves observing patterns, trends, and regularities in data. Abstraction refers to identifying the general principles that generate these patterns. Algorithm design is developing the step-by-step instructions for solving this and similar problems.

2.2. CT in educational settings

Researchers have explored the application of CT in educational settings [6–8]. Barr and Stephenson [6] discussed what is involved in bringing CT to K-12 students and the role of the computer science education community. Grover [7] conducted an extensive review of academic literature and examined the current state of discourse on CT in K-12 education, identifying gaps in research and practice. Yadav et al. [8] implemented and evaluated a CT module in a course for elementary and secondary education majors. They found that participating students' attitudes toward computer science became more favorable and they would be more likely to integrate computing principles in their future teaching.

Most studies have explored how CT education benefits K12 students and what kinds of educational approaches might be most effective. Several studies have focused on college students and explored how to effectively teach CT concepts to engineering students [9,10]. A few researchers have also explored the effect of CT on nonengineering students [19,20]. Adams [19] presented a proposal for creating an interdisciplinary computational science major within the liberal arts. Pulimood et al. [20] described a collaboration between computer science and journalism students and professors at a college and a large metropolitan newspaper. They demonstrated that when computer scientists and journalists connect across disciplinary boundaries, CT and collaboration can solve a real problem and have a substantive impact on society.

Still, empirical studies that investigate the advantages of CT for nonengineering students are rare. In this study, we aimed to explore how CT education can benefit social problem solving and what kind of learning outcomes can be achieved for nonengineering students.

3. Business model-oriented CT framework

For educational purposes, we propose a business model-oriented CT framework in which students develop a business model to solve a real-world problem and clarify their goal to achieve it through CT processes and practices. We assumed that developing a business model would help students clarify their specific goals and the procedures necessary to achieve said goal, thereby facilitating the problem-solving process.

3.1. A two-stage compiler for CT

CT is a conceptual framework that enables programming [21]. We consider CT to be a compiler that transforms a problem-solving idea into computer-friendly logic, called an implementable idea. This allows a person to easily develop a software product or information technology service from a rough idea. CT can extract important elements in an idea, decompose an idea into small parts, recognize similar parts that can reduce repetitive jobs, and translate an idea into computationally executable instructions.

Software and hardware development processes act as another compiler that allows for developing an execution idea for a software product or service that realizes an implementable idea. This compiler is based on knowledge of software and hardware. The implementable idea is transformed into an execution idea that is specialized for specific software and hardware. Figure 1 shows a T-diagram that illustrates the transformation from a source idea (left of T) to a target idea (right of T), realized via an implementation process (inside of T) by an operator (bottom of T). Here, operators are nonengineers who have different levels of understanding of CT, hardware, and software.

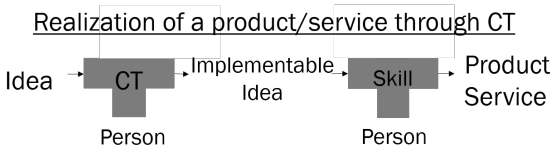


Figure 1. Compilation of an idea for generating a product or a service through CT.

3.2. Enhancing CT by coupling with a business model

A business model specifies how a company creates value for itself while delivering products or services to customers. A business model can be decomposed into value proposition, customer segments, channels, customer relationship, revenue streams, key resources, key partnership, key activities, and cost structure. The value proposition states the problem to be solved explicitly, which will satisfy a customer need. This enables a problem solver to focus on the problem rather than its solution. In many cases, however, people tend to concentrate on the technology that is used for the solution rather than on the problem. In business, a problem or a need of a customer is often considered to be more important than how to solve said problem or how to address said need because the success of businesses often starts from finding the right problem. Another key element is the identification of customers. The archetype of the customers, including their geographic, social, and demographic features, is also crucial in business models. Distribution channels, customer relationship, revenue streams, key resources, key partnership, key activities, and cost structure are followed by customer segments to make businesses practical.

Business models are dependent on the need or the problem because meeting the needs of customers and solving their problems is important [22], as previously stated. Developing a business model helps narrow down potential solutions. Once a business model is ready, CT makes the model concrete in terms of implementation and realization by specializing in certain technologies. From the CT perspectives of Brennan and Resnick [14], a business model conceived as an idea can be compiled and expressed as an implementable idea through CT. Then, the implementable idea can be realized through CT concepts and CT practices to become a product or a service. Figure 2 illustrates how business models and CT supplement each other in the context of problem solving, transforming from a problem to an idea, from an idea to a business model, and from a business model to a tangible product or service via CT.

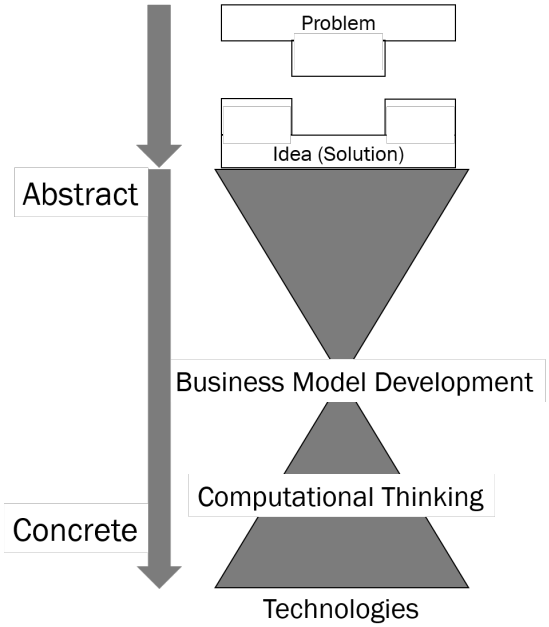


Figure 2. Our framework for problem solving by combining business model development and CT.

4. Course description

In this section, we describe the course curriculum we developed based on the framework explained in the previous section. The course was conducted for 14 weeks during fall 2017. One of the authors of this paper taught the course, and its participants included 2nd-, 3rd-, and 4th-year undergraduate students from the Humanities, Arts, and Social Sciences Division of Underwood International College at Yonsei University.

The course, called the Social Innovation Capstone Project, was designed to encourage students to build business models for tackling social problems. The students implemented the models by developing software on computing devices such as Arduino and Raspberry Pi. To do so, the students learned about business model development and CT by tinkering with Python, Arduino, Raspberry Pi, sensors, and actuators.

4.1. Preworkshop

To help students with no technical background, we held a 5-day workshop that covered the basics of computer programming—using Python, Arduino, Raspberry Pi, sensors, and actuators—and business model development before the semester started. During the workshop, students learned how to read code written in the C and Python programming languages so that they could read and manipulate existing code. Students reviewed several examples of Python, Arduino, and Raspberry Pi code to learn CT concepts such as loops, conditionals, operators, and data. For instance, students learned how to connect a humidity sensor to an Arduino and monitor the numbers generated by the sensor. They also learned that analog sensors work similarly to the humidity sensor. The students had the opportunity to conceptualize each example as a unit module that could be used as a building block for a more complex solution.

4.2. Team organization

Because most students did not have a technical background, such as software engineering and programming, we decided to employ a tutoring model and organized teams accordingly. The teams were formed by the instructor and each team had three or four members, one of whom assumed the role of a leader (i.e., tutor) who could help their teammates understand the course materials aside from help from the instructor. Owing to the intense course schedule and the difficulty of learning a programming language in a short time, it seemed daunting for a course instructor to effectively deliver learning materials to the entire class. Each team leader had experience in programming of approximately 12 months and had also completed 2 weeks of programming training before the course with the instructor. Throughout the semester, the leader helped the other team members understand technical concepts and implement the product. This pyramid scheme for learning, as shown in Figure 3, enables the initial learners to acquire CT through intimate face-to-face tutoring, which has proven to be effective for learning concepts [23], as stated by Brennan and Resnick [14]. Besides, the leaders who played the tutoring roles learned CT more effectively by preparing to teach [24] and by teaching [25,26].

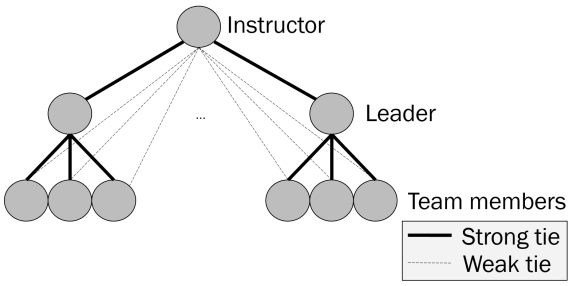


Figure 3. Pyramid scheme for tutoring

4.3. Key learning objectives

Throughout the capstone course, we had four key learning objectives—learning hardware, learning software, developing a business model, and building an internet of things (IoT) solution.

4.3.1. Learning hardware. Students learned how to use sensors and actuators. For example, sensors for temperature, humidity, and pressure and actuators for motion and sound were covered. Additionally, students learned how to control a camera. Because sensors and actuators require computing units to be controlled, students also learned how to use the Raspberry Pi and Arduino platforms. By doing this, students became aware of the connections between functions that may require physical actions and the relevant hardware that can carry out such functions.

4.3.2. Learning software. In addition, students learned how to communicate with the hardware computing units via a computer programming language, such as Python and C. While working in the Linux operating system, students could employ useful example code, such as image recognition, text-to-speech, and speech-to-text programs from OpenCV, TensorFlow, Google API, and other open-source projects. The students learned the necessary functions that could be implemented in software.

4.3.3. Developing a business model. After determining what functions could be realized using hardware and software via CT, the students developed a business model to solve a social problem. Then they compiled the business model via CT to transform it to an idea implementable on hardware and software. The social problem was decomposed into small problems that were a manageable size while focusing on the essence of the problem. Moreover, the students built their business models using building blocks that were conceptualized as functional units in the previous steps. Business models were developed using the business model canvas proposed by Osterwalder and Pigneur [27]. The examples dealt with in Steps 1 and 2 may not be sufficient to ensure the idea would be realized. Thus, through CT, the students learned how to use additional hardware and software as needed.

4.3.4. Building a social IoT solution. Once their business model was ready, the students implemented the relevant functions that constituted the business model using the building blocks of hardware and software they had learned. This enabled them to build an IoT solution for a social problem. An IoT solution is an artifact that can be evaluated. By considering not only the tangible solution itself but also the process required to reach the solution, we measured pre-experience, overall learning experience, team collaboration, and learning outcomes, including creativity and self-efficacy.

4.4. Deliverables and assessment

Throughout the course, students were required to provide (a) a proposal for a team project, (b) a poster to summarize the team project when the project was completed, (c) a prototype of the product or service, (d) an oral presentation, and (e) a final report to illustrate the team project. These deliverables were planned to enable students to set their own goals and illustrate how they envisioned achieving those goals in detail.

The main assessment criteria included the simplicity of the goal of a project and the alignment between functions and hardware and software components to achieve the goal. The delivery effectiveness was also evaluated.

5. Capstone project outcomes

During the semester, the course yielded five IoT services for social problems and all teams created a working prototype using various technologies. We present them with detailed description here.

5.1. Canary: a smart guide clip for people with visual impairment

The team focused on deteriorated or missing tactile pavement around the university's campus that makes it difficult for people with visual impairment to walk. Through interviews with potential users, the team came up with an IoT product called Canary, which is a smart guide clip for visually impaired individuals. It has three main functions: (a) detecting and classifying the type of nearby pavement blocks and aurally notifying the user; (b) detecting and classifying vertical transportation, including stairs, escalators, and elevators; and (c) collecting service request data from the Canary device and transferring this data to city officials. Using Canary, users can safely walk around and easily report problems if needed. Moreover, they no longer require standard white canes anymore, enhancing their mobility.

As for the technologies used, the team employed Open CV for distance estimation, TensorFlow for deep learning, Google Maps API for location detection, and Cloud storage for big data storage.

1. Detecting and classifying the type of nearby pavement blocks, and aurally notifying the user

2. Detecting and classifying the vertical transportation, including stairs, escalators, and elevators

3. Detecting the service request data from the users' Canary and transferring this data to the city council

Figure 4. Canary key functions

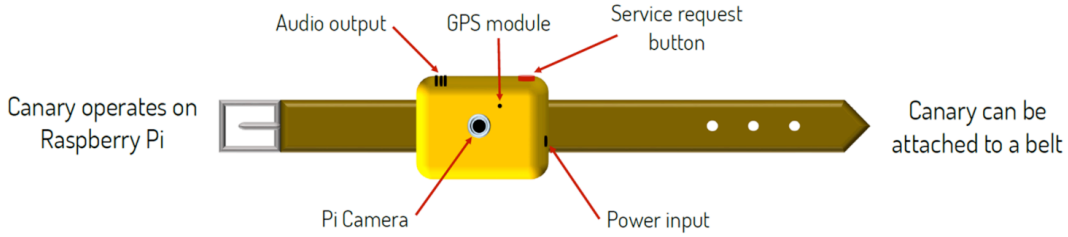


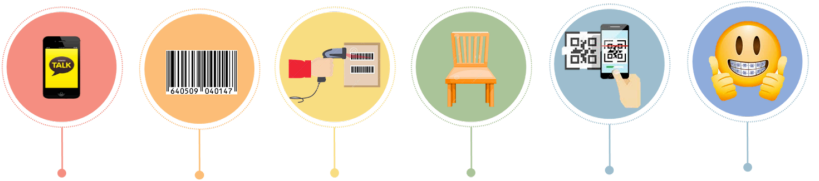
Figure 5. Canary prototype

5.2. PPUCHA: a prenatal public chair for expecting mothers in subways

PPUCHA (which stands for Prenatal PUblic CHAir) was developed to help alleviate some of the struggles pregnant women face in the public sphere, particularly on subways. Many argue that the current priority seats are inefficient because they prevent people from sitting down when there are no pregnant women on board. To address these concerns, the developers of PPUCHA set out to design a system that allows nonpregnant individuals to use those seats yet induces them to yield their seat when a pregnant woman boards the train. PPUCHA uses the KakaoTalk Chatbot/Plus Friend system and barcode scanners to ensure that expecting mothers have a place to sit. The media content that plays in the back "socially shames" nonpregnant users to give up their seats when a pregnant user scans their codes. In addition, the development team of PPUCHA created a website (www.ppucha.herokuapp.com), which can be accessed via a QR code, to share information on health care, laws, and social benefits in place for pregnant women.

5.3. URO-TICTOC: a service for seniors experiencing nocturia

URO-TICTOC aims to provide a solution for seniors experiencing nocturia. Nocturia is a frequent need to arise during the night to urinate. Approximately 70% to 80% of the population aged 65 or older have severe symptoms or the potential of experiencing this disease. Nocturia often results in sleep deprivation, impaired cognitive functions, depression, additional accidents, and secondary impact on family members. Nocturia is caused by numerous factors, and urologists focus on measuring the patient's



1. Add 'PPUCHA' as a friend on KakaoTalk, a chatting app

2. Get barcode vis the app

3. Scan barcode on the scanner next to the door

4. Sit down on PPUCHA

5. Scan QR code onto mobile device

6. Enjoy your ride on PPUCHA

Figure 6. PPUCHA prototype and key functions

volume of urine and frequency of nighttime visits to the bathroom. URO-TICTOC automatically calculates these factors for seniors who face difficulty taking those measurements during the night, ensuring they can provide accurate medical information to doctors more conveniently.

Figure 7. UROTICTOC prototype

5.5. Hangu: an American Sign Language learning device for children

Hangu is an American Sign Language learning device for children. It is connectable to computers and electronic devices, making it portable and accessible. Users can practice alphabets in sign language, form words with alphabets learned, review learning materials, and test learning materials.

Hangu is unique in that its users can have fun playing with 3D-motion programs, develop bilinguality and creativity, and easily carry it around.

5.6. Ga-UI: an augmented mirror for family communication

Ga-UI is an augmented form of a common mirror, with practical functions that can bring family members closer and restore a family relationship. The primary functions of the mirror include allowing access to each family member's personal schedule and sending text messages and external links to family members via a chat service installed in the mirror. The schedule information is shared via iCal, the API of which is installed in the Ga-UI. Family members can check one another's schedule displayed on the mirror screen and develop a better idea of their daily lives. Other functions include weather and temperature alarms and indicators. By providing weather, temperature, and humidity information imported from a national weather service API, the system allows users to obtain useful information before leaving the home.

6. Evaluation

6.1. Methods

To examine the learning experiences from the learners' point of view, we conducted in-depth interviews with 13 students who participated in the course. All interviews were conducted face-to-face and each session lasted approximately 45–60 minutes. Although these were semistructured interviews, we developed planned questions to make sure to cover important topics, such as (a) how they perceived the overall learning experience, (b) what kind of learning outcomes they achieved, (c) what features they liked about the course, and (d) how the course can be improved.

In addition to interviewers taking some notes during the interview sessions, all conversations were audio recorded for further analysis if permission was granted by the participants. The interviews were transcribed and analyzed using a general qualitative analysis technique [28], borrowing techniques from grounded theory [29,30]. This approach relies on detailed readings of raw data to derive themes relevant to the objectives. After skimming through the transcribed texts, we determined the initial coding schemes (i.e., core themes). We then carefully examined the transcript to find data fitting each coding scheme. Because the interview guide already had core concepts and themes, we focused more on disentangling the phenomena and relationships behind the users' experience with the capstone course. Once we went through the first round of analysis, we modified the coding schemes and repeated the process again.

6.2. Results: how did the students perceive their learning experience?

Although we did not systematically evaluate the learning outcomes of students in a quantitative manner, students reported their learning experiences, focusing on the skills they learned. Overall, students perceived that they acquired numerous technical and implementation skills through the course. Creativity and awareness of broad applicability of computing were also mentioned.

6.2.1. Technical skills. Most students stated that they acquired a lot of technical skills throughout the course, mainly because they had little knowledge of technical skills prior to class. Students mentioned that they gained technical knowledge mostly from the workshop, in which they had an opportunity to learn about different IoT technologies. Then, by applying

those technologies to their own project, they were able to refine their knowledge.

Interestingly, even when students did not actively participate in technical tasks, they still said that their skills had improved simply because they became familiar with the concepts and their emotional barrier to learning had been lowered. Moreover, leaders seemed to have learned a great deal of technical skills by teaching and mentoring team members.

"I didn't make huge progress, but in other programming classes, I used to be scared when I saw lines of code and couldn't do anything. But this time I think I've become more confident. You know, those who are good at programming do a Google search when they're stuck. I was like, 'Wow, how could he do that?' But now I think I can do that. Because I didn't take charge of technical stuff, I couldn't develop my skills that much, but I can do that at least."

"The reason why I wasn't scared? Maybe because we used real devices and actually made something with them. It was real, not imaginary or virtual. You could touch it, and it felt so different. It's like when you calculate using fingers, not mentally. You can easily see it, experiment it, and then it gets familiar."

"At the workshop I learned how to code and it was my first time. To be honest, I didn't understand everything, although I followed all the steps without difficulty. Rather than understanding the basics first and applying it, it was more like learning by doing. Although I didn't understand the full lines of code, I learned what kind of modules are needed for this kind of code."

6.2.2. Implementation skills. One of the skills most acquired in the course involved implementation, which is the ability to combine various technologies and turn them into a working product. Even though the students' knowledge was still limited and imperfect, they mentioned that they at least got to know where to start their research if they wanted to build something. That is, even when they are not sure of how to connect the dots, they might know what the dots represent and start from there.

"My implementation skills were improved a lot. At each decision point, we had lots of discussions and refined our prototype over and over again. Now I can imagine how to make something in my head. Now I know where to start researching, at least."

"I guess I learned that part. For example, we decided what sensor to use after a group discussion—what would be good to use, where we want to use a specific sensor, etc. Although I don't know every detail of each sensor, I know what to study further if I need something later."

6.2.3. Creativity. Some may argue that students may feel limited when they have to narrow down the scope of their project in the pattern-matching stage. In our study, however, it turned out that a CT-based capstone course could possibly enhance learners' perceived creativity level. For example, many students commented that they practiced creativity throughout this course, training themselves to be more creative. They defined creativity as "changes in perspective" rather than as "new and different," the latter of which is the classic definition of creativity.

"You may directly go to the solution using different tools, but I kept thinking, 'How can I make this differently even if I take the long way?' And I like that, because that's my definition of creativity—exploring various ways to go from A to B. If there's the only one answer, there would be only one way. But this project didn't have any right or wrong answer, so I had to practice my creativity all the time."

"Well, you may think it's limiting because you actually have to make it and need to be practical. You have limited resources. But I believe creativity is creating the best outcome in a given situation, isn't it?"

"Definitely I learned lots of creativity skills because we made something out of nothing [laugh]. To come up with ideas, I had to think in a different way all the time and that was an eye-opening experience. Now, in everyday life, I always observe things more carefully. What are the problems? Why do they exist? How can we improve that? I somehow got to know how to think for creativity."

6.2.4. Awareness of broad applicability of computing. According to the Association for Computing Machinery curriculum guidelines for undergraduate computer science programs, computer science graduates are expected to have awareness of the broad applicability of computing [31]. That is, students need to understand that computer applications affect nearly every aspect of our lives and many opportunities are available in computing. Students who participated in the CT capstone course stated that they began to truly understand that computing could be applied everywhere by exploring the various possibilities for a solution and also by observing the work of other teams.

"To me, before this class, a device was just a device. I've never thought about what kinds of things are possible using a certain device. After I saw that all types of different applications were made using the same device, I realized how a simple technology can make changes in my everyday life. In a word, my perspective has changed."

"Making is very, very different from listening or seeing. Once you go through the making process and

understand the process, you get to know what is possible. Your thinking, your perspective becomes broader."

"I've heard about the internet of things, Alphago, but I've never thought about how those things are relevant to my everyday life. Through this course, I learned how technologies can be applied to our daily lives and that was impressive. I would never have realized that if I had heard about it through news articles or lectures."

6.3. Discussion: why does this approach work?

In addition to the students' testimonials on their learning experiences, the teams' final work was presented at an exhibition and received lots of attention and positive feedback. In particular, people were surprised by the fact that the prototypes were created by nonengineering students.

Several factors facilitated successful implementation of this course. The first is our interdisciplinary approach that combined a business model and CT. By encouraging students to think about the problem and solution and its stakeholders before creating a prototyping, the curriculum allowed students to clearly identify each step to realize the solution, thereby supporting the prototyping stage in turn. The second is our learning-by-doing approach, in which every team was forced to create a working prototype as a final deliverable. Although nonengineering students often worry about technical difficulties, the experience of making some form of working prototype helped students overcome that initial barrier and obtain the greatest learning benefit out of the course. The third ingredient for success is the pyramid team structure, using a tutoring model. Without leaders' help, it would have been difficult for teams to complete the project because one instructor likely could not have delivered all the required knowledge, given that most students did not have a basic technical background. By having tutors receive pretraining and help the rest of team members with technical implementation, the learning outcomes were maximized, although the degree of skills acquired might vary across individuals.

7. Conclusions

In this paper, we described the development of a curriculum for a social innovation capstone course, designed for undergraduate students of a liberal arts college at a university in Korea. We combined a business model and CT in this course to take a holistic approach to problem-solving education. Using various technologies including Raspberry Pi and Arduino, students defined a social problem, came up with a solution, built a business model, and implemented a working prototype. Class project outcomes proved the effectiveness of our approach and students positively perceived the learning outcome of the course.

Our study had several limitations. First, we observed only the specific case of a CT capstone course with a relatively small number of students. Thus, it might be hard to generalize the results of our study. The course, although conducted in English, took place in a particular cultural context that might have affected our results. Possible future research includes a carefully designed experimental approach that uses a larger sample size with different variables, such as age, gender, level of technical background, and quantified learning outcomes. Moreover, it might be interesting to see how different learning contexts, including team formation, evaluation criteria, and resource accessibility, affect learning experiences.

8. Acknowledgement

This research was supported by Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government (MOTIE) (N0001436, The Competency Development Program for Industry Specialist).

9. References

[1] Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.

[2] Perlis, A. J., & Thornton, C. (1960). Symbol manipulation by threaded lists. Communications of the ACM, 3(4), 195-204.

[3] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

[4] Wing, J. (2011). Research notebook: Computational thinking—What and why? The Link Magazine, Spring.

[5] Wing, J. M. (2016). Computational thinking, 10 years later. Microsoft Research Blog. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later.

[6] Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? Acm Inroads, 2(1), 48-54.

[7] Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

[8] Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011, March). Introducing computational thinking in education courses. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 465-470). ACM.

[9] Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. ACM SIGCSE Bulletin, 41(1), 183-187.

[10] Shell, D. F., Hazley, M. P., Soh, L. K., Miller, L. D., Chiriacescu, V., & Ingraham, E. (2014, October). Improving learning of computational thinking using computational creativity exercises in a college CSI computer science course for engineers. In Frontiers in Education Conference, 2014 IEEE (pp. 1-7). IEEE.

[11] Denning, P. J. (2017). Remaining trouble spots with computational thinking. Communications of the ACM, 60(6), 33-39.

[12] Wing, J. (2014). Computational thinking benefits society. 40th Anniversary Blog of Social Issues in Computing, 2014.

[13] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.

[14] Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada (pp. 1-25).

[15] Computer Science Teachers Association. (2016). Operational definition of computational thinking.

[16] Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers. Google Scholar.

[17] Computer Science Teachers Association. (2011). Operational definition of computational thinking for k-12 education.

[18] Google. (2018). Exploring computational thinking. https://edu.google.com.ph/resources/programs/exploring-computational-thinking/

[19] Adams, J. B. (2008). Computational science as a twenty-first century discipline in the liberal arts. Journal of Computing Sciences in Colleges, 23(5), 15-23.

[20] Pulimood, S. M., Shaw, D., & Lounsberry, E. (2011, March). Gumshoe: A model for undergraduate computational journalism education. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 529-534). ACM.

[21] Denning, P. J. (2017). Remaining trouble spots with computational thinking. Communications of the ACM, 60(6), 33-39.

[22] Dorst, K., & Cross, N. (2001). Creativity in the design process: Co-evolution of problem–solution. Design Studies, 22(5), 425-437.

[23] Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. IEEE Software, 17(4), 19-25.

[24] Muis, K. R., Psaradellis, C., Chevrier, M., Di Leo, I., & Lajoie, S. P. (2016). Learning by preparing to teach: Fostering self-regulatory processes and achievement during complex mathematics problem solving. Journal of Educational Psychology, 108(4), 474-492.

[25] Bargh, J. A., & Schul, Y. (1980). On the cognitive benefits of teaching. Journal of Educational Psychology, 72(5), 593.

[26] Hollingsworth, S. (1989). Prior beliefs and cognitive change in learning to teach. American Educational Research Journal, 26(2), 160-189.

[27] Osterwalder, A., & Pigneur, Y. (2010). Business model generation: A handbook for visionaries, game changers, and challengers. John Wiley & Sons.

[28] Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data. American Journal of Evaluation, 27(2), 237-246.

[29] Charmaz, K., & Belgrave, L. (2012). Qualitative interviewing and grounded theory analysis. In The SAGE Handbook of Interview Research: The Complexity of the Craft (Vol. 2, pp. 347-365).

[30] Strauss, A., & Corbin, J. (1998). Basics of Qualitative Research Techniques. Sage.

[31] Association for Computing Machinery. (2018). Curricula recommendations. https://www.acm.org/education/curricula-recommendations