Assessing Software Engineering Students' Analytical Skills in the Era of Generative AI

Olga Petrovska
Swansea University
Swansea, United Kingdom
olga.petrovska@swansea.ac.uk

Rebecca Pearsall
Open University
Milton Keynes, United Kingdom
rebecca.pearsall@ou.ac.uk

Lee Clift
Swansea University
Swansea, United Kingdom
l.a.clift@swansea.ac.uk

Abstract

This poster showcases an assessment designed to develop and evaluate software engineering students' code analysis skills. We demonstrate how students approached code analysis tasks when given multiple code samples created by a human and various AI tools.

CCS Concepts

• **Applied computing** → **Education**; • **Social and professional topics** → **Software engineering education**; • **Computing methodologies** → *Artificial intelligence*.

Keywords

generative AI, software engineering, education, apprenticeship

ACM Reference Format:
Olga Petrovska, Rebecca Pearsall, and Lee Clift. 2025. Assessing Software Engineering Students' Analytical Skills in the Era of Generative AI. In *Computing Education Practice (CEP '25), January 07, 2025, Durham, United Kingdom.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3702212.3702223

1 Introduction

In recent years the education community has been actively trying to understand the implications of Generative AI on the education practice. When teaching software engineering, it is impossible to look at AI in isolation from the industry. We need to prepare our students for the future of GenAI-enabled software design and development [2]. The question is: how can we help students acquire and maintain relevant programming skills in a world of ChatGPTs and Copilots, that can generate working code in seconds?

If we want students to produce good software, they need to be skilled in distinguishing between quality code that fully meets specifications and is easy to maintain (i.e., uses good coding conventions and documentation) and a poor program. To address this, we designed an experimental assignment in which students were asked to review, analyse and evaluate four different implementations of a dice rolling applications in the style of a 'critique session' (Level C - Guided interaction with AI) as classified in [1].

Implementation A was written by a human, meeting the specification, whilst using some unconventional practices. The other three were generated by AI. Implementations B (meeting the specification) and C (containing a bias) were generated by ChatGPT; implementation D (containing a bias) was generated by Claude. The bias in B caused inequality in chances of different sides to be rolled, while in D it was not possible to have one of the sides rolled altogether.

This assignment was given to a class of 21 first-year undergraduate Software Engineering students. The aim of this was to help students explore their analytical and critical skills regarding GenAI, similar to previous, recent studies, e.g., [3].

2 Insights

Students evaluated all four implementations, identified biases and attempted to justify why some implementations were better than others, as well as to guess which one(s) were written by a human rather than AI.

Our findings reveal that students consistently struggled to differentiate between AI-generated and human-written code, often using similar justifications for both. This suggests that students have varied expectations and understanding of what AI can generate. For instance, 'following coding conventions' and 'lack of complexity' were used to justify both human-written and AI-generated code. The majority of students felt that the human-written implementation A was unnecessarily complex, contained a series of errors and deviated from coding conventions, with over half (57%) of the cohort labelling it as AI-generated. The majority of students showed a clear preference for implementation B. It is noteworthy that this preference does not seem to be impacted by their perceptions about the code's origin.

We propose that a stronger focus needs to be put on the enhancement of students' analytical skills in the context of AI-generated content. By understanding how students analyse generated code, we aim to better equip educators with strategies to prepare learners with the skills necessary for effective, efficient, and ethical use of LLMs in software engineering.

References

[1] Joyce Mahon, Brian Mac Namee, and Brett A. Becker. 2024. Guidelines for the Evolving Role of Generative AI in Introductory Programming Based on Emerging Practice. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. ACM, New York, NY, USA, 10–16. https://doi.org/10.1145/3649217.3653602

[2] Cigdem Sengul, Rumyana Neykova, and Giuseppe Destefanis. 2024. Software engineering education in the era of conversational AI: current trends and future directions. *Frontiers in Artificial Intelligence* 7 (Aug. 2024). https://doi.org/10.3389/frai.2024.1436350

[3] Arne Styve, Outi T. Virkki, and Usman Naeem. 2024. Developing Critical Thinking Practices Interwoven with Generative AI Usage in an Introductory Programming Course. In *2024 IEEE Global Engineering Education Conference (EDUCON)*. 01–08. https://doi.org/10.1109/EDUCON60312.2024.10578746

CEP '25, January 07, 2025, Durham, United Kingdom

ACM ISBN 979-8-4007-1172-5/25/01
https://doi.org/10.1145/3702212.3702223