Effects of the Use of CDIO Engineering Design in a Flipped Programming Course on Flow Experience, Cognitive Load

Hua-Xu Zhong¹, Po-Sheng Chiu²,* and Chin-Feng Lai¹,*

¹ Department of Engineering Science, National Cheng Kung University, Tainan City 701, Taiwan; k43122003@gmail.com
² Department of E-Learning Design and Management, National Chiayi University, Chiayi County 621, Taiwan
* Correspondence: chiups@mail.nctu.edu.tw (P.-S.C.); cinfonssci@gmail.com (C.-F.L.)

Abstract: The flipped teaching method has become increasingly mature and critical. Previous flow experience studies have concentrated on game-based learning, and cognitive load studies have concentrated on different types of teaching materials (e.g., video). Due to the characteristic differences between problem-based learning and the Conceive, Design, Implement, Operate (CDIO) engineering design, the authors were interested in applying the CDIO engineering design to the flipped programming course. This study was proposed to measure students’ cognitive load and flow experience by using CDIO engineering design in the flipped programming course, which used a one-group pretest–post-test nonequivalent-groups design method for 16 weeks. This study recruited 40 college students (males = 14, females = 26) who were first-year freshmen attending a university as its subjects. The results indicate that the students showed no significant improvement between cognitive load and gender difference in cognitive load and flow experience, but that they significantly improved some dimensions of flow experience. This study provides implications and evidence related to applying the CDIO engineering design in flipped programming courses.

Keywords: cognitive load; flow experience; CDIO engineering design; programming; flipped learning

1. Introduction

Due to the advancement of technology in recent years, the flipped teaching method has become increasingly mature and critical. As compared to lecture teaching, flipped learning is still a popular teaching method [1–3]. This teaching method has aroused our interest. Many studies have shown that flipped learning can provide learning benefits to students [4–7]. Flipped learning requires teachers to design an appropriate curriculum according to the different stages of education [8]. Teachers should provide a suitable flipped teaching method and learning materials as a sustainable approach to enhance students’ learning benefits [5]. In particular, the Çakıroğlu and Öztürk [9] study applies problem-based learning to the flipped programming language course based on a self-regulated learning theory in order to support the student programming course. Programming is an essential skill for students in the technological fields associated with computer science [10]. As the benefits of flipped teaching become apparent, teachers should appropriately design flipped teaching for programming courses. It is necessary to develop an appropriate flipped teaching method to help students develop their programming skills and knowledge.

The current programming curriculum is implemented by problem-based learning [9,11,12]. Problem-based learning (PBL) requires the teacher to provide problems that students can discuss in order to facilitate problem-solving and learning [11]. Conceive, Design, Implement, Operate (CDIO) engineering design framework is used to build knowledge and systematically innovate to enhance students’ learning skills and knowledge [13]. Contrary to PBL, CDIO engineering design emphasizes the content of what students need to learn—that is, the student learning outcomes [14]. In particular, Edström and Kolmos [14] identify the differences and nature of problem-based learning (PBL)
and CDIO engineering design and the complementary ways that they can support each other. Due to the differences between PBL and the CDIO engineering design, it would be interesting to apply the CDIO engineering design to the flipped programming course.

Moreover, Anmarkrud, and Andresen [15] indicate the importance of cognitive load theory to examine the potential benefits of using multimedia in learning. Current cognitive load research has been rapidly developing on online course videos in colleges, which investigates different ways of presenting multimedia information [16–18]. Further research on the impact of multimedia teaching materials in terms of cognitive load on students is needed.

In general, the process of flow experience occurs when one engages in online activities [19,20]. Chen and Wigand’s [20] study examined the impact of many flow experiences on internet activity, but due to the participants’ profiles, they did not have programming and visual-design-related evidence. Several studies indicate the importance of the flow experience in programming courses [21,22]. However, due to the characteristics of the game, the teacher must provide the appropriate challenge–skill balance to promote the student’s flow experience [23]. In addition, many studies apply game-based learning to programming courses in order to enhance students’ flow experience [10,22,24,25]. The characteristics of games have made game-based learning widely used in flow experience study. In this study, students are taught to program using flipped teaching, which influences their flow experience when writing programs. It would be interesting to apply flipped teaching methods to examine students’ flow experience and cognitive load.

However, previous research has explored gender differences in programming achievements [26], perceived ease of use and the intention to use [27] for students. Male students are more easily interested in programming than female students in a traditional programming environment. In particular, Rubio and Romero-Zaliz [27] use the physical principles of computing design to reduce gender differences in perceptual and learning outcomes. Moreover, current research lacks a method to reduce the gender difference in online courses or flipped programming courses. Therefore, it is important to note that the use of flipped teaching in programming courses can impact how students of different genders write their programs. Flipped teaching provides teaching materials that are critical to the cognitive load and mind-flow experience of different students of different genders. In particular, Parra-González and López Belmonte [6] indicated that exploring the gender differences in flipped teaching could be a new direction to explore in flipped teaching. As far as we know, no previous research has investigated the cognitive load and flow experiences in flipped teaching programming courses, which still need to be further clarified.

Presently, research has paid little attention to teaching methods that affect students’ flow experiences and reduce gender difference. In particular, components of PBL pedagogy are the defining characteristics of CDIO (design–implementation experience) [14]. Moreover, this study argues that the nature of CDIO engineering design (e.g., creativity) is different from PBL (e.g., problem-based learning) teaching. It may be an appropriate use of the CDIO engineering design that brings students new ways and attitudes (e.g., computational thinking attitudes [28]) for solving problems. Thus, in order to understand the effects of the CDIO engineering design of a flipped programming course on cognitive load and flow experience, the current study provides slides and related video and teaching material in the flipped programming course, which are expected to improve college students’ cognitive load and flow experience. Therefore, this study was proposed to develop students’ cognitive load and flow experience by using CDIO engineering design in flipped programming courses with the following four questions:

1) Do flipped CDIO programming courses have a different effect impact on cognitive load in college students?
2) Do flipped CDIO programming course have a different gender impact on cognitive load in college students?
3) Do flipped CDIO programming course have a different effect impact on flow experience in college students?
(4) Do flipped CDIO programming course have a different gender impact on flow experience in college students?

2. Literature Review

2.1. Flipped Learning

Flipped learning is defined as the implementation of interactive technology to learn basic information before the course, and implemented learning activities during the course by active learning [29,30]. Flipped teaching is not a panacea for all subjects who require teachers to develop a flipped curriculum to fit the learning needs of their students [31,32]. Some studies have shown that the benefits of flipping learning have a positive impact on overall learning [3,31–34]. Other studies, albeit fewer, have shown that flipped classrooms have no effects or negative impact on students’ attitudes [35,36]. Çakıroğlu and Öztürk’s [9] study indicates that flipped learning brings students active learning and learning benefits in a programming course. Although some flipped learning studies have shown the advantages and disadvantages of learning, there is still a lack of appropriate ways to implement it in programming courses.

2.2. Flow Experience

The concept of “flow” was first proposed by Csikszentmihalyi [37], and the factors associated with the flow experience have been summarized into nine elements. Flow experience is defined as the concentration on and enjoyment of an activity and an optimal and most enjoyable experience [38–40]. Flow experience should be considered a whole process or a psychological state that allows an individual to focus effortlessly on programming activities [41]. Currently, many studies use flipped teaching to investigate flow experience [42,43]. Wang and Hsu’s [44] research indicates that a balance of challenge and skill is a necessary prerequisite for flow experience. That is, flow experiences usually occur when the task is challenging and gives individuals the ability to face the challenge [45,46]. The advantages of flow experience are evident, as it improves students’ ability to learn and their attitude to learning [25].

2.3. Cognitive Load

In a learning setting, three different types of cognitive load can be identified: intrinsic cognitive load, extraneous cognitive load, and germane cognitive load [47]. Recently, Sweller and van Merriënboer’s [48] study summarized the last 20 years of cognitive load theory and mentioned the feasibility and main advantages of cognitive load. Cognitive load theory is commonly used as a theoretical framework in the learning and instruction of cognitive theory [49]. In general, the instructional design should reduce unnecessary information in order to reduce the working memory load to a manageable percentage [50]. In a multimedia learning environment, the two theories that are inherently mentioned are Cognitive Load Theory (CLT) [50] and Cognitive Theory of Multimedia Learning (CTML) [51]. In the understanding of the CLT and CTML, it is known that providing irrelevant information in visual or verbal form is not supposed to help learning [52]. Moreover, Mutlu-Bayraktar and Cosğun’s [53] study systematically reviewed articles from 2015 to 2019 that reported detailed information on cognitive load and multimedia learning, as well as understanding learners’ cognitive processes.

2.4. CDIO Engineering Design

CDIO teaching models fundamentally change the way students learn and how teachers teach [54]. The CDIO model is also used to teach different fields such as chemistry [54], design thinking [55], and game-based learning [56]. In particular, Taajamaa and Eskandari [55] adopt the Observe(O)-CDIO method to increase students’ tolerance of ambiguity in problems in order to come up with better solutions. Many studies have used the CDIO model to teach and help students learn, which has improved students’ learning process, communication skills [54,55,57,58], and attitude [54]. CDIO engineering design promotes
the integration of knowledge and practice among students, which helps them to practically learn and innovate products [54,57]. In sum, CDIO engineering design is an approach that supports students to integrate their knowledge and to make practical projects.

In this study, we carry out the CDIO engineering design framework revised as follows [13]:

1. Conceive: Divergence means observing problems from many perspectives.
   Define task: Observe the situation, identify common problems and rules, and distribute the information and resources needed to consider the solutions.
   Design concept: Use the flow chart as a preliminary observation problem; initially define the problem and design the solution.

2. Design: Convergence design solutions.
   Search for relevant information: confirm the required range of information, and use the keyword to query online in order to collect the required information.
   Detailed design: Use a mind map or flow chart to design a detailed solution and present the solution in different ways.

3. Implement: Obtain divergence development solutions.
   Preliminary development: Develop the design as a solution through a validation process.
   System integration and testing: Develop a complete solution for system integration and testing.

4. Operate: Conduct continuous convergence testing and correct solutions.
   The actual operational solution is a continuous revision of the solution until it is corrected to the best solution.

3. Methods

3.1. Develop Course Procedure

From March to June 2020, students used general flipped teaching and the flipped CDIO engineering design to learn various parts of the course, as shown in Table 1. In general flipped teaching, we used the Moodle learning platform for group learning, reading materials, and individual learning to develop a flipped course. In the classroom, teachers address questions and guided programming experiences to help students learn the programs. The design and development process of this flipped classroom was as follows.

Table 1. C# programming course.

| Week | Content                                      | Week | Content                                      |
|------|----------------------------------------------|------|----------------------------------------------|
| 1    | Introduction to the C# programming language | 9    | Array & Class (part1)                        |
| 2    | Console Application, Data type, Operator, basic I/O (part1) | 10   | Graphical User Interface                     |
| 3    | Console Application, Data type, Operator, basic I/O (part2) | 11   | Windows Form Controls (part1)                |
| 4    | Flow Control (part1)                        | 12   | Windows Form Controls (part2)                |
| 5    | Flow Control (part2)                        | 13   | File I/O                                     |
| 6    | Inherence & Exception (part1)               | 14   | Drawing                                      |
| 7    | Inherence & Exception (part2)               | 15   | Windows Presentation Foundation              |
| 8    | Midterm                                      | 16   | Final                                        |

1. Prepare online teaching materials: Teaching materials, including videos, additional reference materials, and web resources, were prepared, and relevant reading materials were developed. The videos were presented by using actual classroom situations and PowerPoint presentations. We used short videos to ensure that students are able to complete the program step-by-step. No questions or quizzes are used in this online programming course, which would prevent students from practically learning the program;
instead, we used programmed assignments. We want students to use trial and error to solve errors by practically writing programs. We did not use questions or quizzes to ensure that students learn to program.

2. Online collaborative learning: We used small groups and the Moodle platform to create relevant discussion forums and collaborative activities. In general, students watched the video after the class; students were required to watch the video, discuss problems, and complete the programming assignments. We asked students to write down the problems they encountered in writing their programs and the solutions they adopted for the discussion forum.

3. Real classroom: Students took their questions to the classroom and asked the teacher. The teacher discussed the forum questions, clarified the students’ problems, and opened up the discussion. Finally, group discussions were used to encourage students to discuss and draw conclusions.

4. Project work: According to the curriculum, students first prepared for their mid-term exams. Finally, teachers provided groups with a variety of projects for them to complete their final project.

Flipped CDIO engineering design is a teaching method that combines flipped teaching with CDIO engineering design. Flipped CDIO engineering design has many advantages over traditional flipped teaching. First, we used learning sheets to help students become familiar with CDIO engineering design and to help them apply it to their learning. Second, we used videos to explain CDIO engineering design concepts and applications, which included teaching students how to apply CDIO engineering design to learning (e.g., group learning, reading materials) and programming examples. Finally, we guided students towards related CDIO engineering design problems and explained programming in the classroom.

This study used CDIO engineering design instead of the general PBL approach, which is different from the problem-centered learning of the PBL approach. Students can use the CDIO process of diffusion and convergence to think about problems both in a divergence way and in a convergence way to solve them. CDIO engineering design can change the way students learn and think.

3.2. The CDIO Engineering Design in the Flipped Programming Course

The CDIO engineering design method was used in this study to help students learn program lessons and practice program projects. CDIO engineering design is characterized by conceiving, design, implement, and operation of the learning process, which helps students to learn programs in innovative ways. The CDIO engineering design was presented by a flipped teaching method, which helps students to learn CDIO engineering design effectively.

The flipped CDIO engineering design learning process was presented as follows. (1) First, a group was composed of three to six students who were randomly grouped. (2) Students watched a video and discussed the learning content through flipped learning. (3) The teacher provided appropriate learning worksheets to help students master conceptual knowledge. (4) CDIO engineering design instruction was implemented to help students write programs using CDIO engineering design. (5) The teacher provides different projects for each group of students so that students can actually apply CDIO engineering design. The flipped CDIO engineering design framework is shown in Figure 1.

3.3. Sample

This study adopted convenience sampling and considered 40 students (males = 11, females = 29) who were first-year freshmen in the Department of E-learning Design and Management at a teacher college in a certain university as its subjects. All of the students (19–20 years old) had e-learning-related backgrounds in a teachers college. They did not have programming or CDIO engineering design prior knowledge or related learning experience.
The flipped CDIO engineering design framework is shown in Figure 1.

### 3.3. Sample

This study adopted convenience sampling and considered 40 students (males = 11, females = 29) who were first-year freshmen in the Department of E-learning Design and Management at a teacher college in a certain university as its subjects. All of the students (19–20 years old) had e-learning-related backgrounds in a teachers college. They did not have programming or CDIO engineering design prior knowledge or related learning experience.

### 3.4. Research Design

In order to help students evaluate the cognitive load and flow experience for the flipped CDIO programming course, which used a one-group pretest–post-test nonequivalent-groups design method, as shown in Figure 2. The independent variables included teaching method and gender, and the dependent variables were cognitive load and flow experience. Thus, in order to avoid the control variables (e.g., teaching time and teaching quality) from impacting the study, all of the students were taught by the teacher who had programming and the CDIO engineering design experience.

Figure 1. Flipped learning framework.

In this study, videos and teaching materials were developed for a flipped CDIO programming course on how to programming using the CDIO engineering design steps in the pre-course, as shown in Figure 3.

Figure 2. Experimental procedure.

In this study, videos and teaching materials were developed for a flipped CDIO programming course on how to programming using the CDIO engineering design steps in the pre-course, as shown in Figure 3.
In this study, videos and teaching materials were developed for a flipped CDIO engineering design each took 7 weeks. Two questionnaires were given to students in the mid-term and final weeks, which took one week. The detailed course content used is shown in the following.

### 3.5. Data Collection

This study aimed to evaluate the students’ cognitive load and flow experience before and after the course, which were given in the midterm and final weeks. The Cognitive Load Questionnaire and Flow Questionnaire were used as the data collection tools in the present study. We cannot assume that our samples were normally distributed, so we used the Kolmogorov–Smirnov test. Results show that all samples had the characteristics of a normal distribution. Therefore, we used the t-test to analyse our questionnaire data.

This study adopted a cognitive load questionnaire [59] with five-point Likert-type scale (from strongly disagree = 1, to 3 (neutral), to strongly agree = 5), and was used to measure flow experience, including intrinsic, extraneous, and germane, in the course. The reliabilities of the dimensions of intrinsic, extraneous, and germane were 0.78, 0.82 and 0.90, respectively. The validation was examined using the Kaiser–Mayer–Olkin (KMO = 0.77) method and Barlett’s test of sphericity (393.90, \( p < 0.001 \)). Overall, the total accumulated variance of the three factors were 73.19%.

This study adopted a flow questionnaire [41] and was used to measure flow experience, including the control, engagement, and enjoyment in the course. A five-point Likert-type scale was used, ranging from 1 (strongly disagree) to 5 (strongly agree). The reliabilities of the dimensions of enjoyment (measures students’ levels of enjoyment from the programming), engagement (measures students’ levels of concentration toward the programming), and control (measures students’ perceived degrees of control of the programming environment) were 0.82, 0.87 and 0.79, respectively. The validation was examined using the Kaiser–Mayer–Olkin (KMO = 0.78) method and Barlett’s test of sphericity (297.56, \( p < 0.001 \)). Overall, the total accumulated variance of the three factors was 72.19%.

After administering the questionnaire, we checked for lack of completion and valid completion. Three invalid questionnaires were found and removed. The number of valid questionnaires obtained for the study was 40.

After revising the two questionnaires, two professors who have a programming educational background reviewed and edited the content to verify and modify the appropriateness of the items and provide expert validity. The questionnaire was found to have expert validity after revision by two professors with a programming educational background.

**Figure 3.** Experimental procedure.

The experimental procedure lasted 16 weeks for 150 min a week, as shown in Table 1. The flipped learning and flipped CDIO engineering design each took 7 weeks. Two questionnaires were given to students in the mid-term and final weeks, which took one week. The detailed course content used is shown in the following.
4. Results

4.1. Cognitive Load

The paired sample t-test was used to complete the pretest and post-test comparison. An independent sample’s t-tests were conducted to determine whether the males and females were different from each other. The study aimed to understand the effects of cognitive load and flow experience on programming courses. The pretest group used a flipped programming course, and the post-test group used a flipped CDIO programming course.

4.1.1. Results of the Paired Sample T-Test on Cognitive Load

Results indicated no significant difference in the intrinsic dimension decreased by 0.04 points (pre-test M = 2.77, SD = 0.54; post-test M = 2.73, SD = 0.67; t = 0.28, p > 0.05), the extraneous dimension decreased by 0.06 points (pre-test M = 2.04, SD = 0.57; post-test M = 1.98, SD = 0.71; t = 0.39, p > 0.05), and the germane dimension increased by 0.05 points (pre-test M = 3.83, SD = 0.52; post-test M = 3.88, SD = 0.45; t = −0.05, p > 0.05), as shown in Table 2.

| Dimension     | Pre-Test M/SD | Post-Test M/SD | Pre Score-Post Score | t    | p   |
|---------------|---------------|----------------|----------------------|------|-----|
| Intrinsic     | 2.77/0.54     | 2.73/0.67      | 0.04                 | 0.28 | 0.77|
| Extraneous    | 2.04/0.57     | 1.98/0.71      | 0.06                 | 0.39 | 0.69|
| Germane       | 3.83/0.52     | 3.88/0.45      | −0.05                | −0.41| 0.67|

p < 0.05.

4.1.2. Results of the Independent Sample t-Test on Cognitive Load

Results indicate the flipped CDIO programming course had no significant difference, but the germane dimension in the female group score was higher than in the male group (t = −0.08, p = 0.56 > 0.05), and intrinsic (t = 0.32, p = 0.74 > 0.05) and extraneous (t = 0.19, p = 0.84 > 0.05) dimensions in the male group score were higher than in the female group, as shown in Table 3.

| Dimension     | Male M/SD     | Female M/SD    | Levene’s Test for Equality | df   | Mean Difference | t    | Sig (2-Tailer) |
|---------------|---------------|----------------|---------------------------|------|----------------|------|----------------|
|               | (N = 14)      | (N = 26)       |                           |      |                |      |                |
| Intrinsic     | 2.78/.84      | 2.71/.57       | 2.62                      | 38   | 0.11           | 0.07 | 0.74           |
| Extraneous    | 2.01/.88      | 1.97/.62       | 1.39                      | 38   | 0.24           | 0.04 | 0.19           |
| Germane       | 3.82/.49      | 3.91/.43       | 0.32                      | 38   | 0.57           | −0.08| −0.57          |

p < 0.05.

4.2. Flow Experience

4.2.1. Results of the Paired Sample t-Test on Flow Experience

Results indicated no significant difference in the enjoyment increased by 0.28 of a point (pre-test M = 3.80, SD = 0.76; post-test M = 4.08, SD = 0.61; t = −1.17, p > 0.05). The results indicated a significant difference in the engagement increased by 0.37 of a point (pre-test M = 3.38, SD = 0.61; post-test M = 3.75, SD = 0.72; t = −2.43, p < 0.05), and flow control increased by 0.39 of a point (pre-test M = 3.32, SD = 0.74; post-test M = 3.71, SD = 0.60; t = −2.41, p < 0.05), as shown in Table 4.
Table 4. Results of the paired sample t-test on flow experience.

| Dimension       | Pre-Test M/SD | Post-Test M/SD | Pre Score-Post Score | t    | p   |
|-----------------|---------------|----------------|----------------------|------|-----|
| Enjoyment       | 3.80/0.76     | 4.08/0.61      | −0.28                | −1.17 | 0.08 |
| Engagement      | 3.38/0.61     | 3.75/0.72      | −0.37                | −2.43 | 0.02 *|
| Flow control    | 3.32/0.74     | 3.71/0.60      | −0.39                | −2.41 | 0.02 *|

* p < 0.05.

4.2.2. Results of the Independent Sample t-Test on Flow Experience

Results indicated the flipped CDIO programming course had no significant difference, but the germane dimension in the female group score was higher than the male group (t = −0.08, p = 0.56 > 0.05), and intrinsic (t = 0.32, p = 0.74 > 0.05) and extraneous (t = 0.19, p = 0.84 > 0.05) dimensions in the male group score were higher than in the female group, as shown in Table 5.

Table 5. Results of the independent sample t-test on flow experience.

| Dimension       | Male (N = 14) M/SD | Female (N = 26) M/SD | Levene’s Test for Equality | df   | Mean Difference | t    | Sig (2-Tailer) |
|-----------------|---------------------|----------------------|----------------------------|------|----------------|------|---------------|
| Enjoyment       | 3.96/.74            | 4.15/.52             | 3.35 0.07                  | 38   | −0.18          | −0.93| 0.35          |
| Engagement      | 3.69/.97            | 3.77/.56             | 3.84 0.06                  | 38   | −0.08          | −0.33| 0.73          |
| Flow control    | 3.94/.64            | 3.59/.55             | 0.53 0.47                  | 38   | 0.35           | 1.79 | 0.08          |

*p < 0.05.

5. Discussion

This study proposed the CDIO engineering design in a flipped programming course in order to enhance the effect of flow experience and reduce the cognitive load. The result indicated that students experienced no significant improvement of cognitive load or gender difference in cognitive load and flow experience, but indicated significant improvement in some dimensions of flow experience.

5.1. The Cognitive Load of Using the CDIO Engineering Design

Although the results demonstrate that cognitive load showed no significant difference between the pre-test and post-test, they also imply some teaching method problems in cognitive load. The use of innovative teaching methods can engage more learning benefits, which may also cause additional cognitive load for students [60]. The type of cognitive load generated by a teaching method is determined by the teaching method associated with a particular instructional goal [61]. Therefore, teachers need to adjust their teaching design to reduce the external cognitive load of students in order to help them learn [62,63]. Currently, from this study results indicated that no significant difference between pre-test (flipped learning) and post-test (flipped CDIO engineering design) in cognitive load. In other words, the CDIO engineering design method may not be appropriate for some instructional goals of the programming course. It may be possible that text programming language environments (e.g., C#, python, etc.) have complex programming functions that make it difficult to reduce the cognitive load. This contrasts with other teaching methods (PBL and inquiry-based learning), which make extensive use of scaffolding, thereby reducing the cognitive load and allowing students to learn complex subjects [64]. However, this may imply that the flipped CDIO engineering design approach does not have an explicit goal or approach to solving the problem. Flipped CDIO engineering design incorporates active and engaging learning approaches [54]. Students need to actively learn programs without the scaffolding support of teachers, which may make students have a higher extraneous cognitive load.
5.2. Different Impacts of Gender on Cognitive Load

This study showed no significant difference between males and females in cognitive load. In other words, teachers provide a curriculum that leads to a similar cognitive load for all students. Flipped teaching allows students to revisit material and discuss problems, which plays a key role in this course. This flipped CDIO engineering design enables all students to learn programs with convergent and divergent thinking. The flipped CDIO approach brings students to a sustainable learning solution. Students are able to revisit courses and questions, which helps all students to easily learn the course material. However, another explanation could be that the course content and its materials are easy to learn.

5.3. The Flow Experience of Using the CDIO Engineering Design

These results are consistent with previous studies indicating that improve students’ flow experience in programming courses [22,24]. From the study results, it was found that flipped CDIO engineering design can enhance the students’ engagement and flow control. In other words, in a programming environment, CDIO engineering can improve students’ levels of concentration and perceived degrees of control. This study speculated the reasons why it affected students’ flow experiences. First, programming requires a high degree of concentration, cognitive demands, and flexible control thinking. Through the CDIO engineering design process, students will have more perception and concentration on programming. Secondly, the CDIO engineering design plays a key role in this study, which has an explicit engineering design process for students. It may imply that the CDIO engineering design may appropriate as a process to support students engage and concentrate to learn programming.

5.4. Different Impacts of Gender on Flow Experience

The results indicated that there was no significant impact on the flow experience for students. In particular, flipped CDIO engineering design is a useful method to reduce the gender gap. Taking advantage of flipped online learning may help all students learn to program. In terms of gender difference, the results are similar to previous studies indicating that female students’ flow experience score was higher than male students [65,66]. In this study, the results showed that females scored higher than males in terms of enjoyment and engagement. In general, females generally have more emotional advantages and are more willing to learn programming. In contrast, males are generally better at logic and have an advantage over females at flow control. It is important to pay attention to the flow experiences of different genders, which is crucial for teachers designing teaching materials and courses. Different forms of teaching materials and courses may reduce the gender gap in the flow experience. Teachers also need to consider different genders and design a curriculum to fit the learning strengths of all students so as to help them learn effectively.

6. Conclusions

The main contribution of this study is the flipped CDIO approach, which provides a sustainable learning solution for students. The study indicates that the flipped CDIO engineering design allows all students to learn programs with integrated thinking. Students have greater flow experience from the flipped CDIO teaching. That is to say, this pedagogy provides students to learn programs with convergent and divergent thinking to promote students’ flow experience and sustainable learning.

This study provides suggestions to relevant practitioners on the adoption of a flipped teaching model for sustainable learning. First, programming courses have multiple skills and learning processes, which leads to the need for teachers to carefully design the curriculum. Our research suggests that practitioners can build constructivist learning environments (e.g., scaffolding, reflective activities, etc.) [67]. Teachers need to appropriately design their teaching and learning materials (e.g., videos) and design a student-centered curriculum. The use of different teaching tools or pedagogies can bring more learning bene-
fits to students in order to make them become self-regulated learners. Second, instructional guidance from teachers and live programming in the classroom are crucial to students' sustainable learning. Students can immediately correct problems with the teacher’s guidance, which allows them to learn programming skills and reflect on their learning. Finally, the study suggests that teachers use a modular approach to develop materials or courses. This has many pedagogical benefits for teachers (e.g., reorganizing teaching materials, reducing development time) and is beneficial for the sustainable education development of students.

7. Limitations and Future Work

The findings of this research are valuable for programming research. First, this study used quantitative methods to collect data, and the sample size in the study curriculum was smaller than expected (n = 40). As the sample size of this study was too small, it should not be generalized to other studies. In this study, the sample was composed of college students from a teacher college in a certain university, and a survey conducted in different universities may not show the same results. Therefore, it is suggested that teachers should design the curriculum in a student-centered way. It is important to make the course easy to learn for all students, regardless of their learning background. In addition, flipped or online courses need to provide response questions for students to suggest and give feedback on the course. These suggestions will allow teachers to revise their instructional design and curriculum. Secondly, this study uses the C# programming language as the programming curriculum. Due to the type of programming language, it may show limited effects (e.g., cognitive load, flow experience) of the CDIO engineering design. Third, in this study, flow experience was measured by a questionnaire, which also makes it difficult to generalize the results of the flow experience. Additional measurement tools are needed to measure students’ programming status. Future work should investigate the game-based learning or other teaching methods impact on the different programming language. Future research can consider different teaching methods (e.g., CDIO model, inquiry-based, game-based, etc.) under flipped teaching, which can provide a clearer understanding of the impact of these teaching methods on students.

Author Contributions: Conceptualization, H.-X.Z. and P.-S.C.; methodology, P.-S.C.; software, H.-X.Z.; validation, H.-X.Z.; formal analysis, P.-S.C.; investigation, C.-F.L.; resources, P.-S.C.; data curation, H.-X.Z.; writing—original draft preparation, H.-X.Z.; writing—review and editing, H.-X.Z.; visualization, C.-F.L.; supervision, H.-X.Z.; project administration, P.-S.C.; funding acquisition, P.-S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, under Grant no. MOST108-2511-H-415-003-MY3.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No datasets were used in the present research.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Bergmann, J.; Sams, A. Flip Your Classroom: Reach Every Student in Every Class Every Day; International Society for Technology in Education: Washington, DC, USA, 2012.
2. Sahin, A.; Cavlazoglu, B.; Zeytuncu, Y.E. Flipping a college calculus course: A case study. J. Educ. Technol. Soc. 2015, 18, 142–152.
3. Stöhr, C.; Demaziere, C.; Adawi, T. The polarizing effect of the online flipped classroom. Comput. Educ. 2020, 147, 103789. [CrossRef]
4. Akçayır, G.; Akçayır, M. The flipped classroom: A review of its advantages and challenges. Comput. Educ. 2018, 126, 334–345. [CrossRef]
5. Buil-Fabregá, M.; Martínez Casanovas, M.; Ruiz-Munzón, N. Flipped classroom as an active learning methodology in sustainable development curricula. Sustainability 2019, 11, 4577. [CrossRef]
6. Parra-González, M.E.; López Belmonte, J.; Segura-Robles, A.; Fuentes Cabrera, A. Active and emerging methodologies for ubiquitous education: Potentials of flipped learning andgamification. *Sustainability* 2020, 12, 602. [CrossRef]

7. Gómez-Garcia, G.; Hinojo-Lucena, F.J.; Caceres-Reche, M.P.; Ramos Navas-Parejo, M. The Contribution of the Flipped Classroom Method to the Development of Information Literacy: A Systematic Review. *Sustainability* 2020, 12, 7273. [CrossRef]

8. Pozo Sánchez, S.; López Belmonte, J.; Moreno Guerrero, A.J.; López Núñez, J.A. Impact of educational stage in the application of flipped learning: A contrasting analysis with traditional teaching. *Sustainability* 2019, 11, 5968. [CrossRef]

9. Çakıroğlu, U.; Öztürk, M. Flipped classroom with problem based activities: Exploring self-regulated learning in a programming language course. *J. Educ. Technol. Soc.* 2017, 20, 337–349.

10. Piteira, M.; Costa, C.J.; Aparicio, M. Computer programming learning: How to apply gamification on online courses? *J. Inf. Syst. Eng. Manag.* 2018, 3, 1–9. [CrossRef]

11. Martins, V.F.; de Almeida Souza Concilio, I.; de Paiva Guimarães, M. Problem based learning associated to the development of games for programming teaching. *Comput. Appl. Eng. Educ.* 2018, 26, 1577–1589. [CrossRef]

12. Topalli, D.; Cagiltay, N.E. Improving programming skills in engineering education through problem-based game projects with Scratch. *Comput. Educ.* 2018, 120, 64–74. [CrossRef]

13. Crawley, E.; Malmqvist, J.; Ostlund, S.; Brodeur, D.; Edstrom, K. Rethinking engineering education. In *The CDIO Approach*; Springer: New York, NY, USA, 2007.

14. Edström, K.; Kolmos, A. PBL and CDIO: Complementary models for engineering education development. *Eur. J. Eng. Educ.* 2014, 39, 539–555. [CrossRef]

15. Anmarkrud, Ø.; Andresen, A.; Bråten, I. Cognitive Load and Working Memory in Multimedia Learning: Conceptual and Measurement Issues. *Educ. Psychol.* 2019, 54, 61–83. [CrossRef]

16. Mayer, R.E.; Lee, H.; Peebles, A. Multimedia learning in a second language: A cognitive load perspective. *Appl. Cogn. Psychol.* 2014, 28, 653–660. [CrossRef]

17. Chen, C.-M.; Wu, C.-H. Effects of different video lecture types on sustained attention, emotion, cognitive load, and learning performance. *Comput. Educ.* 2015, 80, 108–121. [CrossRef]

18. Hong, J.; Pi, Z.; Yang, J. Learning declarative and procedural knowledge via video lectures: Cognitive load and learning effectiveness. *Innov. Educ. Teach. Int.* 2018, 55, 74–81. [CrossRef]

19. Pille, E.M. Flow experiences in information technology use. *Int. J. Hum.-Comput. Stud.* 2004, 61, 347–357. [CrossRef]

20. Chen, H.; Wigand, R.T.; Nilan, M.S. Optimal experience of web activities. *Comput. Hum. Behav.* 1999, 15, 585–608. [CrossRef]

21. Basawapatna, A.R.; Repenning, A.; Koh, K.H.; Nickerson, H. The zones of proximal flow: Guiding students through a space of computational thinking skills and challenges. In Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, San Diego, CA, USA, 12–14 August 2013; pp. 67–74.

22. Wang, L.C.; Chen, M.P. The effects of game strategy and preference-matching on flow experience and programming performance in game-based learning. *Innov. Educ. Teach. Int.* 2010, 47, 39–52. [CrossRef]

23. Bressler, D.M.; Bodzin, A.M. A mixed methods assessment of students’ flow experiences during a mobile augmented reality reality science game. *J. Comput. Assist. Learn.* 2013, 29, 505–517.

24. Liu, T.-Y. Using educational games and simulation software in a computer science course: Learning achievements and student flow experiences. *Interact. Learn. Environ.* 2016, 24, 724–744. [CrossRef]

25. Killi, K. Digital game-based learning: Towards an experiential gaming model. *Internet High. Educ.* 2005, 8, 13–24. [CrossRef]

26. Sullivan, A.; Bers, M.U. Girls, boys, and bots: Gender differences in young children’s performance on robotics and programming tasks. *J. Inf. Technol. Educ. Innov. Pract.* 2010, 15, 145–165. [CrossRef]

27. Rubio, M.A.; Romero-Zaliz, R.; Mañoso, C.; Angel, P. Closing the gender gap in an introductory programming course. *Comput. Educ.* 2015, 82, 409–420. [CrossRef]

28. Barr, D.; Harrison, J.; Conery, L. Computational thinking: A digital age skill for everyone. *Learn. Lead. Technol.* 2011, 38, 20–23.

29. Abeyesekera, L.; Dawson, P. Motivation and cognitive load in the flipped classroom: Definition, rationale and a call for research. *High. Educ. Res. Dev.* 2014, 34, 1–14. [CrossRef]

30. Roach, T. Student perceptions toward flipped learning: New methods to increase interaction and active learning in economics. *Int. Rev. Econ. Educ.* 2014, 17, 74–84. [CrossRef]

31. Cheng, L.; Ritzhaupt, A.D.; Antonenko, P. Effects of the flipped classroom instructional strategy on students’ learning outcomes: A meta-analysis. *Comput. Educ. Res. Dev.* 2019, 67, 793–824. [CrossRef]

32. O’Flaherty, J.; Phillips, C. The use of flipped classrooms in higher education: A scoping review. *Internet High. Educ.* 2015, 25, 85–95. [CrossRef]

33. Frydenberg, M. Flipping Excel. *Inf. Syst. Educ. J.* 2013, 11, 63–73.

34. He, W.; Holton, A.J.; Farkas, G. Impact of partially flipped instruction on immediate and subsequent course performance in a large undergraduate chemistry course. *Comput. Educ.* 2018, 125, 120–131. [CrossRef]

35. Sever, I.; Öncü, B.; Ersoy, A. Using Flipped Learning to Improve Scientific Research Skills of Teacher Candidates. *Univers. J. Educ. Res.* 2019, 7, 521–535. [CrossRef]

36. Cilli-Turner, E. Measuring learning outcomes and attitudes in a flipped introductory statistics course. *Primus* 2015, 25, 833–846. [CrossRef]

37. Csikszentmihalyi, M. *Beyond Boredom and Anxiety*; Jossey-Bass: San Francisco, CA, USA, 1975.
38. Chen, H.; Wigand, R.T.; Nilan, M. Exploring Web users’ optimal flow experiences. *Inf. Technol. People* **2000**, *13*, 263–281. [CrossRef]
39. Csikszentmihalyi, M.; Csikszentmihalyi, M. *Flow: The Psychology of Optimal Experience*; Harper & Row: New York, NY, USA, 1990; Volume 1990.
40. Fullagar, C.J.; Knight, P.A.; Sovern, H.S. Challenge/skill balance, flow, and performance anxiety. *Appl. Psychol.* **2013**, *62*, 236–259. [CrossRef]
41. Pearce, J.M.; Ainley, M.; Howard, S. The ebb and flow of online learning. *Comput. Hum. Behav.* **2005**, *21*, 745–771. [CrossRef]
42. Hwang, G.-J.; Chang, C.-Y. Facilitating decision-making performances in nursing treatments: A contextual digital game-based flipped learning approach. *Interact. Learn. Environ.* **2020**, 1–16. [CrossRef]
43. Wang, P.-Y.; Chiu, M.-C.; Lee, Y.-T. Effects of video lecture presentation style and questioning strategy on learner flow experience. *Innov. Educ. Teach. Int.* **2020**, 1–11. [CrossRef]
44. Wang, C.-C.; Hsu, M.-C. An exploratory study using inexpensive electroencephalography (EEG) to understand flow experience in computer-based instruction. *Inf. Manag.* **2014**, *51*, 912–923. [CrossRef]
45. Kiili, K.; De Freitas, S.; Arnab, S.; Lainema, T. The design principles for flow experience in educational games. *Procedia Comput. Sci.* **2012**, *15*, 78–91. [CrossRef]
46. Liao, L.F. A flow theory perspective on learner motivation and behavior in distance education. *Distance Educ.* **2006**, *27*, 45–62. [CrossRef]
47. Sweller, J.; Van Merrienboer, J.J.; Paas, F.G. Cognitive architecture and instructional design. *Educ. Psychol. Rev.* **1998**, *10*, 251–296. [CrossRef]
48. Sweller, J.; van Merrienboer, J.J.G.; Paas, F. Cognitive Architecture and Instructional Design: 20 Years Later. *Educ. Psychol. Rev.* **2019**, *31*, 261–292. [CrossRef]
49. Korbach, A.; Brünken, R.; Park, B. Measurement of cognitive load in multimedia learning: A comparison of different objective measures. *Instr. Sci.* **2017**, *45*, 515–536. [CrossRef]
50. Sweller, J. Cognitive load theory. In *Psychology of Learning and Motivation*; Elsevier: Amsterdam, The Netherlands, 2011; pp. 37–76.
51. Mayer, R.E. Multimedia learning. In *Psychology of Learning and Motivation*; Elsevier: Amsterdam, The Netherlands, 2002; pp. 85–139.
52. Homer, B.D.; Plass, J.L.; Blake, L. The effects of video on cognitive load and social presence in multimedia-learning. *Comput. Hum. Behav.* **2004**, *20*, 786–797. [CrossRef]
53. Mutlu-Bayraktar, D.; Cosgun, V.; Altan, T. Cognitive load in multimedia learning environments: A systematic review. *Comput. Educ.* **2019**, *141*, 103618. [CrossRef]
54. Wang, Y.; Gao, S.; Liu, Y.; Fu, Y. Design and implementation of project-oriented CDIO approach of instrumental analysis experiment course at Northeast Agricultural University. *Educ. Chem. Eng.* **2021**, *34*, 47–56. [CrossRef]
55. Taajamaa, V.; Eskandari, M.; Karanian, B.; Airola, A.; Pahikkala, T.; Salakoski, T. O-CDIO: Emphasizing Design Thinking in CDIO engineering cycle. *Int. J. Eng. Educ.* **2016**, *32*, 1530–1539.
56. Song, D.; Tavares, A.; Pinto, S.; Xu, H. Setting engineering students up for success in the 21st century: Integrating gamification and crowdsourcing into a CDIO-based web design course. *Eurasia J. Math. Sci. Technol. Educ.* **2017**, *13*, 3565–3585.
57. Schedin, S.; Hassan Osama, A.B. Work integrated learning model in relation to CDIO standards. *J. Appl. Res. High. Educ.* **2016**, *8*, 278–286. [CrossRef]
58. Kohn Rådberg, K.; Lundqvist, U.; Malmqvist, J.; Hagvall Svensson, O. From CDIO to challenge-based learning experiences—Expanding student learning as well as societal impact? *Appl. Psychol. Educ.* **2020**, *45*, 22–37. [CrossRef]
59. Leppink, J.; Paas, F.; Van Gog, T.; van Der Vleuten, C.P.; van Merrienboer, J.J. Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learn. Instr.* **2014**, *30*, 32–42. [CrossRef]
60. Garcia-Perales, R.; Palomares-Ruiz, A. Education in Programming and Mathematical Learning: Functionality of a Programming Language in Educational Processes. *Sustainability* **2020**, *12*, 129. [CrossRef]
61. Kalyuga, S.; Singh, A.-M. Rethinking the Boundaries of Cognitive Load Theory in Complex Learning. *Educ. Psychol. Rev.* **2017**, *29*, 831–852. [CrossRef]
62. Paas, F.; van Gog, T. Optimising worked example instruction: Different ways to increase germane cognitive load. *Learn. Instr.* **2006**, *16*, 87–91. [CrossRef]
63. Mason, R.; Seton, C.; Cooper, G. Applying cognitive load theory to the redesign of a conventional database systems course. *Comput. Sci. Educ.* **2016**, *26*, 68–87. [CrossRef]
64. Hmelo-Silver, C.E.; Duncan, R.G.; Chinn, C.A. scaffolding and achievement in problem-based and inquiry learning: A response to kirschner, sweller, and clark. *Educ. Psychol.* **2007**, *42*, 99–107. [CrossRef]
65. Hsieh, Y.-H.; Lin, Y.-C.; Hou, H.-T. Exploring the role of flow experience, learning performance and potential behavior clusters in elementary students’ game-based learning. *Interact. Learn. Environ.* **2016**, *24*, 178–193. [CrossRef]
66. Yang, J.C.; Quadir, B. Individual differences in an English learning achievement system: Gaming flow experience, gender differences and learning motivation. *Technol. Pedagog. Educ.* **2018**, *27*, 351–366. [CrossRef]
67. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]