2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) | 979-8-3503-6613-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VL/HCC60511.2024.00053

Effective Integration and Use of Non-Development LLMs in Software Development

Fairuz Nawer Meem
Department of Computer Science
George Mason University
Virginia, USA
fmeem@gmu.edu

***Abstract*—There is an increasing utilization of non-development large language models (LLMs), such as ChatGPT, in software development despite the availability of dedicated software development tools. I propose research that investigates the reasons behind use of non-development LLM tools and the impact they have on productivity and well-being of software practitioners, particularly those with diagnosed mental health disorders. I will use insights from these efforts to analyze the advantages and disadvantages of non-development LLMs in order to generate recommendations and interventions that will facilitate the effective integration and use of non-development LLM tools in software development.**

***Index Terms*—VL/HCC GC submission, large language models, non-development LLM, ChatGPT, productivity, well-being, mental health**

I. Introduction

Ensuring the maximum efficiency and productivity of practitioners is essential for any software development team. Nowadays, practitioners have the benefit of software development or automation tools, which has not only reduced time and effort but can increase the overall efficacy of practitioners [9]. This emphasizes the availability and use of automated tooling as an important factor for the productivity of practitioners, as well as the industry itself. However, the recent progress in tools backed by large language models (LLMs), like ChatGPT, has brought about a significant transformation in the area of software development despite not being designed as software development tools. These technologies are highly appreciated for their capacity to replicate human-like interactions and have great potential for optimizing bug fixes and regular development tasks, thus enhancing productivity of software practitioners [8].

Despite any potential benefits, prior work has found that AI-assisted tools may occasionally misinterpret specifications and produce inaccurate output [2], which leads to time consuming validation efforts [7]. Furthermore, given the already stressful nature of software development and the impacts that can have on productivity [1], an additional concern is the impact use of non-development LLM tools could be having on the mental health and well-being of practitioners. Therefore, it is essential to understand how to effectively support use of these tools in practice.

To this end, I propose research that will investigate the use of non-development LLM tools for software development tasks and how we can effectively support productive and healthy use of these tools in practice. My work will provide foundations for research and practice interested in promoting both productivity and general well-being for software practitioners.

II. Motivation and Related Work

My research began with exploring software practitioner knowledge of and experience with automated program repair (APR) tools in practice. APR tools automatically repair faults, reducing the effort and error-proneness of manual fixes [3]. Despite the benefits to use, my prior work found that practitioners are using ChatGPT more often than existing APR tools for program repair tasks [4]. This surfaces questions regarding the use of non-development LLMs for software development tasks. Prior research has proposed conversational APR support for bug fixing [5]. However, there is little to no work that has investigated *why* developers use these tools and the impact they may be having on their productivity and other relevant factors like their mental health.

III. Non-development LLMs for Program Repair

In my prior work, I administered a survey to learn about practitioners' perspectives and experiences with current APR tools and techniques, which received valid responses from 331 software practitioners. We found that practitioners may be using ChatGPT more frequently than APR tools when fixing software defects [4].

Given ChatGPT is a commonly used resource, I next wanted to better understand if and to what extent we can use non-development tools like ChatGPT that attempt to provide human-like interactions during the "automated" repair process. So next, I conducted an empirical study on developer use of ChatGPT to support defect understanding and repair. Using a pre-curated dataset [1] of developer ChatGPT logs, I reported on the kinds of bugs developers may be engaging with ChatGPT to repair and established generalized conversation workflows for successful conversations.

The common categories in this dataset were**Deployment and Configuration Bugs** and **Algorithmic Bugs**. Among 26 conversations of **Deployment and Configuration Bugs**, there were 9 conversations about *Environmental Compatibility*

[1] https://github.com/NAIST-SE/DevGPT

979-8-3503-6613-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VL/HCC60511.2024.00053

Bugs, along with 8 conversations about *Installation Bugs* and 9 conversations about *Permission Handling Bugs*. On the other hand, among the 24 **Algorithmic Bugs**, 23 conversations were about *Logical Bugs*, where only one was about *Computational Bug*. Based on the analyses, we found common conversational patterns between users and ChatGPT. For example, for **refactor requests** from users, ChatGPT usually provides *updated code*. For **clarification requests** from users, ChatGPT provides *example with code snippet*, *explanation or suggestions with or without code snippet* (sometimes followed by a *clarification request* to the users) and *updated code*. For **validation or support request** from users, ChatGPT generally provides *explanation or suggestion with code snippet* as a reference.

I also gleaned insights on context and the different ways developers interact with ChatGPT. For example, some conversations indicate direct human-alike, two-sided conversations between the user and ChatGPT. In those conversations, the user is fully involved and giving feedback or clarifying suggestions provided by ChatGPT by asking for details or cross-questioning. In others, users only ask questions and treat ChatGPT more like a tool that outputs a solution.

IV. Future Work

Despite the evidence of use in practice, little work has investigated why developers use non-development LLMs like ChatGPT in lieu of (or as a supplement to) existing developer tool support. To this end, I propose research that explores the factors contributing to the use of non-development LLMs to support program repair or other development tasks. With a better understanding of why developers use non-development LLMs, in combination with our contributions regarding when and how, we can begin to explore and provide effective tool or environment support.

While we may find that non-development LLM tools like ChatGPT can enhance productivity, there is a lack of specific understanding regarding their influence on the mental health, well-being, and professional productivity, especially for practitioners with diagnosed mental health disorders. Therefore, I aim to examine the impact of non-development LLMs on software practitioners mental health, exploring both the potential for positive and negative effects. The objective is to ensure non-development LLMs can effectively support all software practitioners. The following research questions will guide my future research:

- Why do practitioners use non-development LLM assistants, like ChatGPT, for software development tasks?
- How well do non-development LLM tools support software development tasks practitioners use them for?
- What are the effects (short term-long term and positive-negative) of AI-assisted tools on the mental health of the practitioners?
- What features of AI-assisted tools (e.g. ChatGPT) can be defined as helpful or harmful for mental health of practitioners (with and without mental health issues)?
- What considerations are necessary (individual and organizational) to ensure proper and ethical use of non-development LLM assistants in practice?

Exploring the factors motivating the use of non-development LLMs can help gain useful insights about practitioners' need for software development support. Also, one major focus of my research will be to include practitioners with mental health issues as there is little to no research focusing how use of AI-assisted tools affect them. As these tools are being widely used for development tasks as alternative to other available supports, ensuring the well-being of the practitioners facing mental health limitations while using these tools will be a primary concern for my research.

V. Conclusion

Overall, the goal of my research is to ensure proper and effective AI-assisted tools and environments for software practitioners that can facilitate increased productivity for companies while ensuring the well-being and mental health of practitioners. Our previous work amplifies the use of non-development LLMs in the field of software development. It also explores the way practitioners are using the available non-development LLMs or what might be a good general approach from users to get the best result from these technologies. However, we cannot ensure proper use of these tools without investigating the reasons behind the widespread use of the non-development LLMs for software development tasks. Exploring these reasons can give us a better understanding of the expectations or workflows of developers and help in building AI-assisted tools that support practitioners better. In better understanding the effects on practitioners with diagnosed mental health disorders, we can ensure everyone's productivity and well-being while using non-development LLMs in sotware development.

References

[1] Bubonya, Melisa, Deborah A. Cobb-Clark, and Mark Wooden. "Mental health and productivity at work: Does what you do matter?." Labour economics 46 (2017): 150-165.

[2] Kuhail, Mohammad Amin, et al. ""Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools." Science of Computer Programming 235 (2024): 103111.

[3] Parnin, Chris, and Alessandro Orso. "Are automated debugging techniques actually helping programmers?." Proceedings of the 2011 international symposium on software testing and analysis. 2011.

[4] Meem, Fairuz Nawer, Justin Smith, and Brittany Johnson. "Exploring Experiences with Automated Program Repair in Practice." Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 2024.

[5] Xia, Chunqiu Steven, and Lingming Zhang. "Conversational automated program repair." arXiv preprint arXiv:2301.13246 (2023).

[6] de Oliveira, Claire, et al. "The role of mental health on workplace productivity: a critical review of the literature." Applied health economics and health policy 21.2 (2023): 167-193.

[7] Mozannar, Hussein, et al. "Reading between the lines: Modeling user behavior and costs in AI-assisted programming." Proceedings of the CHI Conference on Human Factors in Computing Systems. 2024.

[8] Brandtzaeg, Petter Bae, and Asbjørn Følstad. "Why people use chatbots." Internet Science: 4th International Conference, INSCI 2017, Thessaloniki, Greece, November 22-24, 2017, Proceedings 4. Springer International Publishing, 2017.

[9] Guinan, Patricia J., Jay G. Cooprider, and Steve Sawyer. "The effective use of automated application development tools." IBM Systems Journal 36.1 (1997): 124-139.