# A Comprehensive Experiment Approach to Enhancing Computer Engineering Ability

1st Liang Zhang
*School of Computer Science and Engineering*
*Beihang University*
Beijing, China
liang.z@buaa.edu.cn

2nd Jianwei Niu
*School of Computer Science and Engineering*
*Beihang University*
Beijing, China
niujianwei@buaa.edu.cn

***Abstract*—This Research to Practice Full Paper presented a comprehensive experiment approach to enhancing computer engineering ability. This approach integrated Swift programming language, iOS development, UML, software testing, MVC, Cocoa Touch Framework and Design Patterns into a comprehensive experiment, through which students can master the engineering methods to solve complex application problems.**

**In college, traditional computer programming courses focus on the grammar and classical algorithm of programming language. Usually the amount of code is far lower than that of industrial products. Such programming courses can't effectively improve students' ability to solve complex engineering problems. They also can't meet the requirements of industrial development. Students are not satisfied with the results of these courses. There is an intense need for the studies of enhancing student's computer engineering ability.**

**Taking Swift Language Programming course as an example, this paper presented a comprehensive experiment approach to enhancing students' computer engineering ability by developing classic industrial iOS Apps.**

**Flipped classroom pedagogy is conducive to free much time in class. Lecturers can fully communicate with students and help students complete challenging tasks. The comprehensive experiment consists of pre-class activities and in-class activities. Before class, the lecturer provides experiment materials online including theoretical handouts of Design Patterns, manuals of UML 2.0 specifications and Cocoa Touch reference manual, etc. Students learn the materials by themselves, practice and discuss online and complete the corresponding pre-class tests. In class, the lecturer analyzes in detail the problems students encounter after class and guides them to solve these problems. The lecturer also participates in each group discussion to ensure the smooth progress of students' project.**

**The implementation of comprehensive experiment is divided into four sub tasks. These tasks are app function analysis, App detailed design, programming implementation, and App release and launch. First, according to the requirements of the App, the function is analyzed in detail and defined with UML. Second, based on functional analysis, the App's system architecture, data structure, view combination, logic execution process and core algorithms are designed. The system is defined in detail with UML Class diagram. Third, according to the detailed design of the App, user interface is built by Xcode storyboard, and the model layer, view layer and control layer are implemented in Swift. Then unit test and system test are conducted on the App and bugs are repaired. Finally, App launch is completed including App internationalization, developer certificate applying, creating description file, setting product identification and deployment information, and submitting App online.**

**To assess the effect of this comprehensive experiment approach, three-year teaching data were analyzed using statistical methods. The results show that students' engineering ability (measured by code scale) and student satisfaction (measured by questionnaires) were significantly improved.**

**Our contribution is to propose a detailed comprehensive experiment approach to enhancing computer engineering ability. The analysis of teaching data show that it is helpful to improve students' computer engineering ability and course satisfaction.**

***Index Terms*—engineering ability, comprehensive experiment, iOS App, active learning, flipped classroom**

Jianwei Niu is the corresponding author (niujianwei@buaa.edu.cn).

## I. Introduction

In order to meet the needs of the rapid development of information industry, the cultivation of College Students' engineering ability is a crucial purpose in higher education. Educating the engineer of 2020 program [1] was initiated by the American Academy of engineering to train engineers to satisfy the demands of the new period. ABET's new criteria for engineering programs redefined its relationship with engineering programs by basing its new criteria on outcomes rather than inputs [2]. In the past two decades, people have made great efforts to improve engineering education and made progress such as student-centered methodologies [3]. Due to China's accession to the Washington Agreement, professional construction in line with international engineering education became an urgent task [4]. In 2018, Wu [5] proposed that the goal of curriculum construction in Chinese higher education was to reasonably improve the difficulty of academic challenges, increase the difficulty of courses, expand the depth of courses and effectively improve the quality of teaching. In 2019, the construction of first-class professions launched in China, which emphasized that the basic orientation of undergraduate engineering education was to cultivate students' ability to solve complex engineering problems [6], [7].

In order to respond to the call of the Chinese government and catch up with the advanced level of international engineering education, we tried to develop new teaching materials around enhancing engineering ability. The comprehensive experiment proposed in this paper is a typical case of these materials. Computer related disciplines have strong

engineering requirements in essence and solving problems in practical engineering is an important goal [8].

Wu and Sun et al. [9] conducted a study on practical teaching methods of software development courses and concluded that practical teaching oriented to the goal of ability training was an effective way to satisfy the demands of engineers for the rapid development of software industry.

Aziz and Islam [10] tried to comprehensively use various teaching methods in online engineering education, and analyzed their impact on students' learning experience.

Silla Jr. [11] presented a novel approach based on active learning pedagogical strategies and entrepreneurship programs to teach entrepreneurship to computer science and engineering students.

Singh [12] shared the learning ecosystem of a project-based safety-critical systems course, identifying course contents that support self-directed learning and how assignments develop students' knowledge and understanding toward meeting the demand of industries.

In computer engineering profession, problem-solving skills, strong communication skills, and teamwork are the desired traits [13], [14]. However, the new engineering graduates sometimes lacked these skills, and they found it difficult to apply the fundamental knowledge to practical problems [15].

Taking Swift Language Programming course as an example, this study presented a comprehensive experiment approach to enhancing students' computer engineering ability by developing classic industrial iOS Apps.

The remainder of this paper is structured as follows. Section II presents the pedagogy of the comprehensive experiment. Section III provides a detailed implementation of the comprehensive experiment which consists of four sub tasks. Section IV analyzes the effect of the comprehensive experiment from App complexity and student satisfaction. Finally, conclusions are given in Section V.

## II. Pedagogy

The existing engineering education research focuses on providing effective learning experience for students through technology [10]. These studies pointed out that flipped classroom-based methods were effective for engineering education in blended mode [16]. Flipping classroom pedagogy in higher education is conducive to improving students' self-study ability [17]. Flipped classroom also frees much time required for lecturers to convey more information and provide instant feedback to students [18].

The comprehensive experiment adopted flipped classroom pedagogy. This experiment was a new developed teaching material of Swift Language Programming course. It integrated Swift programming language, iOS development, UML, software testing, MVC, Cocoa Touch Framework and Design Patterns. The teaching process of experiment is divided into four segments: pre-class activities, in-class lecture, in-class discussion and post-class submission.

*Pre-class Activities:* Two weeks before the experiment, provide relevant learning materials online, including Design Patterns, UML specification and Cocoa Touch manual, etc. Two weeks before the experiment, relevant learning materials are provided online, including design patterns, UML specifications, cocoa touch manuals, etc. Students are required to complete the study of these materials before class and complete the pre-class test online.

*In-Class Lecture:* Frist, the lecturer reviews the relevant theories and technologies in the experiment. Second, the lecturer focuses on the difficulties and key steps in project development. Finally, the lecturer explains in detail the common problems in pre-class test.

*In-Class Discussion:* Students work in teams on the project, including requirements analysis, system design, architecture design, detailed design and implementation of MVC, coding, unit test and function test and App launch. The lecturer and teaching assistant participate in each team's discussion on requirements analysis and system design to ensure that their projects are carried out in the right way. The rest of the work is done by the students independently. The lecturer and teaching assistant randomly check the progress of each team and correct the problems in development in time.

*Post-Class Submission:* Students are required to continue to complete the project development and relevant documents, summarize the engineering experience, and submit the results in the form of experimental report and source codes after class. The teaching assistant reviews the experimental reports submitted by each team online. The lecturer checks and accepts students' App and corresponding source codes. Student's submission are evaluated in terms of quality, difficulty, workload, innovation, usefulness of their projects through anonymous peer grading. Finally, the completion teams of excellent projects are required to demonstrate their App in class and share their engineering development experiences.

## III. Implementation

This approach integrated Swift Programming Language, iOS Development, UML, Software Testing, MVC, Cocoa Touch Framework and Design Patterns into a comprehensive experiment. By appropriately reducing the functions of App, repeated engineering development is greatly reduced. In addition, the system architecture was also redesigned to simplify the system, which was suitable for classroom teaching. In order to help students to carry out the project smoothly, theoretical and technical materials related to the experiment other than textbooks are provided to students. This experiment is mainly divided into four tasks: functional analysis, system design, implementation and App launch.

### A. Functional Analysis

*Instructional Objective:* Through the system function analysis of App "2048", learn to use UML modeling method (activity diagram) to define system function.

*Requirements:* Make a detailed system function design for App "2048" and draw a sketch of main interface. Then UML activity diagram was used to describe the main functions of system and the constraint relationships among functions.

*Experimental Process*

Step 1: Sketch App User Interface

In function analysis stage, students are required to draw a sketch of App's main interface first. Based on the main function of App "2048", students can design auxiliary functions such as leaderboard, scoreboard and brick appearance customization on the interface. These functions are fully designed and innovated by students. Based on the App design sketch, students are required to analyze the main functions of App according to game rules, and then describe the App functions through UML activity diagram.

The reference user interface of App "2048" is shown in Fig.1. It consists of two parts: score board and game board. After each movement of the scoreboard, the scores are accumulated and recorded. The game board is composed of 16 square grids of 4 * 4, and each grid contains a brick. At the beginning, all grids are empty. As the game progressed, bricks with number "2" or "4" are randomly generated in blank position. The App interface can recognize gestures, including up, down, left and right. Bricks are moved according to these gestures.

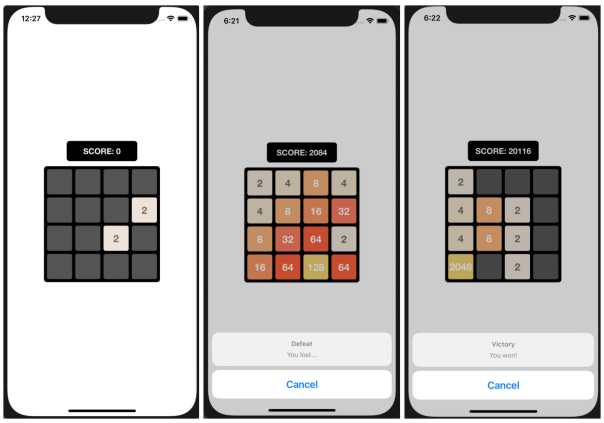


Fig. 1. User interface of App "2048".

Step 2: Summarize the rules of game

Students are required to summarize the rules of the game according to their game experience. The reference game rule of "2048" is: Players can choose one sliding direction at a time. Each time you slide, all the digital bricks will move in the sliding direction. At the same time, the system will randomly generate a new brick with the number "2" or "4" in the blank,. When bricks with the same number collide, they will be merged and the numbers will be accumulated. Players should try to find the digital brick of "2048" in the range of 16 game boards. When the digital brick "2048" appears, players win the game. If the range of the game board has been filled, the digital bricks have not appeared "2048", and there are no adjacent bricks that can be merged, the game fails.

Step 3: System modeling based on activity diagram

According to the system design sketch and game rules, UML activity diagram is used to model the system functions. The result is the basis for further detailed design. The UML activity diagram at least includes: start the game, exit the game, push bricks, merge bricks, scoreboard update, randomly generate bricks and other activities, as well as the judgment of conditions such as game goal achievement and game failure. The reference activity diagram of App 2048 is shown in Fig.2.

## B. System Design

*Instructional Objective:* Master the method of system design based on MVC pattern through the system architecture design of app "2048". Through the detailed design of the model layer, view layer and controller layer of App, master the method of detailed design of system based on UML class diagram. The result will be used as the framework for the implementation.

*Requirements:* Based on MVC pattern, complete the system architecture design of App "2048" and the detailed design of each layer in MVC. In the model layer, the relevant objects in the App are abstracted into specific model classes which are defined by UML class diagrams. In the view layer, it is necessary to decompose the designed App interface sketch into several sub view components, and model the sub view classes using UML class diagram. In the controller layer, its logic is defined according to the design of model layer and view layer, and the view controller classes are also defined by UML class diagram.

*Experimental Steps*

Step 1: System Architecture Design

In the architecture design stage, the system architecture is designed based on MVC pattern. The model layer, view layer and controller layer are designed in detail, and modeled using UML class diagram. The system architecture of App "2048" is shown in Fig.3. The model layer consists of data objects and game models. The model layer encapsulates the definition, calculation and various operations of data through model objects. It is composed of UI layers, UI objects, UI views. UIWindow coordinates the presenting of multiple views in the app on the screen. Views and UI objects provide visual app content, drew visual view components within the specified range, and responded to external interaction events. The control layer consists of UIApplication, Application Delegate and View Controller. UIApplication manages the event loop queue and the behavior of the App, and reports the key state changes and special events of the App to its agent. Application Delegate and UIApplication object handle App initialization, state transition management, and App event processing together. View controller presents the App view.

Step 2: Model Layer Design

Use UML class diagram to design the model layer in detail. The class diagram includes the main classes in the model layer and the relationships among them. The incomplete class diagram of model layer is given in Fig.4. GameModel is the core class. Its related attributes define the appearance and scoring data of the game board. Its methods define the main operations in the App, including: game board initialization, resetting, inserting bricks, moving bricks, merging bricks, and judgment of the current game state (win or lose?). Auxiliary classes

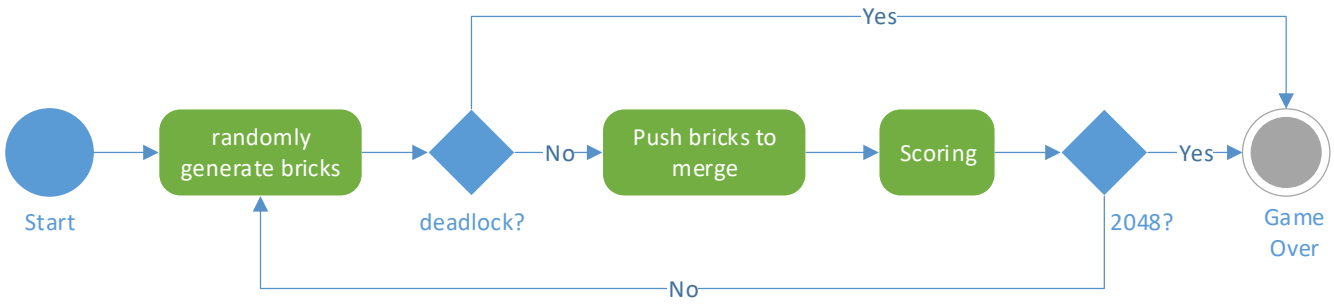


Fig. 2. UML activity diagram of App "2048".

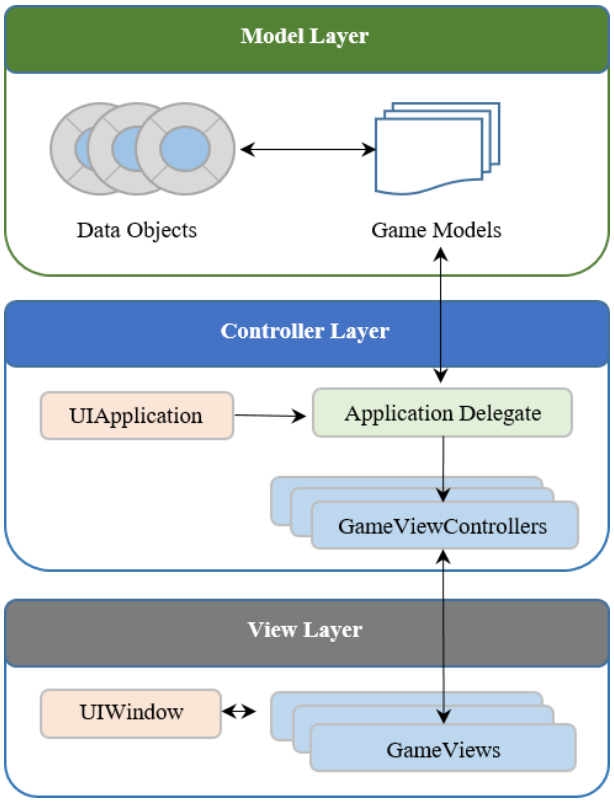


Fig. 3. System Architecture of App "2048".

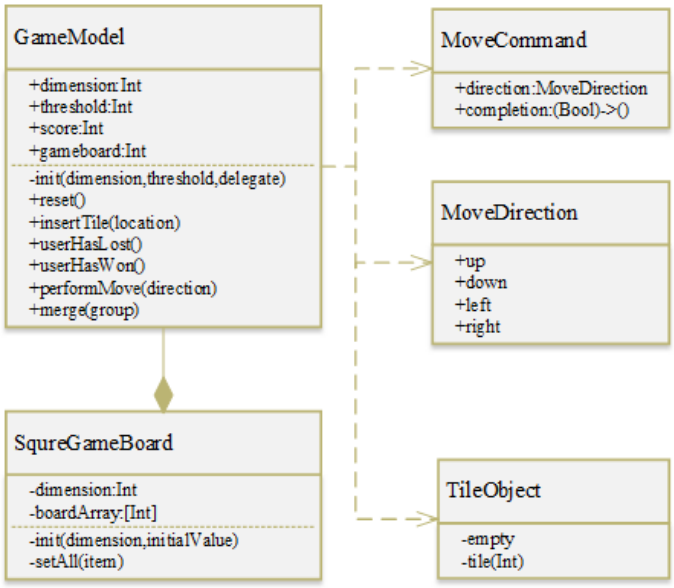


Fig. 4. Incomplete class diagram of model layer.

associated with GameModel are also defined in the class diagram, including MoveCommand (moving brick instruction), MoveDirection (moving direction) and TileObject (brick). There is a combination relationship between SqureGameBoard and GameModel. SquareGameBoard defined the appearance of the game board and the state of all grids. It is a part of GameModel. Based on the incomplete class diagram, students are required to complete the definition of attributes and methods in the classes according to their own design.

Step 3: View Layer Design

Use UML class diagram to design the view layer in detail. The class diagram includes the main classes in the view layer and the relationships among them. The incomplete class diagram of view layer is given in Fig.5. TileView is a brick view class, which describes the brick view. It includes the numerical properties, numerical labels and initialization methods of bricks. ScoreView is a scoreboard view class, which displays the game scoreboard. ControlView is a control view class, which defines the size of default frame of view. ScoreViewProtocol is a scoreboard view protocol, which specifies the function interface for changing the score of scoreboard. GameboardView is a game board view class, which defines the properties related to the game board, including: size, brick width, brick frame size, corner radius, brick view set and other variables. The class diagram of view layer defines the basic view elements of App "2048". On this basis, students are required to appropriately expand and modify the class diagram according to their own design.

Step 4: Controller Layer Design

Use UML class diagram to design the controller layer in detail. The class diagram includes the main classes in the controller layer and the relationships among them. The incomplete class diagram of controller layer is given in Fig.6. UIViewController is the default view control class of the

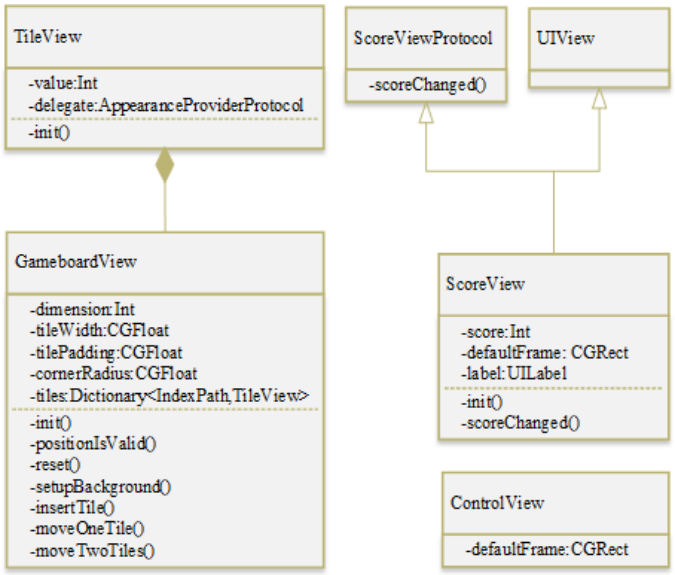


Fig. 5. Incomplete class diagram of view layer.

system and the base class of NumberTileGameViewController. GameModelProtocol is a protocol defined by the model layer. GameboardView and ScoreView are the view classes defined in the view layer, and GameModel is the game model class defined in the model layer. NumberTileGameViewController is the core class of the control layer, which defines all view controller logic in the App main view.

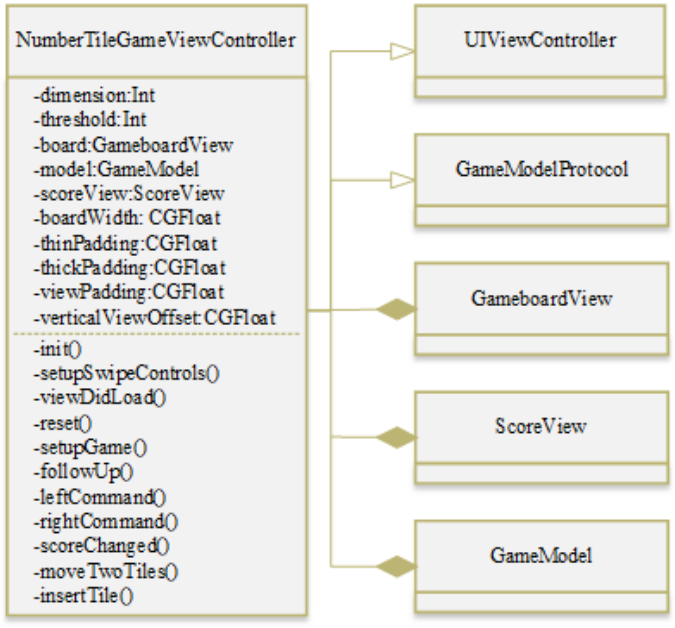


Fig. 6. Incomplete class diagram of controller layer.

### C. System Development

*Instructional Objective:* According to the system design of App "2048", complete the definition of classes in the model layer, view layer and control layer in Swift.

*Requirements:* According to the detailed definition of class diagram of each layer, all classes are coded using Swift in Xcode. All the functions defined above are implemented. Complete App testing and fix all found bugs.

*Experimental Steps*

Step 1: Model Layer Programming

The core class in the model layer is GameModel. We provide students with the reference code framework of it, including its attribute part and method part. We only give the interface of the methods, and don't provide the complete implementation of methods. For difficult methods (such as performMove(), condense() and collapse()), reference codes are provided to students.

Step 2: View Layer Programming

The core class in the model layer is GameModelView. We provide students with the reference code framework of it, including its attribute part and method part. We only give the interface of methods, and don't provide the complete implementation of methods. For difficult methods (such as insertTile(), moveOneTile() and moveTwoTiles()), reference codes are provided to students.

Step 3: Controller Layer Programming

The core class in the model layer is NumberTileGameViewController. We provide students with the reference code framework of it, including its attribute part and method part. We only give the interface of methods, and don't provide the complete implementation of methods. For difficult methods (such as setupGame() and followUp()), reference codes are provided to students.

Step 4: Test and Debug

Unit test is required while coding. After coding, a comprehensive functional test is done in the simulator according to the functional design of App. Errors found during testing are required to be located and repaired through Xcode debugging tool.

### D. App Launch

*Instructional Objective:* Learn the whole process of App launch in App Store, including the closing work of project and App release process.

*Requirements:* Complete the closing work, including App internationalization, making App icons and startup screen to improve user experience. Complete the relevant processes of App launch, including applying for developer certificate, creating description file, setting product identification and deployment information, and submitting App.

*Experimental Steps*

Step 1: Internationalization

It is a process of translating one language in App into multiple languages, so that the App can support the display of different languages.

Step 2: Icon making

Due to the large range of Apple mobile devices, the pictures used in the App are required to be available to all types of Apple devices.

Step 3: Startup Screen making

Apple official requires all Apps to have a startup screen. Due to the diversity of screen sizes of Apple devices, the size requirements of startup screen are also complex. Using storyboard to create the startup screen is a convenient solution.

Step 4: Developer Certificate applying

To launch an App to App Store, students need a developer certificate. There are two kinds of Developer Certificates: individual developer and enterprise developer. Students can apply for individual version or use the enterprise version provided by our course.

Step 5: Provisioning Profile creating

Provisioning profile is used to compile Apps on devices. There are two kinds of description files: development description files and release description files. In the developer Management Center, students can complete the work.

Step 6: Product information setting

Students are required to specify the corresponding developer ID in the signature information of project and import the description files. In addition, it is also necessary to specify the identification of this product release, including the name displayed on the App, package identifier and version number, etc. Finally, the deployment information should be set according to the actual deployment target device. After setting, recompile and submit the App to App Store.

Step 7: Submission

Students use their own Apple ID to submit Apps online and wait for review results. If the review fails, it needs to be modified according to the feedback. After approval, the App can be downloaded from App Store.

## IV. Evaluation

From 2019 to 2021, the number of students was 89, 41 and 42 respectively. In 2019, students mainly practiced Swift language programming by designing algorithms with single functions. The average code lines was about 30-50 lines. From 2020, the course began to introduce industrial iOS apps with a code scale of more than 1000 lines. At the end of this course, each student was required to implement a free design App. The lecturer did not provide any materials and source codes. Apps were implemented by students completely according to their own ideas. The number of core code lines is used to measure the engineering complexity of App. In addition, students' satisfaction is evaluated through course questionnaire.

### A. Complexity Analysis of App

The complexity of App is measured by the number of core code lines. By comparing the numbers, we can roughly evaluate whether students' engineering ability are improved. The numbers of students' App from 2019 to 2021 is shown in Fig.7.

We used the Kruskal–Wallis test which was based on complexity and was an overall test of any differences across 2019, 2020, 2021. The results are: test statistic = 58.4902, p(2-sided) = 0.000, N = 172, meaning that there are differences between years. There are significant pairwise comparisons ($p<0.001$) for two comparisons, namely, between 2019 and 2020, and 2019 and 2021. The result shows that there is a significant difference between 2019 and 2020, and 2019 and 2021. Due to the fact that this comprehensive experiment wasn't introduced into the course in 2019, students still had great difficulties in designing large-scale engineering applications. In 2020, the significant increase in the number of core code lines might be explained as that after taking the comprehensive experiment as the teaching content in 2020, students gradually mastered the engineering design methods of complex Apps and solved the main obstacles in their design. In 2021, the median number of codes increased slightly. In fact, the App designed by students significantly improved in code quality and functional complexity, but it was not reflected in the data. The reasons for the difference between 2020 and 2021 may come from two aspects. First, after the attempt in 2020, lecturers and students in 2021 were more familiar with the flipped classroom pedagogy and the project-based comprehensive experiments. Second, learning materials were continuously improved and increased in 2021, which provided great convenience for students' self-study. In conclusion, after the implementation of the comprehensive experiment approach, the complexity of students' free design App increased significantly, which showed that students' engineering ability was effectively promoted.

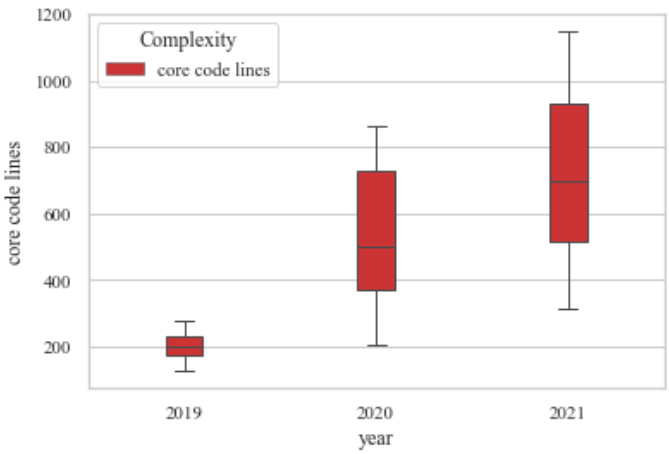


Fig. 7. The number of core code lines from 2019 to 2021.

### B. Student Satisfaction Analysis

After each course, the course questionnaires were distributed to all students. Through the analysis of the questionnaire, we could known the students' feelings about the course. The typical question was "What is your overall impression of the course? Rated from 1 (very poor) to 5 (excellent)". The descriptive statistics for the question are shown in Fig.8.

The result means that the comprehensive experiment approach can effectively improve students' course satisfaction. Compared with the data in 2019, the students' evaluation in 2020 is lower, mainly because the comprehensive experiment was introduced into the curriculum for the first time in 2020 and the flipped classroom pedagogy was adopted. Students

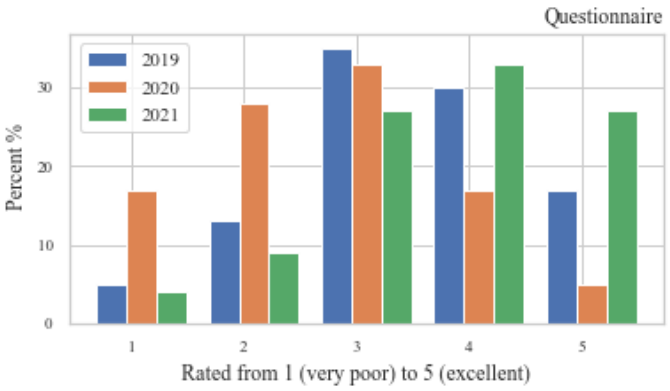


Fig. 8. What is your overall impression of the course? Rated from 1 (very poor) to 5 (excellent).

did not adapt to the new teaching contents and pedagogy. However, by comparing the data of 2019 and 2021, it is clearly found that the satisfaction of students was greatly improved, and the number of students with low evaluation scores of 1 and 2 decreased significantly. After adapting to the new comprehensive experimental approach, students learned more knowledge and promoted their engineering ability, so their course satisfaction was naturally improved.

In the course questionnaire, there are four questions for evaluating the comprehensive experiment, as shown in Fig. 9.

Q1: Compared with the traditional lecture, do you agree Flipped Classroom pedagogy adopted in the comprehensive experiment is more conducive to your learning? Rated from 1 (Completely disagree) to 5 (Strongly agree).

Q2: Do you agree that the learning difficulty of the comprehensive experiment in this course is appropriate? Rated from 1 (Disappointing) to 5 (Very useful).

Q3: By implementing complex iOS App, your mastery and understanding of Swift language have been improved? Rated from 1 (Completely disagree) to 5 (Strongly agree).

Q4: Through the practice of comprehensive experiments, do you think your ability to implement complex iOS App has been greatly improved? Rated from 1 (Completely disagree) to 5 (Strongly agree).

The descriptive statistics for above questions are shown in Fig. 9, which shows students' impression of the comprehensive experiment. According to the statistical results, there are still a small number of students who fail to adapt to the flipped classroom teaching method. The majority of students think that the comprehensive experiment is very helpful for their programming learning.

## V. Conclusion

This paper proposed a comprehensive experiment approach including the corresponding pedagogy and detailed implementation process. In 2020, we began to teach this comprehensive experiment in Swift Programming course of Beihang University. Through the analysis of teaching data from 2019-2021, it was concluded that this comprehensive experiment approach

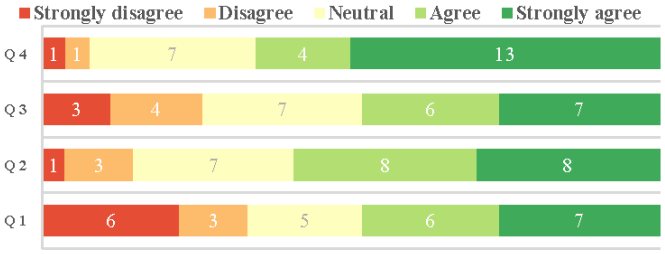


Fig. 9. 2021 Questionnaires. Rated from 1 (very poor) to 5 (excellent).

could significantly enhance students' computer engineering ability and course satisfaction. This approach might be used as a reference for comprehensive experiment design in programming course.

To evaluate engineering ability, the number of core code lines was used to measure the complexity of App, but it is not comprehensive. In future research, we will try to use a combination of peer grading and expert grading to comprehensively evaluate the project complexity. In addition, the questionnaire for the comprehensive experiment will be used to iteratively improve the experiment approach.

## Acknowledgment

This work was partially supported by the Beihang University education reform funding. We would like to thank the teaching assistants for their help in the implementation and improvement of the course. We would also like to thank the anonymous reviewers for their thoughtful feedbacks on this paper.

## References

[1] "Educating the engineer of 2020: Adapting engineering education to the new century," in IEEE Engineering Management Review, vol. 37, no. 1, pp. 11-11, First Quarter 2009, doi: 10.1109/EMR.2009.4804343.

[2] J. Lucena, G. Downey, B. Jesiek, and S. Elber, "Competencies Beyond Countries: The Re-Organization of Engineering Education in the United States, Europe, and Latin America," Journal of Engineering Education, vol. 97, no. 4, pp. 433–447, Oct. 2008, doi: 10.1002/j.2168-9830.2008.tb00991.x.

[3] M. Borrego, J. E. Froyd, and T. S. Hall, "Diffusion of Engineering Education Innovations: A Survey of Awareness and Adoption Rates in U.S. Engineering Departments," Journal of Engineering Education, vol. 99, no. 3, pp. 185–207, Jul. 2010, doi: 10.1002/j.2168-9830.2010.tb01056.x.

[4] Ministry of Education of the People's Republic of China. Notice of the Ministry of education on paying close attention to the implementation of the spirit of the National Conference on undergraduate education in higher education in the new period [EB/OL].

[5] Y. Wu. Developing Chinese gold curriculum [J]. China University Teaching, 2018, 2018(12):4-9.

[6] X.P. Gao. Process control and evaluation system of College Students' online training [J]. China University Teaching, 2020, 2020(8):37-42.

[7] Z.L. Jiang. Undergraduate engineering education: focusing on the cultivation of students' ability to solve complex engineering problems [J]. China University Teaching, 2016, 2016(11):27-30.

[8] Z.L. Jiang. Engineering professional certification guides the reform of Engineering Education in Colleges [J]. Industry and information education, 2014, 2014(1):1-5.

[9] J. Wu, Q. Sun, W.G. Rong et al. Research on practical teaching methods of software development courses driven by ability training objectives [J]. China University Teaching, 2018, 2018(10):37-43.

[10] A. Aziz and S. N. Islam, "Impact of Mixed Pedagogy on Engineering Education," IEEE Trans. Educ., vol. 65, no. 1, pp. 56–63, Feb. 2022.

[11] C. N. S. Jr. “Teaching Entrepreneurship for Computer Science and Engineering Students Using Active Learning Pedagogical Strategies,” in 2021 IEEE Frontiers in Education Conference (FIE), Lincoln, USA, Oct. 2021, pp.1-6

[12] P. Singh and L. K. Singh, “Engineering Education for Development of Safety-Critical Systems,” IEEE Trans. Educ., vol. 64, no. 4, pp. 398–405, Nov. 2021.

[13] J. E. Mills and D. F. Treagust, “Engineering education—Is problem-based or project-based learning the answer?” Aust. J. Eng. Educ., vol. 3, no. 2, pp. 2–16, Dec. 2003.

[14] National Academy of Engineering (NAE), The Engineer of 2020: Visions of Engineering in the New Century, Washington, DC, USA: Nat. Acad. Press, 2004.

[15] C. E. Vergara et al., “Aligning computing education with engineering workforce computational needs: New curricular directions to improve computational thinking in engineering graduates,” in Proc. 39th IEEE Front. Educ. Conf., San Antonio, TX, USA, 2009, pp. 1–6.

[16] R. M. Clark, A. Kaw, and M. Besterfield-Sacre, “Comparing the effectiveness of blended, semi-flipped, and flipped formats in an engineering numerical methods course,” Adv. Eng. Educ., vol. 5, no. 3, pp. 1–38, 2016.

[17] Zainuddin, Z., Attaran, M. Malaysian students’ perceptions of flipped classroom: A case study. Innovations in Education and Teaching International, 53(6), 660-670, 2016.

[18] H. Al-Samarraie, A. Shamsuddin, and A. I. Alzahrani, “A flipped classroom model in higher education: a review of the evidence across disciplines,” Education Tech Research Dev, vol. 68, no. 3, pp. 1017–1051, Jun. 2020, doi: 10.1007/s11423-019-09718-8.