Design and Implementation of Gamified Learning System for Mutation Testing

Nien-Lin Hsueh*, Zeng Hung Xuan, and Bilegjargal Daramsenge

***Abstract*—Mutation Testing is used to test case improvement mechanism for evaluating the effectiveness of test cases by generating a large number of mutants. In the past, some approaches are proposed to improve the performance of generating the mutants. In recent years, many studies have begun to explore the software engineering education of mutation testing, trying to make students understand its concept through gamification. In this paper, we apply gamification theory and build a gamified learning system for mutation testing, named code immunity boost (CIB), taking the story of vaccine development as a sense of mission. We invited students to learn mutation testing through the relationship between leukocyte (test case), vaccine (mutant) and human body (program). Students can play the role of a vaccine and stimulate the testing of test cases by writing mutants to improve the quality of the program. We adopted the benchmark programs commonly used in mutation testing research, and developed a code vaccine incubator (CVI) tool to generate a large number of mutants as the experimental cornerstone of this study. Final experiments show that our tool can help mutation testing education, as the performance and quality of the experimental group is better than that of the control group. The response from the questionnaire also shows students like learning by our gamified tool. We therefore recommend to promote such software testing education approach by integrating our tool with popular online programming tools such as Online Judge system.**

***Index Terms*—Mutation testing, mutant, gamified education**

I. Introduction

Software engineering has been an important engineering technology recently. Students in each software field need to go through a series of training, such as version management, requirements analysis, system design, software testing technology, architecture design, etc. But most of the attention was paid to programming education, while software testing and program quality are ignored. Although many students have good programming ability, they did not have enough skills to test software improve their quality. Therefore, software engineering education has also become an important research topic [1–4].

On the other hand, with the popularity of the Internet, APP, and games, traditional education methods are gradually difficult to stimulate students' interest in learning. Learning by gamification is favored by people [5–14]. The application of gamification theory has also become one of the important researches in software engineering education. In the paper [15], 156 papers were discussed on this topic, and it was found that gamification can help software engineering in the following ways: 1) improve students' participation in courses; 2) improve students' professional knowledge; 3) encourage students to adopt software engineering practices; 4) improve students' teamwork skills. In general, the gamification techniques used in such papers include Leaderboards, Points, Milestones, Levels, Paths and Progress Competition and Rewards. The applied software engineering knowledge fields include software development process, software design, requirements analysis, software validation, process improvement, software construction and software maintenance.

In software engineering gamification learning, most of the fields involve software design, software process and requirements analysis, and the number of software tests is relatively low [15]. However, the importance of software testing is unquestionable. In order to allow students to learn mutation testing, Rojas and Fraser developed a game of mutation testing combined with gamification learning, called Code Defenders [16], the game is played by two people, one person plays the role of the attacker to write mutants, the other plays the defender to write test cases, and the two sides can compete, so that players can improve their knowledge and writing ability of mutation testing in the process of competition.

Inspired by Code Defenders and the COVID-19 pandemic in the past two years, this study intends to design a gamified learning system for mutation testing based on the concept of the human body's story of boosting immunity.

In order to provide a learning environment for learning mutation testing, this study combines mutation testing with gamification education, and constructs a mutation test based on the story of vaccine development. We have developed a learning system, named code immunity boost (CIB), and developed a mutant generate tool, named code vaccine incubator (CVI).

The work is organized as below. The second section introduces the techniques used in this study and related research literature, including the mutant testing and the related gamification work. The third section is our game design including overall system architecture and the gamification methods used in the research. The fourth section is the results of a pilot experiment and questionnaire. The fifth section is the conclusion, which describes the contribution and future development direction of this research.

II. Background and Related Work

Mutation testing was originally proposed by DeMillo *et al.* as a test method for evaluating the effectiveness of test cases for unit testing [17]. DeMillo *et al.* believed that the original program could be modified by a small amount to form a

Manuscript received December 20, 2022; revised January 28, 2023; accepted February 12, 2023.

The authors are with the Department of Information Engineering and Computer Science, Feng Chia University, Taiwan.

*Correspondence: nlhsueh@fcu.edu.tw (N.L.H.)

program that are extremely similar but different are called mutants.

The process of mutation testing is described as follows: First, generate a large number of different mutants from the original program; second, test the original program and the mutants by the test cases; and then compare their output:

- if the outputs are the same. It means that the test cases cannot detect the difference between the mutant and the original program. In this case, the mutant is regarded as *alive*, waiting for testers to add more effective test cases to remove (kill) it from the mutant set.
- if the outputs are different. It implies the test suite is effective enough to detect the difference between the original program and its mutants. In this case the mutant will be *killed* and removed from the mutant set.

Adding test cases to kill mutants is an iterative process in mutation testing until all mutants except equivalent mutant are killed. An equivalent mutant is a mutant that do not change the behavior of the original program and would not be killed by any test case. The mutation score (*ms*) is an indicator used to calculate the effectiveness of the test case:

$$ms = \frac{killed}{all - equiv}$$

where *killed*, *all* and *equiv* represent the number of killed mutants, all mutants and equivalent mutants. The range of the mutation score is between 0.0 ∼ 1.0. Mutation score is the percentage of mutants that were killed by the test suite. The higher the mutation score, the more effective the test suite is at detecting errors.

A. *Mutation Testing with Gamification*

Code Defenders is a game developed by Jos and Gordon based on mutation testing [18, 19]. The research team believes that there are still many computer science students do not understand mutation testing. They combine mutation testing with gamification, hoping that students can learn mutation testing through competition and entertainment. In the game, player can choose to play the role of attacker or defender. The attacker will write mutants to not be killed by the test cases that are created by defender; The defender has to create test cases to kill the mutants that are created by attacker.

While Code Defenders was a successful mutation testing game, the study initially required at least two people to play, and it would happen that the ability gap between the attacker and defender was so large that one side would suffer from a lack of ability. Although subsequent studies have added a computer mode that can be played by one player, the difficulty of the single player mode does not have a clear difficulty index, and it is difficult to completely replace the two-player mode.

III. Game Design and Tool Implementation

In this section, we will introduce the design of our game, and its implementation.

A. *Game Design*

The following will explain the game cores, game level design, implementation of Code Immunity Boost, and implementation of Code Vaccine Incubator.

In our design, students will learn about mutation testing through the relationship between leukocyte (test case), vaccine (mutant), and human (program). Fig. 1 shows this relationship. Students can play the role of vaccine and stimulate the testing of test cases by writing mutants to improve the quality of the program. Playing the role of leukocyte and eliminate the mutants by writing test cases to improve the quality of the program. Our system design game levels and points based on the mutation score of the mutation testing and the fragility of the mutant. No matter what kind of role-playing it is, it can make students think about the relationship between the program and the test case, and then stimulate the students' learning motivation through the stimulation of achievement such as leveling and scoring, and then learn the mutation testing.

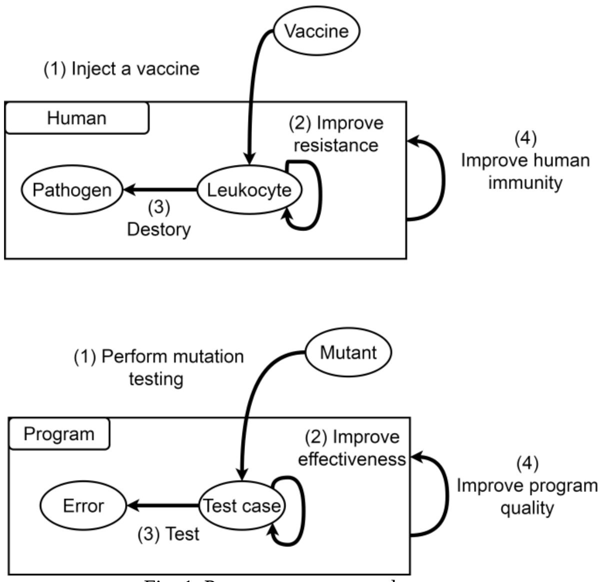


Fig. 1. Program, test case and mutant.

B. *Game Cores*

Three cores in Octalysis are embedded in our approach.

1) **Meaning** is player believes that he is doing something greater than himself. The reason for choosing meaning is to give the user a background. The background of our game is set as the pathogen is raging all over the world. The user needs to develop mutant, and let the test case kill the mutant. The test case will upgrade ability of detect errors, thereby reducing the risk of program errors.
2) **Accomplishment** is that people are driven by a sense of growth towards a goal and accomplishing it. Status points are an easily identifiable signal, and by showing small improvements, they will motivate people to move in the right direction. In this work we use the immunity (mutation score) and vaccine efficacy (1−*fragility*) to allow users to be more involved in the background of the story, but also to learn the meaning of both the mutation score and fragility at the same time; The progress bar is a way of using incomplete graphics to encourage people to fill up the blank areas in the graphics.
3) **Empowerment** brings imagination to real life. The reason for choosing empowerment is that people can play two roles leukocyte and vaccine, and use their

software knowledge and play experience to continuously improve the skill to write test cases and mutants, and the writing of test cases and mutants is varied.

C. Benchmark Programs

Many studies about mutation testing use the following programs to generate mutants, and use these mutants to conduct benchmark tests [20]. In this work we select the following programs as our benchmark examples: Triangle, Computing the median, Euclidean GCD, Fibonacci Search, Binary Search, Interpolation search and Leap year.

D. Game Level Design

In this study, the fragility is used as the standard of level difficulty, which is divided into four types of difficulty, Easy, Medium, Hard, Difficult. Users can gradually challenge from easy to difficult.

The level screenshot is shown in the Fig. 2. The user needs to pass the level to unlock the next level. (1) The level that has been cleared will be able to see the corresponding ranking, and the level that has not been cleared will display the locked symbol; (2) The level number, program, and difficulty; (3) The basic condition and better condition that each level needs to meet. The user needs to meet the basic condition of the level at least to pass the level, and the better condition are used to encourage users to challenge more difficult problems.

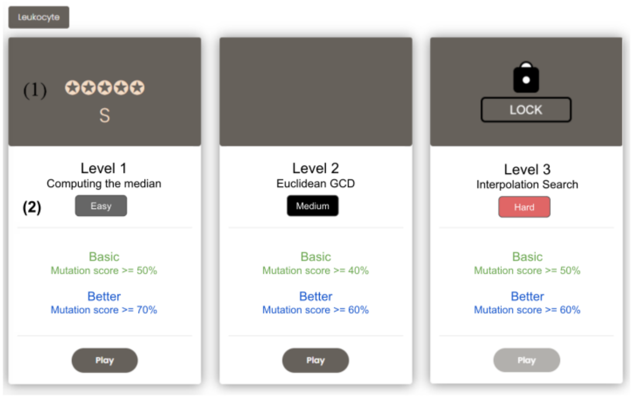


Fig. 2. Level screenshot.

E. How to Play

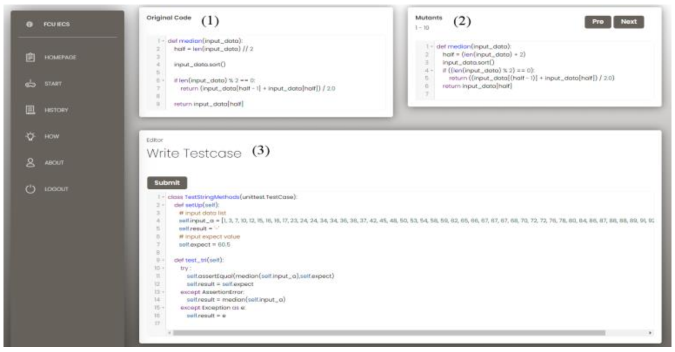
CIB is a mutation testing game based on Python and the framework unittest. Each level assigns the user to play one of the roles of leukocyte or vaccine, and immunity and vaccine efficacy to refer to the two indicators of mutation score and fragility. The purpose is to get closer to the background of vaccination to improve human immunity. Immunity is equal to the mutation score, which corresponds to the ability of leukocyte to resist mutants from invading the human body. A powerful leukocyte can kill a large number of mutants. The more effective test cases, the more errors can be detected. Vaccine efficacy is equal to 1 − *fragility*, which corresponds to the vaccine can improve the protective efficacy of the human body. Vaccine with insufficient vaccine efficacy cannot protect the human body.

When playing leukocyte, you need to follow the mutants provided by the level, and develop test cases to find out the errors of the mutants (see Fig. 3). Table I shows an example for playing leukocyte. Assuming that there are 10 mutants ($m_1 – m_{10}$), the player will pass the level if the test cases kill at least 5 mutants (immunity greater than or equal to 0.5), and will get points if test cases kill more than 8 mutants (immunity greater than or equal to 0.8). The following is a playing scenario:

1) In the first round, the user submitted the test case t_1, t_1, killed the mutants m_1, m_2, resulting in the immunity to be 0.2.
2) In the second round, the user submitted test cases t_2, t_2, killed the mutants m_1, m_3, m_4, m_5 and m_6. The immunity is 0.5, greater than the level condition. The user passed the level.
3) In the round, the user submitted test cases t_3, t_3, killed the mutants m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8. The immunity is 0.8. The user passed the level with high points.

TABLE I: LEUKOCYTE EXAMPLE

R	*TC*	*Killed Mutants*	*Immunity*	*Result*
1	t_1	m_1, m_2	0.2	Not Pass
2	t_2	m_1, m_3-m_6	0.5	Pass
3	t_3	m_1-m_8	0.8	Pass



```
def median(input_data):
    half = len(input_data) // 2

    input_data.sort()

    if len(input_data) % 2 == 0:
        return (input_data[half - 1] + input_data[half]) / 2.0

    return input_data[half]
```

```
def median(input_data):
    half = (len(input_data) + 2)
    input_data.sort()
    if ((len(input_data) % 2) == 0):
        return ((input_data[(half - 1)] + input_data[half]) / 2.0)
    return input_data[half]
```

```
class TestStringMethods(unittest.TestCase):
    def setUp(self):
        # input data list
        self.input_a = [1, 3, 7, 10, 12, 15, 16, 16, 17, 23, 24, 24, 34, 34, 36, 36, 37, 42, 45, 48, 50, 53, 54, 58, 59, 62, 65, 66, 67, 67, 67, 68, 70, 72, 72, 76, 78, 80, 84, 86, 87, 88, 88, 89, 91, 9
        self.result = ''
        # input expect value
        self.expect = 60.5

    def test_tri(self):
        try :
            self.assertEqual(median(self.input_a),self.expect)
            self.result = self.expect
        except AssertionError:
            self.result = median(self.input_a)
        except Exception as e:
            self.result = e
```

Fig. 3. Screen of playing leukocyte.

F. Code Immunity Boost

The system architecture of the code immunity boost developed in this research is shown in Fig. 4. The functions of the system are described as follows:

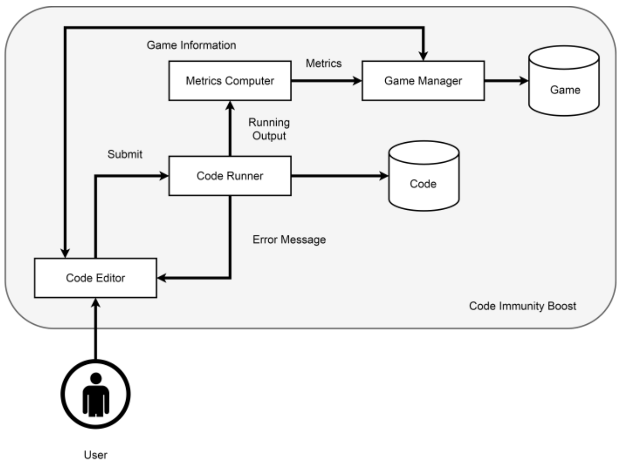


Fig. 4. The architecture of code immunity boost.

- **Code Editor:** a front-end editor for users to create test cases or mutants.
- **Code Runner:** a compiler to compile submitted test cases or mutants, and sends the execution results to Metrics Computer module.
- **Metrics Computer:** a module will execute mutation

testing, and then evaluate the mutation score of test case and the fragility of mutant.

- **Game Manager:** a module to provide information about the game to Code Editor for showing to user, calculate the score of the game and check if pass the level.
- **Code Database:** a module to save and manage user defined code.
- **Game Database:** a database used to store game information.

IV. Pilot Experiments

In order to evaluate feasibility and cost of using the CIB before formal experiment, a pilot experiment was performed, and the test cases and questionnaires generated during the experiment were used to explore the influence of the participants on the learning performance and learning motivation of the mutation testing.

A. Experimental Setting

In this experiment, feedback was collected through real operations. We want to explore *"How do students learn by using the CIB tool?"*

A total of 10 college students participated in this experiment, 3 females and 7 males. All the students were from the Department of Information Engineering of Feng Chia University, and none of them had studied mutation testing before this experiment. The students who use CIB are designated as *an experimental group*, they will design test cases to kill mutants during the game. Other students are classified as *a control group*, who would learn the mutation test using handwriting by killing mutants.

B. Result and Discussion

As the experiment is a pilot experiment to test if the tool and the process work well during the learning process. There were not many participants in our experiment. We use descriptive statistics to analyze our result. The experiment shows two results:

- The experimental group outperformed the control group in constructing valid test cases;
- Although the experimental group has high performance, its invalid rate is higher than that of control group.

The performance of building test cases was assessed by comparing the numbers of test cases submitted by the control and experimental groups. Table II shown the control group submitted a total of 18 test cases, with an average of 3.6 (18/5) for each assignment; the experimental group submitted 152 test cases totally, with an average 30.4 (152/5) for each assignment. Fig. 5 presents the great difference between the two groups.

The obvious difference in the number of submissions may be due to the fact that the control group needs to write all the mutation testing procedures by itself. In addition to designing test cases, it is necessary to perform simulated execution manually for each mutation, record the output, and then calculate the mutation score. Therefore, submission is very slow. In contrast, the experimental group only needs to design test cases, and the rest of the process is automatically completed by the CIB, therefore the submission speed is faster than that of the control group.

TABLE II: Numbers of Test Cases Submitted in Each Group

	All	*GCD*	*Search*	*Triangle*
Control G.	18	11	4	3
Experimental G.	152	13	34	105

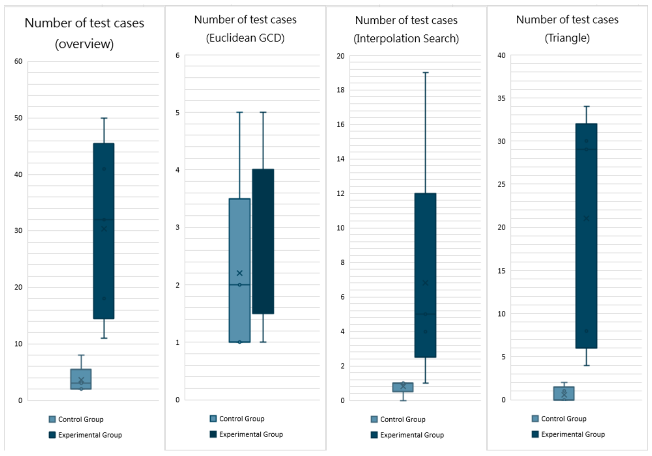


Fig. 5. Number of test cases submission.

The performance influences the willing to try new test cases. Some of the test cases designed by the students are *invalid* if they can't be compiled or executed correctly. We go further to compare the valid/invalid test cases in two groups. Fig. 6 represents the comparison.

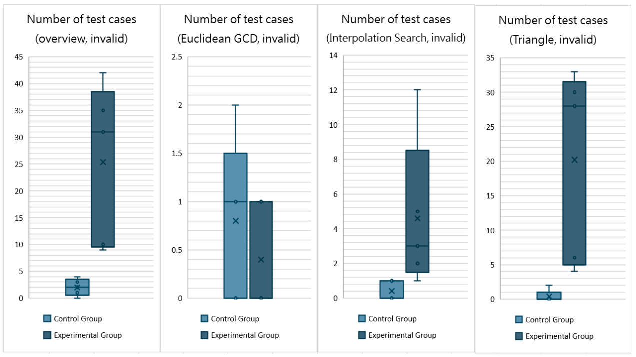


Fig. 6. Number of invalid test cases.

The control group has only 8 valid test cases, while the experimental group has 25 valid test cases, much more than the control group. Thought the experimental group has more valid test cases, its proportion of invalid test cases is higher than that of control group. We speculate that this phenomenon is due to CIB's quick feedback to students. When an error occurs, the experimental team will fix the problem as soon as possible according to the error message-even the fix is still invalid. The quick response results in high invalid rate. However, we thought this is a trial and error process for learning test case design.

As the difficulty of the questions increases, the control group has been unable to write effective test cases, and the test cases submitted tend to be invalid. In the cases of Interpolation search and Triangle, the numbers of valid and invalid submission are close to 0.

To sum up, the experimental group benefited from the automation of most of the CIB processes, and the performance and correctness of its test case design were significantly better than the control group's handwriting method.

C. Participant Satisfaction Survey

The content of the questionnaire in this experiment is based on the content of the questionnaire developed by Barrio's research [21]. The whole questionnaire is divided into three parts, namely motivation, attention and learning performance, with a total of 11 questions. The questions marked with '*' symbol are questions specific to the experimental group, and are used to observe whether our gamified tool is good for learning. We apply a 5-point Likert scale to collect students' feedback. The questions are shown in Table III.

In the *motivation* questions, as shown in Table IV, the average of the three questions in the experimental group is higher than that in the control group. The experimental group believes that not only they can learn mutation test by the tool, but the embedded gamified approach can also increase learning interest. The instant feedback mechanism in the experimental group made the participants more willing to try more test cases. In the question Q1-3 *"I think I'd like to go deeper into mutation testing"*, the control group got a very low score (3.0), which means that students lost their motivation to learn the mutation testing through the traditional approach.

TABLE III: EXPERIMENTAL QUESTIONNAIRE

Questions	
Motivation (Q1)	
Q1-1	I like this way of learning to learn mutation testing
Q1-2	I feel like this way of learning will make me more aware of mutation testing
Q1-3	I think I'd like to go deeper into mutation testing
Q1-4*	I think the level design is challenging and can increase my motivation to keep learning
Q1-5*	I prefer this way of learning over the traditional way of teaching
Attention (Q2)	
Q2-1	I feel that I have few distractions and can concentrate on this study
Q2-2	I think the process has been a high-intensity cognitive thinking
Q2-3	I feel like I am actively participating in the process
Learning outcome (Q3)	
Q3-1	I can understand the approach of mutation testing
Q3-2	I can apply the technique of mutation testing
Q3-3*	I think this teaching method has better learning effect than traditional teaching method

TABLE IV: RESULT OF MOTIVATION QUESTIONNAIRE

Q1. Motivation					
	Q1-1	Q1-2	Q1-3	Q1-4*	Q1-5*
Control G.	3.5	3.5	3.0	-	-
Experimental G.	**4.5**	**4.83**	**4.17**	4.33	4.67

In the *attention* part, the experiment group gets a higher score than the control group in questions Q2-1 and Q2-3. Q2-2 *"I think the process has been a high-intensity cognitive thinking"* is a special case—the control group get a higher score (see Table V). The result can reflect that the handwriting method will force the participants to be think deeper. Even though the process to compare the output of mutants is difficult for learners, it is impressive the process to understand the mutation process. In the question Q2-1 *"I feel that I have few distractions and can concentrate on this study",* the control group get low score (3.5 points), which shows that the computation is really hard for learners.

TABLE V: RESULT OF ATTENTION QUESTIONNAIRE

Q2. Attention			
	Q2-1	Q2-2	Q2-3
Control G.	3.5	**4.5**	4.5
Experimental G.	**4.5**	4.17	**4.83**

Table VI shows the results in the *Learning Performance* part. Three questions are designed to collect students' feedback on learning performance. Both groups felt they could understand and apply the concept and techniques of mutation testing. The experimental group also expressed high interest in the CIB tool—they believed that this gamified learning method is a good way of learning than the traditional lecturing approach.

In general, both the control group and the experimental group were satisfied with learning mutation testing. After experiencing this experiment, they would like to understand the technology of mutation testing, and even want to challenge more difficult problems. The control group mainly felt the pressure on the handwriting method, hoping to reduce the number of mutants to make the handwriting process a little easier, while the experimental group did not think that the number of mutants was too large at all, but expressed more interest in the CIB.

TABLE VI: RESULT OF LEARNING PERFORMANCE QUESTIONNAIRE

Q3. Learning performance			
	Q3-1	Q3-2	Q3-3
Control G.	**4.75**	4.0	-
Experimental G.	4.67	**4.33**	4.83

V. CONCLUSION

Mutation testing is a software testing technique that has been proven to effectively improve the effectiveness of detecting errors in the test cases. However, in the actual educational environment, there are few appropriate teaching materials to support the teaching of mutation testing, which reduces students' chances of learning mutation testing. In this work we develop the code immunity boost (CIB), a gamification system developed based on the concept of pathogen mutation.

A pilot experiment was conducted to evaluate our approach. The control group used handwriting to learn the mutation testing, while the experimental group used our developed gamified CIB tool to learn. We also conducted a questionnaire to collect students' feedback from the viewpoints of motivation, attention and learning performance. The result shows that our tool can provide an effect method for learning mutation testing. Although learning without our tool can make students impress with the knowledge of mutation testing, but because the process is too difficult and no fun, they lose their motivation to learn.

In the field of software testing research, most of them propose testing techniques to improve the effectiveness of testing, and seldom use gamification mechanisms and tools to promote software testing education. We believe our method can serve as a reference for this type of research. In the future, we plan to integrate our approach with the more widely used Online Judge system, through which we can make software engineering education more robust.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Hsueh conducted the research; Xuan implemented the system; Daramsenge wrote the paper.

Funding

This work was supported by the Ministry of Science and Technology, Taiwan, under grant MOST 110-2221-E-035-031-MY2.

References

[1] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal, "A survey on online judge systems and their applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, p. 3, 2018.

[2] A. Knutas, J. Ikonen, and J. Porras, "Computer-supported collaborative learning in software engineering education: A systematic mapping study," *arXiv preprint arXiv:1906.10710*, 2019.

[3] M. Shaw, "Software engineering education: A roadmap," *ICSE-Future of SE Track*, pp. 371–380, 2000.

[4] M. Souza, R. Moreira, and E. Figueiredo, "Students perception on the use of project-based learning in software engineering education," in *Proc. the XXXIII Brazilian Symposium on Software Engineering*, pp. 537–546, 2019.

[5] M. Milosz and E. Milosz, "Gamification in engineering education–a preliminary literature review," in *Proc. 2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1975–1979, 2020.

[6] Y.-F. Xu and L.-S. Shi, "Research and design of app for primary school students' safety education based on embodied cognitive theory," in *Proc. 2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI)*, pp. 1–4, 2020.

[7] W. Chao, C. Yang, S. Hsien, and R. Chang, "Using mobile apps to support effective game-based learning in the mathematics classroom," *International Journal of Information and Education Technology*, vol. 8, no. 5, pp. 354–357, 2018.

[8] D. Ding, C. Guan, and Y. H. Yu, "Game based learning in tertiary education: A new learning experience for the generation z," *International Journal of Information and Education Technology*, vol. 7, no. 2, p. 148, 2017.

[9] L. C. Kho, S. S. Ngu, A. Joseph, D. A. A. Mat, L. Y. Ng, and J. L. Hau, "Gamification approach towards engineering students' engagement in online learning," *International Journal of Information and Education Technology*, vol. 12, no. 6, 2022.

[10] J. Jitsupa, M. Takomsane, S. Bunyawanich, N. Songsom, and P. Nilsook, "Combining online learning with gamification: An exploration into achievement, motivation, and satisfaction of the undergraduate," *International Journal of Information and Education Technology*, vol. 12, no. 7, 2022.

[11] J. Kummanee, P. Nilsook, and P. Wannapiroon, "Digital learning ecosystem involving steam gamification for a vocational innovator," *International Journal of Information and Education Technology*, vol. 10, no. 7, pp. 533–539, 2020.

[12] N. Willert, "A systematic literature review of gameful feedback in computer science education," *International Journal of Information and Education Technology*, vol. 11, no. 10, pp. 464–470, 2021.

[13] D. N. Prata, P. Letouze, S. Cerri, and E. Costa, "A game approach to assessing learning outcomes," *International Journal of Information and Education Technology*, vol. 6, no. 2, p. 137, 2016.

[14] T. A. Shimoda and M. Borge, "The web of inquiry: Computer support for playing epistemic games," *International Journal of Information and Education Technology*, vol. 6, no. 8, p. 607, 2016.

[15] R. A. Mauricio, L. Veado, R. T. Moreira, E. Figueiredo, and H. Costa, "A systematic mapping study on game-related methods for software engineering education," *Information and Software Technology*, vol. 95, pp. 201–218, 2018.

[16] J. M. Rojas and G. Fraser, "Code defenders: a mutation testing game," *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 162–167, 2016.

[17] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer*, vol. 11, no. 4, pp. 34–41, 1978.

[18] J. M. Rojas, T. D. White, B. S. Clegg, and G. Fraser, "Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game," in *Proc. 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pp. 677–688, 2017.

[19] G. Fraser, A. Gambi, M. Kreis, and J. M. Rojas, "Gamifying a software testing course with code defenders," in *Proc. the 50th ACM Technical Symposium on Computer Science Education*, pp. 571–577, 2019.

[20] Y. Jia and M. Harman, "Higher order mutation testing," *Information and Software Technology*, vol. 51, no. 10, pp. 1379–1393, 2009.

[21] C. M. Barrio, M.-O. Mario, and J. S. Soriano, "Can gamification improve the benefits of student response systems in learning? An experimental study," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 429–438, 2015.