Kashihara, A. et al. (Eds.) (2024). Proceedings of the 32nd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

A Proposal of Quality Assurance Programming Exercise

Nobuya ISHIHARA[a*], Samsul HUDA[a] & Yasuyuki NOGAMI[b]
[a]*Green Innovation Center, Okayama University, Japan*
[b]*Graduate School of Environmental, Life, Natural Science and Technology, Okayama University, Japan*
*n.ishihara@okayama-u.ac.jp

Abstract: This paper proposes a new exercise format for programming assignments called Quality Assurance Programming Exercises (QAEx), which incorporates the use of generative artificial intelligence (AI). In QAEx, students are allowed to use AI tools to generate solutions for programming problems and are also required to submit comprehensive test cases to validate these solutions. Students can use AI to create these test cases as well. The inclusion of AI in both solution development and testing underscores the importance of effective software testing and encourages students to engage more deeply with programming and quality assurance practices. A preliminary trial of QAEx was conducted to assess students' engagement and performance.

Keywords: programming education, AI, test codes, AI hallucinations, Quality Assurance Programming Exercises

1. Introduction

AI in education has evolved to include web systems, robots, and chatbots, assisting with grading, improving teaching quality, and customizing the curriculum (Chen, et al., 2020).

Today, generative AI is being used by students for report writing and solving assignments, independent of educational institutions. Instead of banning the use of AI in traditional hands-on exercises, we propose customization.

2. Traditional Programming Exercise

Figure 1 illustrates the common stages of the two typical system development methodologies, the 'Waterfall Model' and 'Agile Development,' (Stoica, et al., 2016) and shows the relationship between specifications, programs, and test specifications. In traditional programming exercises, a simple specification given as an assignment is used to create a program. Due to the difficulty of creating test specifications, which encompass the two major types of testing: white-box testing and black-box testing, they are rarely created by learners. Instead, they are often used for grading assignments or as supplements to assignment descriptions.

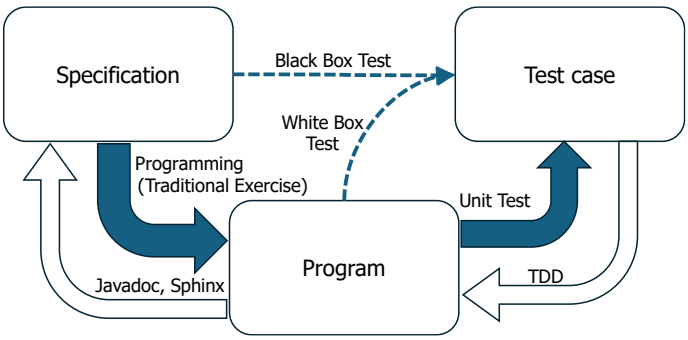

Figure 1. Three Steps for Programming.

In recent years, generative AI has become capable of generating programming code with increasing accuracy. Therefore, we propose QAEx, a method that utilizes AI to help learners create test code as well.

3. Proposal Programming Exercise (QAEx)

In QAEx, learners use AI to prepare test code and submit the source code, which has passed the tests, along with the test code. The implicit requirements for learners when submitting are as follows:

Completion of unit tests: The completion of tests ensures that the source code functions correctly for the tested items. If the tests are not completed, the process cannot proceed.

Review of source code: If the tests are not completed or the test specifications are insufficient, learners must review and modify the source code to add necessary test items.

Review of test code: The test code may also contain errors, so it needs to be reviewed.

Increasing **the number of test items**: For tests to be considered complete, there must be a meaningful number of test items. It is necessary to consider boundary conditions and coverage, and increase the number of test items accordingly. Here, a higher number of test items is deemed better.

4. Preliminary Trial Exercise (Small-Scale Pilot Study)

Table 1. *A Result of the Small-Scale Trial Exercise*

	A	*B*	*C*	*D*	*E*
ex01 (easy)	111(9)	100(1)	111(1)	101(5)	111(7)
ex02 (middle)	110(5)	101(3)	111(1)	111(5)	101(1)
ex03 (difficult)	100(1)	100(2)	111(1)	111(3)	010(1)
ex04 (difficult)	100(1)	100(1)	111(1)	111(2)	010(1)

Table 1 shows how the A~E students behaved in response to the four tasks. The three digits 1/0 indicate that the unit tests were completed/not completed, the source code was reviewed/not reviewed, and the test code was reviewed/not reviewed, and the numbers in parentheses indicate the number of test items.

In many assignments/persons, the source code or test code was reviewed after the tests were run, which was as expected. In E-ex03 and E-ex04, the test is incomplete; in A-ex04 and B-ex04, the test is complete but the number of test items is submitted with 1. The former indicates that the assignment solution is difficult for E even with the support of AI, while the latter indicates that the source code and test code generated by AI were tested as is and completed, so they are submitted. Additionally, our findings indicate that some learners utilize print statements as test code. Furthermore, AI hallucination code was identified in ex03 and ex04. Notably, the AI returned an answer in the form of a file rather than a string on the screen.

5. Conclusion and Future Tasks

We propose a new exercise format called "Quality Assurance Programming Exercises (QAEx)," where students are allowed or encouraged to use AI to solve exercise problems. Instead of merely submitting the solutions, students are required to submit test cases (test code) along with their solutions. In our preliminary trial exercise, we found that difficult question such that AI return hallucination-code, are also difficult in our exercise, too.

In the future, we will introduce a copy-paste/D&D type tool on the browser so that in addition to this theme, we can use unittest, a Python test environment, naturally while checking the code to be submitted on the screen. We would like to verify the extent to which test code generation by AI can be used.

References

Chen, L., Chen, P., Lin, Z. (2020). Artificial Intelligence in Education: A Review. *IEEE Access*, 8, 75264-75278. https://doi.org/10.1109/ACCESS.2020.2988510

Stoica, M., Ghilic-Micu, B., Mircea, M., Uscatu, C. (2016). Analyzing agile development-from waterfall style to scrumban. Informatica Economica, 20(4), 5.,5-14. https://doi.org/10.12948/issn14531305/20.4.2016.02