EasyChair Preprint

№ 13562

Impact of AI Tools on Software Development Code Quality

Boris Martinović and Robert Rozić

June 6, 2024

Impact of AI tools on software development code quality

Boris Martinović[1][0009−0009−1491−7956] and Robert Rozić[2][0000−0001−8443−1200]

[1] Postgraduate doctoral programme, University of Mostar, Mostar, Bosnia and Herzegovina
boris.martinovic@phd.sum.ba
[2] Faculty of Science and Education, University of Mostar, Mostar, Bosnia and Herzegovina
robert.rozic@fpmoz.sum.ba

Abstract. Artificial intelligence (AI) is a powerful tool that has been widely used in various industries, including software development. In this study, we explore the perceived impact of AI tools on the quality of software development code. The study aims to provide a comprehensive understanding of the current state and potential future trends of artificial intelligence in software engineering. Through a survey conducted in various tech companies, the findings of this study aimed to provide insight into the effectiveness of AI assistance in software development, particularly focusing on code quality. The overall results show that there is high satisfaction among developers using AI tools, with more than three-quarters of them stating that the adoption of these tools positively impacted their overall satisfaction and productivity in the software development sector.

Keywords: · Artificial Intelligence · AI Tools · AI impact · Software development · Code quality

1 Introduction

The rise in popularity of artificial intelligence transformed the approach to the coding process of software development. The integration of AI in that domain has helped to emerge various AI-powered tools that have the potential to positively impact code quality. This technology is still in its early stages of adoption and exploration. The number of software developers using AI tools is increasing, as tools can offer various capabilities such as code generation, automated testing, bug detection, etc. The most common use case is for the automation of time-consuming coding tasks. AI models are trained using large data sets from the code repository, which allowed them to understand common coding patterns.

Our research focuses on a crucial aspect of this technological integration: whether AI tools contribute to an improvement in code quality from the developer's perspective. To investigate this, a survey was designed and distributed

to software developers in various tech companies. Software developers are using artificial intelligence (AI) to minimize repetitive coding tasks that consume a lot of time. This enables them to focus more on the fundamental logic and architecture of the applications they are developing. This shift in focus could lead to improvements in the overall quality of the code. Previous research related to this topic has been conducted in terms of empirical studies [18] of AI-assisted tools. In the study, it was found that the latest versions of ChatGPT, GitHub Copilot, and Amazon CodeWhisperer at the time generate the correct code 65. 2%, 46. 3% and 31. 1% of the time, respectively.

The main argument of this research suggests that code quality increases when AI-based tools are used. This is based on the assumption that AI tools help refine the structure, logic, and readability of the code while also improving other aspects such as security and overall robustness. In the following sections of this paper, we will discuss metrics and elements of code quality, such as readability, maintainability, efficiency, and accuracy.

The target audience for this research includes not only software developers, but also code testers and other professionals involved in the software development process. The findings of this study aim to provide these groups with valuable information on the current usage of AI tools in software development, the perceptions of these tools among practitioners, and potential future trends in this domain.

2 Elements of Code Quality

In this study, we examine the impact of AI tools on software development, particularly focusing on code quality. Code quality is a multidimensional concept that is crucial to the success of software projects. We identified four key components of code quality for our investigation: readability, maintainability, efficiency, and accuracy. These elements were selected based on their importance in software development and their potential to be influenced by AI tools. Our survey respondents, who included both users and nonusers of AI tools, provided insight on how these tools impact these aspects of code quality.

2.1 Readability

Code readability is a fundamental element of the quality of the code. It is defined as how easily and logically the code can be read and understood by others. The readability of a program is related to its maintainability and is a key factor in overall software quality, [15] Highly readable code is less prone to errors, is more accessible for debugging and maintenance, and promotes collaboration among team members.

Survey Question:

> To what extent do you believe AI tools contribute to improving code readability in your projects?

AI tools can improve the readability of the code by providing suggestions for clearer variable names, improving the code structure, and providing helpful comments. These tools also aid in enforcing coding standards, ensuring consistent formatting, and identifying redundant or confusing code segments. As a result, the use of AI tools can contribute to an overall improvement in code readability, positively impacting the quality of software development.

One of the studies underscores the capability of GitHub Copilot, an AI-powered tool, to produce code with readability comparable to that of human programmers. It points towards AI's role in enhancing code readability, while stressing the need for programmers to review AI-generated code to maintain quality and maintainability[13].

2.2 Maintainability

Maintainability is another crucial aspect of code quality. It can be defined as the ease with which a software system or component can be modified to be corrected, improved, or adapted to its environment[14]. Maintainability is essential for the long-term success and sustainability of software projects. It ensures that future modifications or updates can be made efficiently and without negatively impacting the overall functionality of the system.

Survey Question :

> Have you noticed changes in code maintainability since incorporating AI tools?

Clean, well-structured code generated by AI tools can enhance maintainability. AI tools have the potential to predict maintenance issues early and suggest code refactoring or improvements.

2.3 Efficiency

Code efficiency refers to the ability of software code to perform tasks quickly and effectively, utilizing minimal system resources. The efficiency of the code is directly related to the performance and speed of the software, by which the quality can be evaluated on its basis. In order to improve the code efficiency, it is necessary to remove unnecessary or redundant code.[16]

Survey Question:

> Do you think AI tools have positively influenced the overall efficiency of your coding process?

Efficiency refers to the speed and utilization of resources in coding. AI tools can significantly contribute to code efficiency by optimizing algorithms, identifying performance bottlenecks, and suggesting improvements in code structure.

2.4 Accuracy

Code accuracy is arguably the most important aspect of code quality. It refers to the correctness and precision of the code in performing its intended tasks. Achieving high code accuracy is a complex task for AI, as it involves understanding the intricate logic and requirements of the software and ensuring that the code runs flawlessly under a variety of conditions.

Survey Question

> How confident are you in the accuracy and relevance of AI-generated code suggestions?

Accuracy ensures that the code performs as intended. AI tools can play a crucial role in improving code accuracy by automating testing processes, detecting and fixing bugs, and identifying potential errors or vulnerabilities. However, some argue that the use of AI tools in software development may lead to an increased dependency on automated solutions and reduce the need for critical thinking and problem solving skills among developers.

2.5 Traditional Methods vs. AI Influence

Traditionally, code quality has been ensured through manual methods such as peer code reviews, adherence to style guides, and writing tests. AI tools are now augmenting these practices, making error detection, test writing, and code review more efficient. This integration signifies a shift in traditional methods, where AI tools not only contribute to the quality of the code but also improve the effectiveness of conventional quality assurance processes.

3 Overview of tools discussed in the survey

In this research, we wanted to find a general overview of some of the popular and diverse AI-centric tools that are present on the market at the time of writing and the overall usage of those tools. This was important because of the potential outlook on AI depending on the use of those tools. All of them were chosen for their features and differences compared to each other. Although the majority of the mentioned AI tools could be all used for performing similar tasks, we decided to divide them into two categories: general purpose and developer-specific. As its name says, general-purpose tools can be used for performing a multitude of tasks and are used by a wider population. However, developer-specific tools are related to improving software development workflows and improving them. This classification makes it easier to compare both the usage of these tools by developers and to have better insight into developer habits.

3.1 General-purpose AI tools

General purpose AI tools have uses in different spheres of life and are practically things that all people with basic computer-using skills can get their hands on. For this research, we chose four particular tools that we classified as general-purpose AI tools. They are ChatGPT, Bing Copilot, Perplexity, and DALL-E. Because of their general-purpose nature, the aforementioned tools could be used to complete developer-specific tasks, such as code generation, bug fixing, explanation of code, generating diagrams, and other different uses.

ChatGPT is arguably the most popular AI-based tool in the current market. It is a chat-like interface where users input their prompts in a text box, and users get the response as a streamed text inside the chat. This approach is making it very easy to understand and familiar to a larger population, because the context of the "chat" is preserved. By doing this, it enables users to quickly iterate on their prompts and create a "conversational" flow of informational exchange and, ultimately, more precise answers. It offers multiple models, including GPT 3.5 and GPT4. GPT 3.5 version can be used free of charge, while the GPT 4 version requires a paid subscription.[7]

Bing Copilot is a tool that is extremely similar in form to ChatGPT, but with a very distinct difference: it is built directly into the Microsoft Edge browser. Microsoft Edge browser is the default browser for Windows operating systems, which takes about 70% [4] of the whole operating system market, eclipsing the total numbers by MacOS, various GNU/Linux distributions and other desktop operating systems. This provides a good benchmark for usage compared to ChatGPT itself, since we estimate that Bing usage will be much lower compared to ChatGPT, despite being more available directly to consumers.[12]

Perplexity.ai is a new AI tool that aims to replace search engines by giving users direct answers to their questions without the need to go to their questions, among others. What differs Perplexity from ChatGPT is that it offers its sources to the data as well, making it easily verifiable as needed. Users can utilize Perplexity for multitudes of tasks, such as before mentioned answering basic questions or going deeper into topics with its Copilot search. Copilot allows follow-up questions with context, deeper answers, and more references.[5]

DALL-E is the tool that is very specialized when compared to the previous ones. Where others aim to mainly give users a way to get the data in written form, DALL-E is using GPT models to generate images from text. DALL-E, allows users to create images and art from text prompts, allowing users to combine concepts, attributes, and styles within generated images. Additionally, it can edit existing images, create variations, and understand how objects evolve over time. We included DALL-E in the survey to see how much developers are using image generation tools, since visual tools can also be an important part of the software development process.[3]

3.2 Developer-specific AI tools

Tools used for developer-specific tasks are a lot more diverse in nature compared to general-purpose tools mentioned in the section above. They offer different amounts of integration into developer workflows and can be used at various stages of development. Some of them are integrated development environments themselves, while others integrate into development environments or code editors, and some are used to generate the project initially. Tools that are covered are GitHub Copilot, Google IDX, Codeium, Wasp Mage, and Cursor.

GitHub Copilot is the the most popular AI-based tool that is specific to development workflows. It is not a standalone product, but has to be used within integrated development environment or a code editor. Currently supported editors are Visual Studio Code, Visual Studio, NeoVIM and in various Jetbrains IDEs.[6] GitHub Copilot is available with a subscription for both private users and companies. It works with different programming languages, claiming that it supports all languages that appear in public repositories. For some, like JavaScript, GitHub Copilot offers better support, since it is prominently used in public repositories. It offers support for various operations, such as explanation of code features, completion of code snippets, creation of tests with the code.[17]

Google IDX unlike GitHub Copilot is a web-based integrated development environment that runs in a user's web browser environment and executes in the cloud. It supports projects in React, Next.js, Angular, Flutter, Vue, Svelte, Go, Python, and more. [11] It allows for both web development and multi-platform development and allows users to utilize built-in AI tools that aim to enhance both speed of writing and quality of code. At the time of writing, it is in the public beta phase. [8]

Codeium is an AI-powered toolkit that supports AI code completion, search capabilities, and an AI chat function for developers. It shares some of its features with GitHub Copilot, namely auto-complete and chat features. The main difference and the reason why it is included is the fact that it has a free tier, which can make it appealing to developers who want to try out AI tools for the first time, without spending money. [1]

Wasp Mage is vastly different tool compared to the other mentioned in this study. Wasp (Web Application Specification) is a tool that aims to simplify development of full-stack applications by combining the main.wasp configuration file with the React and Node.js files code in order to output a full-stack application consisting of React/Node.js/PostgresDB. The main.wasp file contains declaration for important parts of the application, such as full-stack authentication, database schema, asynchronous jobs, full-stack type safety. Wasp Mage is primarily used to kick start the development process by creating a full-stack app based on the initial prompt explaining the purpose of the application. The resulting output of the whole process is a Wasp application. [9] [10]

Cursor is code editor that aims to integrate AI into various software development processes. It is a standalone development environment that allows users to ask for information about the codebase, automatically completing code snippets, chat, automatic code debugging, and more. Since it is a development environment for desktop use, it is useful to compare it directly with a similar cloud product, such as IDX. [2]

4 Experiment

The main way to access the data for this experiment is through a survey. Survey itself is divided into three parts: demographic information, section for participants not using AI tools, and a different one for using AI tools.

For non-users of AI tools, we looked at reasons for not adopting these technologies and what could encourage their future use. Understanding these perspectives helps contextualize the adoption barriers and potential areas for improvement in the development of AI tools.

The demographic information section will provide information about age group, country of residence, current employment status and primary business sector of the participant's business. This information is gathered to form potential relationships between responses and to determine whether there are any correlations between the participant's demographic information and the use and satisfaction with AI tools. At the end of the section, there are two questions that provide more information about the participant's outlook on AI. One asks about the fear of AI replacing the participant in terms of work or employment, and the other inquires about the frequency of usage of AI in participant's software development workflows.

The section for participants not using AI tools will be filled by participants who answered "Never" on the question that asks "How frequently, on average, do you use AI-powered tools in your software development workflows?". This section is equally important for this study, as it will create the environment where participants will share their reasons for not using AI and also things that need to be improved in AI tools in order for participants to become users of AI-powered tools.

The section for participants who use AI is the main section of this survey. It is designed to find out the ways that AI-tools affect participant's software development workflow. The questions are related to the usage of particular AI tools mentioned in Section 3 of this paper and how tools that participants use affect their software development workflow. Before going to tools that affect code quality, the goal is to find out where participants use AI tools and how they pay for it. Participants can specify whether they use AI tools for business-related projects, private projects, or both. They can also state how they finance those tools: through the company they work out, paying for licenses out of their own

pocket, or whether they only use free tools. Besides effects on the code quality, which was explained in Section 2 of this paper, there are questions regarding AI tools influencing overall process and participant's satisfaction with them. It is also important to gauge the impressions of participants about the overall impact of AI for them as developers. For that, there are questions that pertain to recommending the AI tools that participants use themselves to others and how did AI tools impact participant's software development workflows.

The means of distributing this survey was conducted through multiple channels. The snowball sampling method was used as the optimal method for this survey. It was chosen because of specific participants that were needed (software engineers and developers) and to also maximize the reach since the survey was designed to last for 7 days. The survey was posted and shared on LinkedIn, X (formerly Twitter) and personally shared with people and companies. The main target audience were people who write software, whether it is professionally or as a hobby, who have reached the adult age of 18 years. The survey itself did cover the potential for users who both use and don't use AI tools in their software development, so any distinction based on that was not needed. Also, the potential distribution of users and non-users of AI tools can be a potential interesting point for the results.

5 Results

In our survey, which lasted from February 20th to 28th 2024, we collected responses from 84 participants. Their responses revealed several key insights into the use of AI tools in software development and their impact on perceived code quality. In the following subsections, we present the data and results of the survey.

5.1 Demographics

The participants in this survey were various professionals from various countries and different age groups (Figure 1), mainly from Bosnia and Herzegovina and Croatia. (Figure 2).

5.2 Work position

Participants reported a wide range of job positions within the software development industry. Most of the respondents identified themselves as senior engineers or junior to mid-level engineers, highlighting a broad spectrum of expertise and experience levels among the users of AI tools in software development. (Figure 3) Most of the survey respondents indicated that their business primary sectors are IT and education.

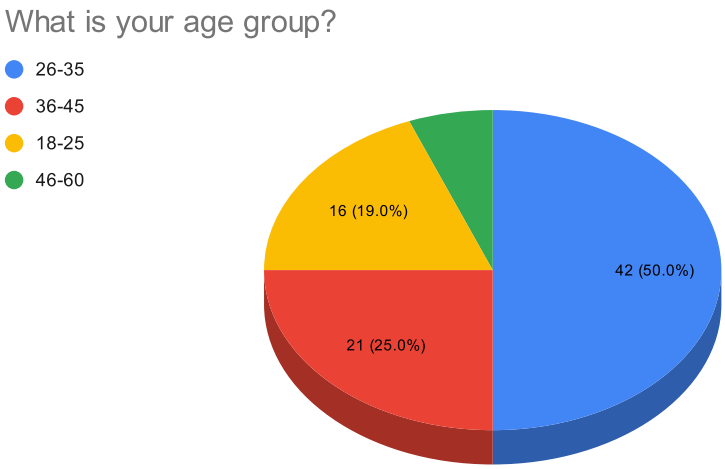

Fig. 1. Age group of respondents

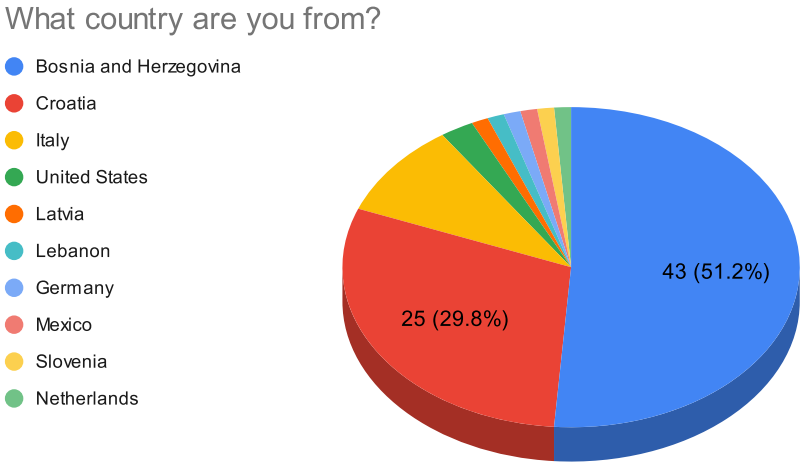

Fig. 2. Respondents country of residence

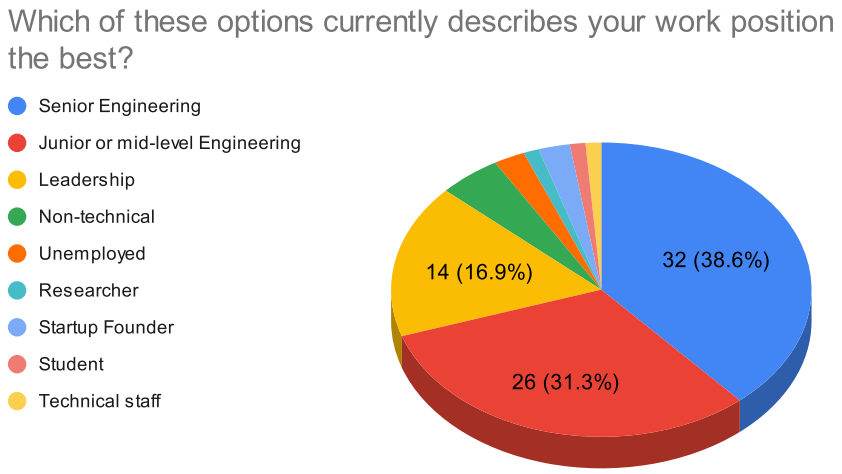

Fig. 3. Respondents work positions

5.3 Frequency of usage of AI tools

When asked about how often participants use AI tools, only 7.2% (6) study participants responded that they do not use them at all, and more than 59% (49) of them use AI tools on a daily basis. (Figure 4)

The frequency of use of AI tools indicates strong integration into daily workflows, and many participants use the AI tools multiple times per day.

Most of the survey participants expressed no fear that AI would replace their jobs in the near future, despite their daily use of these technologies. They see them as tools to enhance and speed up development process, reduce time wasted on time-consuming tasks with more focus on application architecture and fundamental logic.

5.4 Popularity of AI tools

Among the survey respondents, ChatGPT was shown to be the most used general-purpose AI tool, illustrating its widespread adoption in the software development sector. Similarly, GitHub Copilot stands out as the leading developer-specific AI tool, showing a significant trend in the industry.

Regarding payment for AI tools, the survey revealed a nearly balanced distribution, with 45% of users using paid versions, while a slight majority of 55% prefer the available free tools.

There is no clear dominant pattern in the use of AI tools for personal or business projects. 9% (7) use them only for personal purposes, 17% (13) for

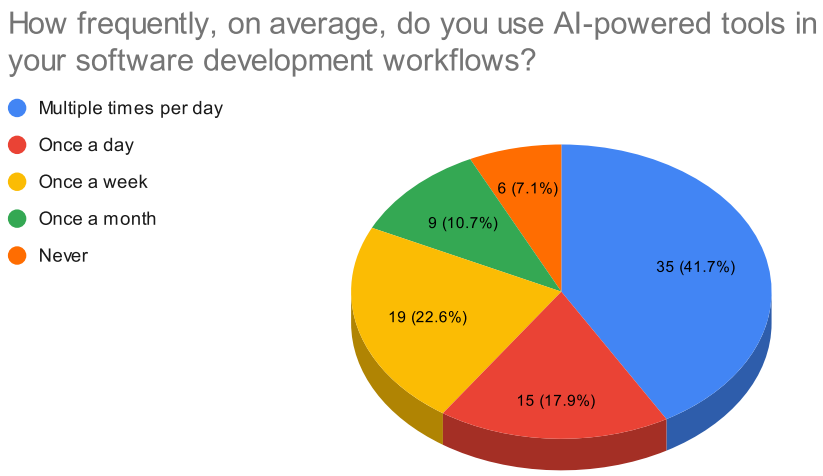

Fig. 4. AI tools frequency usage

company projects, and the other uses them for a combination of both types of projects.

5.5 Self-perceived impact on code quality

Following graphs show self-perceived impact of AI tools on the code quality elements that we used as a metric in this study: readability (Figure 5), maintainability (Figure 6), efficiency (Figure 7) and accuracy (Figure 8).

5.6 Impact on overall satisfaction

Out of 78 participants using AI tools, more than 78% (61) have reported a positive influence on overall satisfaction and productivity in software development (Figure 9). The general sentiment of the respondents is that they are willing to recommend AI tools to their peers to improve code quality.

5.7 Perspective of non-users of AI Tools

Our survey yielded a small group of 7 respondents who do not use AI tools in their software development processes. The main reasons include concerns about trust, cost, and lack of information on the benefits of AI tools. To encourage adoption among this group of respondents, improvements in code quality, user experience, and affordability of AI tools were identified as key factors.

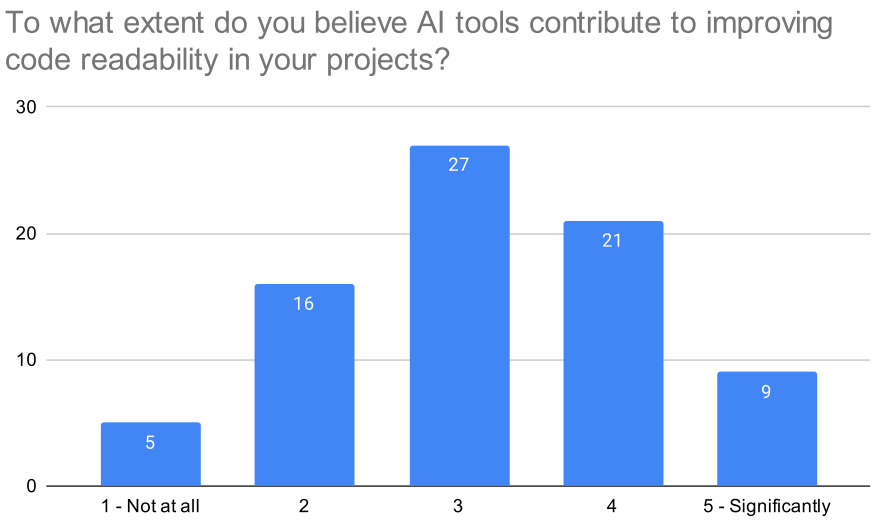

Fig. 5. Self-perceived impact on code readability

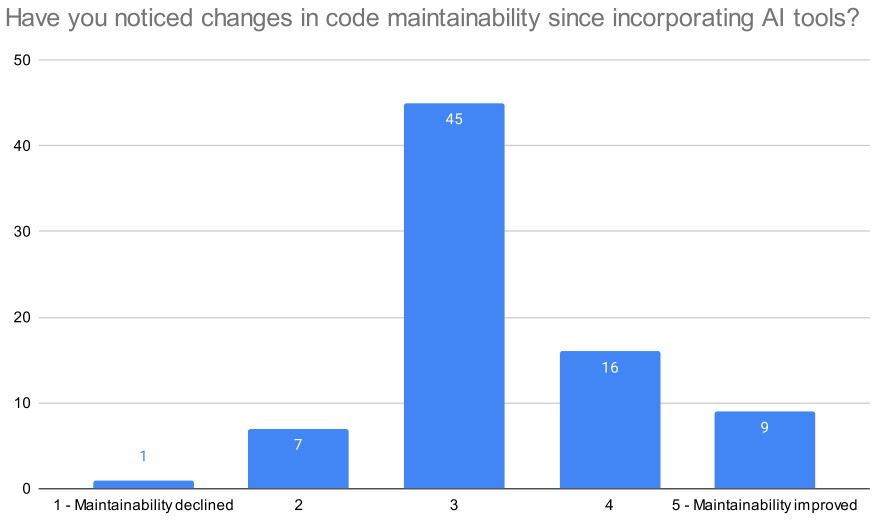

Fig. 6. Self-perceived impact on code maintainability

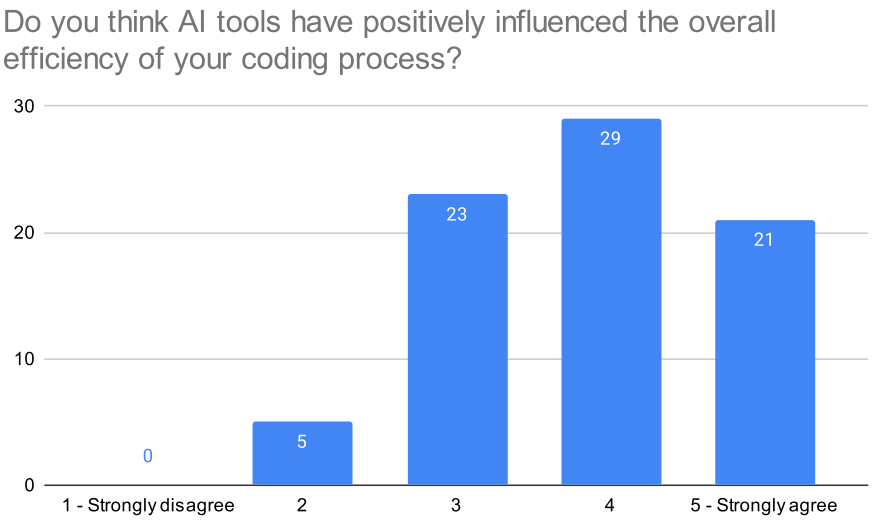

Fig. 7. Self-perceived impact on code efficiency

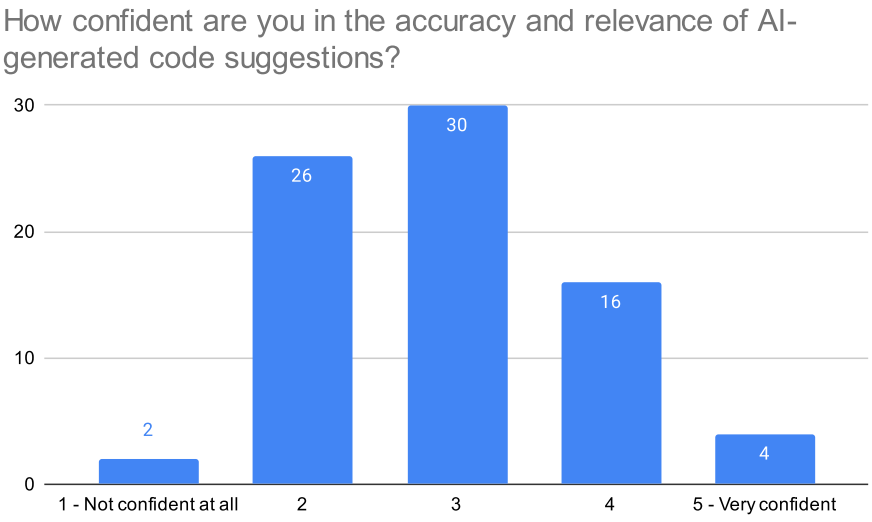

Fig. 8. Self-perceived impact on code accuracy

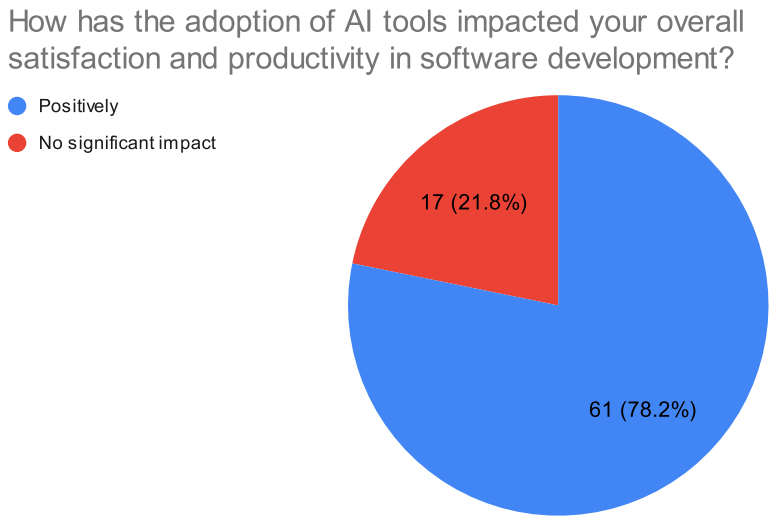

Fig. 9. Overall satisfaction

6 Conclusion and future work

The rapid rise of AI tools also brought some new and interesting ways of working for developers. At the moment, many software developers use some form of AI in their workflow, especially OpenAI's ChatGPT. The general findings among the users of AI tools in software development are very interesting. Developer-specific tools, such as GitHub Copliot, are less widely used compared to ChatGPT, meaning that there is much room for growth in terms of adoption and use. The overall results show that there is high satisfaction among developers using AI tools, with more than three-quarters of them stating that the adoption of AI tools positively impacted their overall satisfaction and productivity in software development.

On the other hand, there is a different pattern in terms of satisfaction with individual elements of code quality. Most of the answers show that the AI tools are still not providing a significant number of users with a sufficiently high level of satisfaction. The average scores on the scale of 1 to 5 in terms of satisfaction are 3.17 for readability, 3.32 for maintainability, 3.85 for efficiency and 2.92 for precision. These results show how the perceived value of all elements is not directly related to the satisfaction and willingness to recommend AI tools to others. With current data, it is visible that developers feel like AI tools are, in fact, useful to them and that they enhance the developer workflow.

When it comes to software developers who do not use AI tools in their workflow, they cite reasons such as not being able to afford paid tools and not trusting the

tools as the main reasons. From both previous research and our own, it is visible that those claims are not without merit. In the end, users decide for themselves which degree of code quality they can accept from the AI tools to be useful.

This paper is a stepping stone into further research. There are different ways of deepening the understanding of the findings made in this paper. First of all, this survey's population was mostly consisting of developers from Croatia and Bosnia and Herzegovina among developers from European Union countries and the USA. The population of research would have to be broadened to also affect other parts of the world, such as Asia, Africa, and Latin America. On the other hand, there is also a need to find deeper connections between high overall satisfaction with AI tools, among with relatively mediocre ratings of the aspects of code quality.

References

1. Codeium; Free AI Code Completion & Chat — codeium.com. https://codeium.com/, [Accessed 28-02-2024]
2. Cursor - The AI-first Code Editor — cursor.sh. https://cursor.sh/, [Accessed 28-02-2024]
3. DALL·E: Creating images from text — openai.com. https://openai.com/research/dall-e, [Accessed 28-02-2024]
4. Desktop Operating System Market Share Worldwide — Statcounter Global Stats — gs.statcounter.com. https://gs.statcounter.com/os-market-share/desktop/worldwide, [Accessed 28-02-2024]
5. Getting Started with Perplexity — blog.perplexity.ai. https://blog.perplexity.ai/getting-started, [Accessed 28-02-2024]
6. GitHub Copilot · Your AI pair programmer — github.com. https://github.com/features/copilot, [Accessed 28-02-2024]
7. Introducing ChatGPT — openai.com. https://openai.com/blog/chatgpt, [Accessed 28-02-2024]
8. Introducing Project IDX, An Experiment to Improve Full-stack, Multiplatform App Development — developers.googleblog.com. https://developers.googleblog.com/2023/08/introducing-project-idx-experiment-to-improve-full-stack-multiplatform-app-development.html?m=1, [Accessed 28-02-2024]
9. Introduction — Wasp — wasp-lang.dev. https://wasp-lang.dev/docs, [Accessed 28-02-2024]
10. MAGE GPT Web App GeneratorMageGPT — usemage.ai. https://usemage.ai/, [Accessed 28-02-2024]
11. Project IDX — idx.dev. https://idx.dev/, [Accessed 28-02-2024]
12. Your Everyday AI Companion — Microsoft Bing — microsoft.com. https://www.microsoft.com/en-us/bing, [Accessed 28-02-2024]
13. Al Madi, N.: How readable is model-generated code? examining readability and visual inspection of github copilot. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. pp. 1–5 (2022)
14. Ardito, L., Coppola, R., Barbato, L., Verga, D.: A tool-based perspective on software code maintainability metrics: a systematic literature review. Scientific Programming **2020**, 1–26 (2020)

15. Buse, R.P., Weimer, W.R.: Learning a metric for code readability. IEEE Transactions on Software Engineering **36**(4), 546–558 (2010). https://doi.org/10.1109/TSE.2009.70
16. Dhaduk, H.: Code Quality: Its Importance in Custom Software Development — simform.com. https://www.simform.com/blog/code-quality/, [Accessed 28-02-2024]
17. Scarlett, R.: 8 things you didn't know you could do with GitHub Copilot — github.blog. https://github.blog/2022-09-14-8-things-you-didnt-know-you-could-do-with-github-copilot/, [Accessed 24-02-2024]
18. Yetiştiren, B., Özsoy, I., Ayerdem, M., Tüzün, E.: Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt (2023)