Use of Generative Artificial Intelligence in the Education of Software Verification and Validation

Ayca Tuzmen
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe, Arizona
atuzmen@asu.edu

Abstract*—Since the introduction of generative artificial intelligence (GenAI), education in computer science has prompted efforts to incorporate it into the educational curriculum. This innovative practice full paper presents a study into using GenAI to enhance student learning of software engineering. It outlines the initiatives to introduce GenAI into a graduate-level software engineering course in Software Verification and Validation (SV&V). The paper presents the educational goals, methodologies and findings of these endeavors in this course. The primary education goal of this course is that students have a solid understanding of principles and practices of software quality assurance and seek to introduce students to diverse techniques employed for SV&V. The study presented in this paper centers on the practical application of GenAI within the domain of testing strategies. The paper introduces the findings of an exercise where GenAI was used to apply testing strategies for unit testing. The exercise consisted of the use of GenAI in the development of unit tests for an algorithm. Rigorous assessments were conducted to gauge the effectiveness of the unit tests developed for validating the accurate implementation of the algorithm. This exploration shed light on the tangible impact of GenAI on the precision and efficiency of unit testing procedures. The findings underscore the significance of encouraging students to actively explore emerging trends and methodologies in the realm of software verification and validation. By incorporating GenAI into the educational framework, students not only gain insights into the capabilities and limitations of this technology but also foster a mindset of continuous learning in software quality assurance. The paper demonstrates that it is not sufficient to use the test cases developed by GenAI for software validation since test cases recommended by GenAI do not cover corner cases which causes gaps in coverage in unit testing. The majority of the students were able to understand the limitation of GenAI in SV&V but appreciated its support in suggesting test cases for the most common cases. This exercise allowed students to enhance their creative problem-solving through human-guided AI partnership which is pivotal in cultivating a new generation of professionals capable of contributing to the ongoing evolution of software quality.

***Keywords*—generative artificial intelligence, software verification and validation, unit testing**

I. Introduction

The field of software engineering is continuously evolving, driven by advancements in technology and changing industry demands. There is a growing need to equip future software engineers with the knowledge and skills necessary to tackle real-world challenges effectively. Generative Artificial Intelligence (GenAI) has emerged as a powerful tool with applications across many domains in software engineering. GenAI has made a significant impact on software engineering and offers a promising avenue for enhancing software development. It is shown to be applied in various phases of the software engineering process to support or perform various tasks such as requirements gathering [1], design [2], implementation [3], testing [4] and maintenance [5].

In the evolving landscape of education, faculty are continuously seeking innovative methods to engage students, enhance their learning objectives and prepare students for the expectations of the software industry. Artificial Intelligence in Education (AIED) is one of the currently emerging fields in education technology. Horizon Report [6] identifies GenAI as a key technology in teaching and learning. GenAI is finding its way into classrooms, whether intentionally or unintentionally and its impact is multifaceted. It is shown to both help and hurt teaching and learning. GenAI has helped teaching and learning in areas such as automation of repetitive tasks, data analysis and insights, self-paced tutoring, grading, educational data mining, enhancing accessibility and predictive and personalized learning [7]. On the other hand, the use of GenAI in classrooms without a well-thought integration plan is hurting teaching and learning in academic integrity, creativity, bias and ethics [8]. Learners are using GenAI to cheat on course assignments and assessments. Students are relying on AI to do the work for them potentially leading to learning loss [8]. Some stakeholders claim that GenAI will lead to the end of creative expression and individual thought. Others claim it increases human creativity and boosts productivity [9].

This paper presents an exploratory study into the integration of GenAI into the education of graduate-level courses in software engineering. It introduces the design, implementation and observations, highlighting the pedagogical strategies employed and their impact on student performance. As part of the study, new exercises were designed to enhance student engagement and learning objectives in the graduate-level course on Software Verification and Validation (SV&V) at a large American university during the 2023-2024 academic year.

The motivation behind this initiative stemmed from the growing importance of AI technology in various fields and the inclination of higher-level education institutes to equip students with relevant skills and knowledge in the use of AI technology. However, this motivation later followed intending to test out and

reflect upon the pedagogical strategies in teaching software verification and validation (SV&V).

The study presented here illustrates the potential applications of GenAI in teaching SV&V concepts. A new AI-based exercise was designed and implemented where certain pedagogical strategies were tested, and observations were made regarding the impact of GenAI on the achievement of the learning objectives of a course. This is a descriptive study rather than a prescriptive one. Nonetheless, conclusions are still drawn based on observations made during these exercises.

The paper is divided into four sections. The first section discusses the pedagogical strategies addressing the use of GenAI in the enhancement of student learning in higher-level education. The second section introduces the construct of one of the exercises redesigned to integrate GenAI, including its structure and learning objectives. In the third section, observations of the outcome of the exercise, analysis of the student performance and student's feedback on their experience with AI technology is presented. The final section discusses the successes and challenges during the study with insights into potential improvements and future directions for integrating GenAI into the educational process.

II. Pedagogical strategies

Generative AI is recommended to be used in education to support various pedagogical principles [8] [10]. Its applications extend to the realm of software engineering education as well. Literature consists of exploratory studies where different ideas, prompts and pedagogical principles are presented [11][12]. Additional studies focused on the prescription of some techniques to support pedagogical principles including guidelines developed for using GenAI in implementing teaching strategies [13].

One of the pedagogical principles practiced in this study is role-playing. Role-play is a powerful pedagogical strategy that involves students acting out roles in specific scenarios to explore concepts, practice skills and gain deeper understanding through experimental learning. In an educational context, it is not always possible to bring in individuals to play these roles. Instead, students are asked to play different roles in group and team settings. However, in some cases, they may not be able to fully understand the responsibilities of such roles. Therefore, instructors need to detect and identify such shortcomings and, in some cases, must play such roles themselves.

Through role-play, students can practice theoretical knowledge in a practical context apply critical skills and make decisions based on their roles. Studies show that there is a close correlation between students and their peers [14]. The effect of peer learning in education has long been noted and thus been utilized to enhance student's learning [15].

AI technology can be designed to play specific roles during various stages of the software engineering process. GenAI can be used to simulate to play the role of a peer. It can be asked to act to play the role of a customer, designer, tester, or programmer. In this study, GenAI is used to play the role of a peer programmer and a tester in the design and development of test cases. Through interaction with GenAI, the learners were able to interact with another peer and work collaboratively in the development of test cases which are used for the verification and validation of an implementation for an algorithm.

Another principle that is practiced in this study is evidence-based learning. Evidence-based learning is shown to be a powerful technique used to support learning through simulation by allowing students to apply a framework in a new situation. By allowing access to data and resources, students can identify missing data or gaps in a solution. This allows them to focus on designing solutions for the gaps, allowing them for targeted interventions.

III. Methodology

The author embraces role-play and evidence-based learning strategies as a methodology for the integration of GenAI into the education of software engineering. This section introduces the exercises that are redesigned and executed as part of the graduate-level course at a large American university. The course is entitled CSE 565 Software Verification and Validation. In the CSE 565 course, a flip-style teaching strategy is adopted. Students watch lecture recordings and attend live lectures to review and further discuss the course topics. Learners complete knowledge checks, quizzes and exams and complete five (5) assignments individually.

Two of these assignments were redesigned so that students can complete the assignment with support from GenAI. The objectives of the exercises were to introduce GenAI into the teaching and learning environment and allow students to explore, observe and assess the effectiveness and limitations of generative AI in meeting teaching and learning objectives. This paper introduces the findings from the design, execution and analysis of only one of these exercises.

The focus of the paper is on the exercise where students practice the design of test cases for a software using a testing framework during unit-level testing. Unit testing framework is a tool that provides a structured and automated way to test individual units of source code. Unit testing is completed by the developer of the source code with the intention to validate that each unit of software performs as expected. If it is not done effectively, delay in integration of other individual source code and entry to following test cycles are blocked.

A new AI-based exercise was designed to support students in the:

- Applying unit-level testing learning principles to develop tests for an algorithm,
- Developing and executing test cases using a unit testing framework

In this exercise, the students were asked to develop test cases implemented using a unit testing framework for the validation of an algorithm. They were expected to download or write code to implement the Heapsort algorithm using a high-level programming language of their choice such as Java, C++, Python, or JavaScript language. The tasks that were completed included:

- Task 1- Identify or create an algorithm for the heapsort algorithm

- Task 2 - Research and identify a unit testing framework to test their code
- Task 3- Research and use a GenAI tool to create unit-level test cases using a unit-test framework
- Task 4- Execute and report on the outcome of the tests created
- Task 5 - Analyze the results of the test execution and assess the validity of the test cases created
- Task 6- Write additional test cases themselves to further improve the coverage and performance of the test cases
- Task 7- Assess and comment on the effectiveness of the GenAI tool in creating unit-level test cases using the unit-level testing framework

The exercise did not provide and thus required the use of a template or a starting prompt to be used when interacting with GenAI. It was assumed that students have interacted with a GenAI platform before and thus they know how to create prompts and retrieve a response. Therefore, the students were not given any training on prompt engineering. It was left up to the students to define and design the prompt required for completing the given tasks. Some students who needed further assistance in creating prompts attended staff office hours and brief guidance was provided regarding how to work with the selected GenAI.

This exercise allowed students to co-create test cases with GenAI where it played the role of a peer where students take the recommendation of a peer, assess the effectiveness of the recommendation and further improve it so that it meets the objectives of software validation. It was expected that students execute the recommended test cases and report the outcome of the execution. The students were instructed to review the design of test cases, analyze the outcomes of the execution of the test cases and assess how well the test cases validate the functionality and performance of the implemented algorithm. In case they observe any gaps or identify coverage, the students were to develop additional test cases. In the final part of the exercise, students assessed how well the GenAI was in generating unit-level test cases using the given unit test framework and the correctness of the code for the given algorithm.

IV. Analysis

The newly designed exercise was completed by graduate-level students during the Fall and Spring semesters of the 2023-2024 academic year. The analysis of the outcomes of this exercise is based on 290 reports submitted during this time. The reports were collected from 180 graduate-level students during the Fall and Spring semesters of 2023-2024.

The analysis of the outcomes of this exercise is done based on 1) the type of unit framework used 2) the coverage of the unit tests 3) the effectiveness of the test cases 4) assessments of the GenAI in creating unit-level testing.

It is observed that most of the students used ChatGPT 3.5 and the rest used Bard-AI for the completion of this exercise.

For the completion of Task 1, Java or Python programming languages were mostly used for implementing the heap-sort algorithm (Fig. 1).

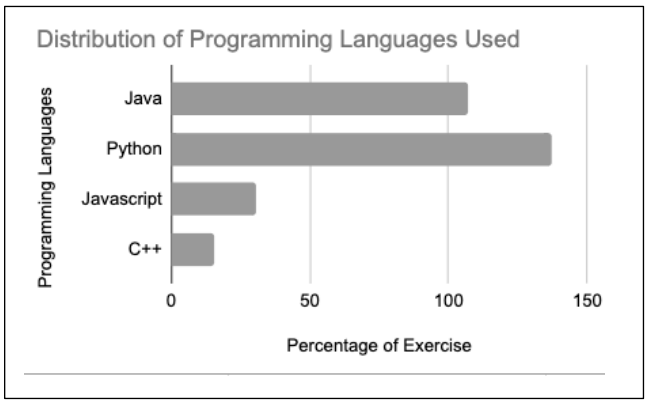

Fig. 1. Distribution of programming languages used for the implementation of the algorithm

For completion of task 1, most of the students created the code for the heap sort algorithm from public websites. Others asked GenAI to create the code of the algorithm. The responses of the GenAI tool included code written in the corresponding programming language using a unit testing framework. Some of the examples of such code can be seen in Fig 3. And Fig 4. As shown in these figures, GenAI was able to produce code for the algorithm given to GenAI using the framework requested.

```
no usages
public void heapSort(int arr[]) {
    int n = arr.length;

    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i,  i: 0);
    }
```

Fig. 3. A sample code created by GenAI for the heapsort algorithm using Java

```
def heapSort(arr):
    n = len(arr)

    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)

    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        heapify(arr, i, 0)
```

Fig. 4. A sample code created by GenAI for the heapsort algorithm using Python

For the completion of Task 2, Junit, Pitesti and UnitTest are some of the unit testing frameworks used. Fig. 5 shows the percentage of the type of unit frameworks used in this exercise.

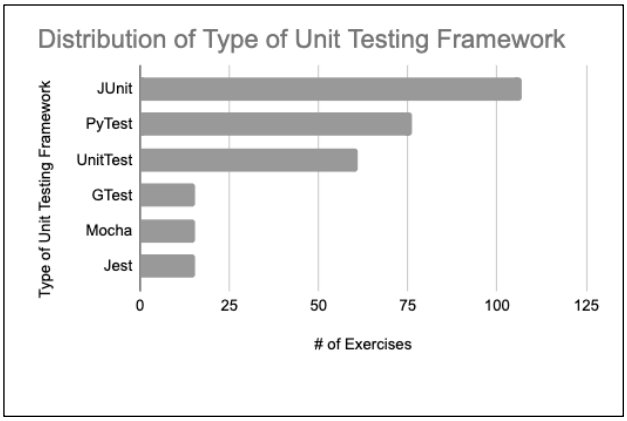

Fig. 5. Distribution of unit framework used for creating unit tests

For completing Task 3, students used prompts that included a code implementation of the Heapsort algorithm followed by a statement that requested the development of several unit test cases using a specific unit testing framework. In some cases, the student did not mention which unit testing framework to use. However, GenAI was able to select an appropriate framework and develop several unit test cases. Some of the examples of the prompt used when integrating with the GenAI tool were:

- "Write Junit test cases for the code provided by you above",
- "Generate five unit-level corner test cases for Heapsort algorithm in Python",
- "Give me unit tests for this code using Jest covering all possible scenarios".

Fig. 5 shows a sample output of ChatGPT in response to a prompt where the student prompted the development of unit tests without giving the code implementation and specifying the number of test cases needed. In this response, GenAI only lists the specs of the test cases with required inputs and expected output without any code implementation. The test cases were limited by the number specified in the prompt. Another response by GenAI included generated code for unit test cases with the given framework and programming language (Fig. 6).

For the completion of Task 4, students executed the recommended test cases using an Integrated Development Environment (IDE) of their choice which included Microsoft Visual Studio, Anaconda and Eclipse. From the execution of the recommended test cases, it is observed by students that all recommended test cases are valid and can be executed with correct and expected results. No compile time errors were observed when executing the recommended code.

Fig. 6. A sample response by GenAI with the test case specs

```
class TestHeapSort(unittest.TestCase):

    def test_empty_list(self):
        arr = []
        result = heapSort(arr)
        self.assertEqual(result, [])

    def test_sorted_list(self):
        arr = [1, 2, 3, 4, 5]
        result = heapSort(arr)
        self.assertTrue(is_sorted(result))

    def test_reverse_sorted_list(self):
        arr = [5, 4, 3, 2, 1]
        result = heapSort(arr)
        self.assertTrue(is_sorted(result))

    def test_unsorted_list(self):
        arr = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
        result = heapSort(arr)
        self.assertTrue(is_sorted(result))
```

Fig. 7. Sample unit tests created by GenAI with the code implementation

As the outcome of Task 5, most of the students were able to identify gaps and recommended that additional tests should be developed for full test coverage of the given code. A few number

of the students concluded that test cases developed by GenAI have thorough coverage and did not recommend additional test cases. They concluded that the test cases recommended by GenAI were sufficient for unit-level testing.

It is observed that most of the students realized that the unit-level tests recommended by GenAI focused solely on the functionality of the sorting algorithm. These students identified gaps and designed additional tests for the verification of sorting in the following cases:

- An array of integers with zero or negative values
- An array with null/none values
- An array of integers in descending order
- An array of other data types such as floating numbers, characters, strings and objects
- An array with minimum and maximum integer values
- An array with a combination of integers and strings
- An array of positive integer numbers with only one element

Additional tests were also recommended by students to test the error-handling functionality of the algorithm such as for invalid inputs using special characters. Students also recommended non-functional performance testing of the unit code. Additional tests were developed for handling large sizes of arrays. Table 1 shows the percentage of students who created additional test cases on top of the ones that were recommended by GenAI. This table illustrates that the highest percentage of additional tests recommended by students included corner test cases such as tests with large arrays, arrays with single elements, arrays with identical elements, or arrays in descending order.

TABLE I. PERCENTAGE OF NEWLY DESIGNED TESTS

Type of Test	*Percentage*
SortWithLargeArray	89%
SortWithNegativeNumbers	89%
SortInDescendingOrder	89%
SortSingleElementArray	89%
SortIdenticalElementsArray	89%
SortNonIntegerTypeArray	79%
SortNullTypeArray	63%
SortMinMaxBoundary	26%
SortWithNegativeZeroInFloatingNumbers	6%

The outcomes of Task 7 included the students' assessments of the GenAI tool in creating unit-level testing. Some of the highlights of the students' assessment include the following:

- Just enough coverage - The majority of the students concluded that GenAI produced just enough test cases. One student's assessment stated that "AI is beneficial for quickly outlining high-level test cases covering core functionality".
- No corner case coverage - did not achieve extensive coverage for special cases and corner cases. GenAI is missing edge cases, unusual scenarios and failure points
- Lack of domain expertise, creativity and experience - Human capability in writing test cases is found to be valuable because people apply domain expertise, creativity and experience when designing test cases.
- Useful and effective tool - Even though students were able to find gaps in test coverage when working with GenAI, the use of GenAI in test development was found to be useful and effective.

V. DISCUSSION

The findings of the study suggest that the students had varying types of experiences from the interaction with the GenAI. When GenAI played the role of a peer developer, GenAI played the junior as well as the senior level of expertise in the design of unit-level test cases. GenAI has recommended to some students the test cases which were not recommended to other students. Some of the prompts provided by the students were limiting the number of tests the GenAI can recommend to the student. However, even if there was no limitation on the number of test cases needed, some tests were still missing. As shown in Table 1, missing test cases were mostly observed in the testing of corner cases or edge cases.

The variation in outputs of GenAI prompted an investigation into the following question:

- Why did GenAI recommend different test cases for different users?

The author investigated this question and made a further inquiry into this question:

- Is there a correlation between the nature of the prompts provided, the programming language and unit test framework selected, the GenAI tool interacted and the type of test cases developed by the GenAI tool?

After careful analysis of the study, the author found no significant correlation found between the type kind of prompts used and the coverage recommended. Regardless of which programming languages were used (i.e. Java, Python, C++, or JavaScript) or which unit framework was selected (Junit, PyTest, UnitTest), or which GenAI tool (ChatGPT or Bard-AI), the test cases recommended by GenAI was not always the same number of test cases and type of coverage. For example, for some students, GenAI recommended test cases for only integer numbers and for others, it suggested tests for different data types.

The author argues that this is due to the very large set of textual data used in the development of the machine learning (ML) model used in GenAI. Once the model has been trained, GenAI adapts to specific tasks such as answering questions or generating text in a particular domain [16]. Fine-tuning is another technique used to adapt a pre-trained machine learning model to a specific task by re-training it on a smaller dataset

specific to that task. Thus, GenAI was able to leverage learning in pre-training and by fine-tuning a large amount of textural data.

VI. Conclusion

The integration of GenAI in education requires that we go through the five stages described by Rogers [14]. As per Roger's framework for diffusion of innovation, the first stage which is the creation of an awareness of the innovation but lacking complete information about it has already happened. Students are aware of GenAI and have been using it since ChatGPT was first introduced in 2022. Interest is the second stage of diffusion of innovation and there is already a growing interest and information seeking about GenAI. In the third stage, evaluation of innovation is done and the decision on whether to try innovation based on the present and future situation is made. Some higher education institutions are aware of the impact of GenAI and already decided to try GenAI to support teaching and learning. However, there are teaching faculty in engineering education who decided to ban the use of GenAI in their course because of some of the misuse and threats they foresee. The fourth stage requires the trial and making use of the innovation. The final stage of diffusion of innovation requires the adaptation and continued use of the innovation.

This paper describes all the five stages that were followed in the diffusion of GenAI in the education of software verification and validation. A conscious decision was made to incorporate GenAI since there was great awareness and interest by students to use GenAI to do their assignments, quizzes and exams. However, due to a lack of complete information about the shortcomings of GenAI, students were solely depending on the outcomes produced by GenAI. Through trial and adaptation of updated exercises in their SV&V course, they were able to observe and assess the potential and shortcomings of GenAI.

The use of GenAI can be a game-changer when it comes to generating code and developing test cases. However, despite its promising capabilities, the question of trusting GenAI as-is raises questions and concerns. GenAI can cause hallucinations regarding the correctness of a code or test case. However, it can be used to create evidence in the application of a framework and the generation of data as an outcome of this application. By allowing students to collaborate with AI technology, they can co-create a solution and assess the effectiveness of a solution by observing the evidence related to that solution.

References

[1] K. Liu, S. Reddivari and K. Reddivari, "Artificial Intelligence in Software Requirements Engineering: State-of-the-Art," 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), San Diego, CA, USA, 2022, pp. 106-111.

[2] S. Martínez-Fernández et al, "Software Engineering for AI-Based Systems: A Survey." ACM Transactions on Software Engineering and Methodology, vol. 31, no. 2, 2022, pp. 1–59, [Online]. Available: https://doi.org/10.1145/3487043.

[3] R. Pudari and N, A. Ernst, "From Copilot to Pilot: Towards AI Supported Software Development". 2023, [Online]. Available: https://doi.org/10.48550/arxiv.2303.04142.

[4] Z. Khaliq et al, "Artificial Intelligence in Software Testing: Impact, Problems, Challenges and Prospect." arXiv.Org, 2022, https://doi.org/10.48550/arxiv.2201.05371.

[5] H. Alsolai and M. Roper. "A Systematic Literature Review of Machine Learning Techniques for Software Maintainability Prediction." Information and Software Technology, vol. 119, 2020, pp. 106214-, [Online]. Available: https://doi.org/10.1016/j.infsof.2019.106214.

[6] K. Pelletier et al, "2024 EDUCAUSE Horizon Report, Teaching and Learning Edition". Educause, May 2024, Accessed: May 19, 2024. [Online]. Available: https://library.educause.edu/resources/2024/5/2024-educause-horizon-report-teaching-and-learning-edition

[7] T. Fitria, "Artificial intelligence (AI) in education: Using AI tools for teaching and learning process." In Prosiding Seminar Nasional & Call for Paper STIE AAS, 2021, pp. 134-147.

[8] C. K. Y. Chan, "A comprehensive AI policy education framework for university teaching and learning." International journal of educational technology in higher education 20.1, 2023.

[9] E. Zhou and D. Lee, "Generative artificial intelligence, human creativity and art", *PNAS Nexus*, 3.3, 2024.

[10] H. Crompton and D. Burke. "Artificial intelligence in higher education: the state of the field." International Journal of Educational Technology in Higher Education 20.1, 2023.

[11] G. Cooper. "Examining science education in ChatGPT: An exploratory study of generative artificial intelligence." Journal of Science Education and Technology 32.3, 2023, pp. 444-452.

[12] A. Farazouli et al, "Hello GPT! Goodbye home examination? An exploratory study of AI chatbots impact on university teachers' assessment practices." Assessment & Evaluation in Higher Education 49.3, 2024, pp. 363-375.

[13] E. Mollick and L. Mollick. "Using AI to implement effective teaching strategies in classrooms: Five strategies, including prompts." Including Prompts, March 17, 2023.

[14] B. Sacerdote, "Peer Effects in Education: How Might They Work, How Big Are They and How Much Do We Know Thus Far?" Handbook of the Economics of Education, vol. 3, 2011, pp. 249–77, [Online]. Available: https://doi.org/10.1016/B978-0-444-53429-3.00004-1.

[15] J. S. Coleman et al, "Equality of Educational Opportunity", United States of Health, Education and Welfare, Office of Education, 1966, [Online]: https://files.eric.ed.gov/fulltext/ED012275.pdf

[16] E. Sarrion, "Exploring the Power of ChatGPT Applications, Techniques and Implications", 1st ed. 2023., Apress, 2023,