Student Use of AI Assistance in Software Design Tasks

Callum Curtis, Software Engineering, University of Victoria
Supervised By: Neil Ernst, PhD; Margaret-Anne Storey, PhD

Introduction

Advanced artificial intelligence (AI) code completion tools like GitHub Copilot have been found to enhance software developer satisfaction, productivity, and flow [1, 2, 3, 4]. Recent studies have focused on the performance of these tools in simple programming tasks [5, 6, 7, 8, 9]. However, the helpfulness of these tools in more complex activities, such as software design, remains largely unexplored [6, 7].

This research project aims to assess the impact of GitHub Copilot on software students' learning experience, productivity, performance, and satisfaction in completing software design tasks.

By understanding the role of GitHub Copilot in the context of students and software design, this study seeks to contribute to the growing body of research on the capacity of advanced code completion tools to aid software practitioners. Additionally, the findings can help educators make better-informed decisions about integrating such tools into software design education.

```
// Book visitor that calculates the total number of pages in the chapters
// and dedication. Does not include the title page or afterword. The main
// method creates a book with a titlepage, dedication, four chapters, and
// afterword, and demonstrates counting the correct number of pages.
public class ChapterAndDedicationPageCountVisitor implements BookVisitor {
    private int pageCount = 0;

    public void visit(TitlePage titlePage) {
        // do nothing
    }

    public void visit(Dedication dedication) {
        pageCount += dedication.getPageCount();
    }

    public void visit(Chapter chapter) {
```

Figure 1. Example GitHub Copilot code suggestion © Callum Curtis, 2023

Objectives

The research objective is to explore the consequences of software student usage of GitHub Copilot in software design tasks.

RQ1: How does GitHub Copilot impact the learning experience of software students engaged in software design?

RQ2: How does GitHub Copilot influence the actual and perceived productivity of software students engaged in software design?

RQ3: How does GitHub Copilot affect the satisfaction and well-being of software students engaged in software design?

RQ4: How does GitHub Copilot impact the characteristics and quality of software design solutions?

Methodology

Participants:

Target Group
- Students taking SENG 371: Software Evolution at the University of Victoria
- 77 students when the study was implemented in Spring 2023

Salient Characteristics
- Knowledge of software development fundamentals
- Learning design patterns as part of the SENG 371 curriculum

Experiment:

Session 1 — Intro (50 min)
- Purpose, risks, benefits, and details of voluntary participant involvement
- Instructions for setup of required software development environment
- Tutorial on the usage of GitHub Copilot
- Review of software design patterns

Session 2 — Exercises (30 min)
- Four software design pattern exercises created by the researchers for the study
- Included problem descriptions, skeleton code, constraints, and completion criteria
- Use of notes and online resources was permitted
- No communication with others was allowed
- Each participant used GitHub Copilot for a randomly assigned half of the exercises

Session 2 — Survey (5 min)
- 15 questions with an estimated time to complete of three minutes
- Examined participant perception of learning experience, productivity, performance, and satisfaction in completing the exercises
- Collected relevant participant information, including familiarity with software design patterns and GitHub Copilot

Session 3 — Reflection (50 min)
- Explored participant experience completing the exercises and using GitHub Copilot
- Semi-structured group discussion led by researchers
- Anonymous submissions from participants supported through Mentimeter

Question 4

Description

You are designing the software for a bookstore. There is a single object structure, `Book`, but the number of operations performable on instances of `Book` is likely to increase indefinitely. You are tasked with using the visitor pattern to support this requirement and implement the first visitor, which should find the total number of pages in the `chapters` and `dedication` (not `title page` or `afterword`).

Criteria

Add a main method to your visitor class that creates a book with:

- 7 page dedication
- 4 chapters
  - Chapter 1 having 675 pages
  - Chapter 2 having 98 pages
  - Chapter 3 having 842 pages
  - Chapter 4 having 524 pages
- 37 page afterword

and show that your visitor calculates the correct number of pages.

Figure 2. Example software design pattern exercise © Callum Curtis, 2023

Survey - Student Use of AI Assistance in Software Design Tasks

* 1. What is your GitHub username?

* 2. Before participating in the study activities, how familiar were you with software design patterns?
- Extremely familiar
- Very familiar
- Somewhat familiar
- Not so familiar
- Not at all familiar

* 3. Before participating in the study activities, how familiar were you with programming in Java?
- Extremely familiar
- Very familiar
- Somewhat familiar
- Not so familiar
- Not at all familiar

* 4. Before participating in the study activities, how familiar were you with GitHub Copilot?
- Extremely familiar
- Very familiar
- Somewhat familiar
- Not so familiar
- Not at all familiar

* 5. I felt more productive when using GitHub Copilot.
- Strongly agree
- Agree

Figure 3. Initial questions in research survey © Callum Curtis, 2023

Results

To ensure the voluntariness of participant consent and due to the involvement of professors in the research – who could hold positions of power over student participants – the researchers are not permitted to access the data or know who opted out of the study until after the course marks for SENG 371 have been finalized. For this reason, further work must remain pending until the completion of the Spring 2023 term.

Pending Work

While the experiment has been implemented successfully, several items remain for completion once the data is available to researchers.

Pending work includes data cleaning, data analysis, results interpretation, and creating the finalized written research submission.

Future work could include reimplementing the methodology with a different participant group or AI code completion tool.

Acknowledgements

I would like to express my sincere gratitude to Dr. Neil Ernst and Dr. Margaret-Anne Storey for their invaluable support and guidance, Zane Li for significant contributions in implementing the experiment and providing advice during project meetings, and all the student participants who made this research possible.

This research was supported by the Jamie Cassels Undergraduate Research Awards at the University of Victoria.

References

[1] P. Vaithilingam, T. Zhang, and E. L. Glassman, “Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models,” in Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, Apr. 2022, pp. 1–7. doi: 10.1145/3491101.3519665.

[2] C. Bird et al., “Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools,” Queue, vol. 20, no. 6, p. Pages 10:35-Pages 10:57, Jan. 2023, doi: 10.1145/3582083.

[3] E. Kalliamvakou, “Research: quantifying GitHub Copilot’s impact on developer productivity and happiness,” The GitHub Blog, Sep. 07, 2022. https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/ (accessed Mar. 12, 2023).

[4] “ML-Enhanced Code Completion Improves Developer Productivity,” Jul. 26, 2022. https://ai.googleblog.com/2022/07/ml-enhanced-code-completion-improves.html (accessed Mar. 11, 2023).

[5] D. Hendrycks et al., “Measuring Coding Challenge Competence With APPS,” May 2021, doi: 10.48550/arXiv.2105.09938.

[6] P. Denny, V. Kumar, and N. Giacaman, “Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language.” arXiv, Oct. 26, 2022. doi: 10.48550/arXiv.2210.15157.

[7] M. Chen et al., “Evaluating Large Language Models Trained on Code.” arXiv, Jul. 14, 2021. Accessed: Mar. 11, 2023. [Online]. Available: http://arxiv.org/abs/2107.03374

[8] J. Austin et al., “Program Synthesis with Large Language Models.” arXiv, Aug. 15, 2021. doi: 10.48550/arXiv.2108.07732.

[9] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, “The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming,” in Proceedings of the 24th Australasian Computing Education Conference, New York, NY, USA, Feb. 2022, pp. 10–19. doi: 10.1145/3511861.3511863.