Master of Science in Engineering: Software Engineering
May 2024

Large Language Models for Documentation

A Study on the Effects on Developer Productivity

Adam Alrefai
Mahmoud Alsadi

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Engineering: Software Engineering. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Adam Alrefai
E-mail: ahar19@student.bth.se

Mahmoud Alsadi
E-mail: maau19@student.bth.se

University advisor:
Prof. Niklas Lavesson
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

This thesis explores the integration of generative AI and large language models (LLMs) into software documentation processes, assessing their impact on developer productivity. The research focuses on the development of a documentation system powered by an LLM, which automates the creation and retrieval of software documentation. The study employs a controlled experiment followed by a survey involving software developers to quantify changes in productivity through various metrics such as effectiveness, velocity, and quality of documentation generated by the system.

Background: The increasing complexity of software development necessitates efficient documentation systems. Traditional methods, often manual and time-consuming, struggle to keep pace with the dynamics of software development, potentially leading to outdated and inadequate documentation.

Objectives: To investigate whether a documentation system powered by an LLM can enhance developers' productivity in software documentation tasks by assisting developers with the creation of development documentation and facilitating the retrieval of relevant information.

Method: A controlled experiment followed by a survey were conducted, wherein participants were tasked with generating and using documentation through both manual and LLM-assisted methods. The effectiveness, velocity, and quality of the documentation were measured and compared.

Results: The findings indicate that the LLM-powered documentation system significantly enhances developer productivity. Developers using the system were able to produce and comprehend documentation more quickly and accurately than those using the manual method. Furthermore, the quality of the documentation, assessed in terms of comprehensibility, completeness, and readability, was consistently higher when generated by the LLM system.

Conclusions: The integration of LLMs into software documentation processes can significantly enhance developer productivity by automating routine tasks and improving the quality of documentation. This supports software developers in maintaining current projects and also assists in the onboarding process of new team members by providing easier access to necessary documentation.

Keywords: Generative-AI, LLM, Software Documentation

Sammanfattning

Denna avhandling utforskar integrationen av generativ AI och stora språkmodeller (LLM) i processer för mjukvarudokumentation, och bedömer deras inverkan på utvecklares produktivitet. Forskningen fokuserar på utvecklingen av ett dokumentationssystem drivet av en LLM, som automatiserar skapandet och hämtningen av mjukvarudokumentation. Studien använder ett kontrollerat experiment följt av en enkät som involverar professionella mjukvaruutvecklare för att kvantifiera förändringar i produktivitet genom olika mått som effektivitet, hastighet och kvalitet på dokumentation genererad av systemet.

Bakgrund: Den ökande komplexiteten i mjukvaruutveckling kräver effektiva dokumentationssystem. Traditionella metoder, ofta manuella och tidskrävande, har svårt att hålla jämna steg med dynamiken i mjukvaruutveckling, vilket potentiellt kan leda till föråldrad och otillräcklig dokumentation.

Syfte: Att undersöka om ett dokumentationssystem drivet av en LLM kan förbättra utvecklares produktivitet i uppgifter relaterade till mjukvarudokumentation genom att assistera utvecklare med att skapa utvecklingsdokumentation och underlätta hämtningen av relevant information.

Metod: Ett kontrollerat experiment följt av en enkät genomfördes, där deltagarna hade i uppgift att generera och använda dokumentation genom både manuella och LLM-assisterade metoder. Effektiviteten, hastigheten och kvaliteten på dokumentationen mättes och jämfördes.

Resultat: Resultaten visar att dokumentationssystemet drivet av LLM väsentligen förbättrar utvecklarnas produktivitet. Utvecklare som använde systemet kunde producera och förstå dokumentation snabbare och mer exakt än de som använde den manuella metoden. Vidare var kvaliteten på dokumentationen, bedömd i termer av begriplighet, fullständighet och läsbarhet, konsekvent högre när den genererades av LLM-systemet.

Slutsatser: Integrationen av LLM i mjukvarudokumentationsprocesser kan väsentligen förbättra utvecklarnas produktivitet genom att automatisera rutinuppgifter och förbättra kvaliteten på dokumentation. Detta stöder inte bara mjukvaruutvecklare i att underhålla pågående projekt utan hjälper också till med introduktionen av nya teammedlemmar genom att ge enklare tillgång till nödvändig dokumentation.

Nyckelord: Generativ AI, LLM, mjukvarudokumentation

Acknowledgments

We begin by expressing our deepest gratitude to our families, whose unwavering support and encouragement have been our pillars of strength throughout our academic journey. Their endless patience and understanding have been fundamental to our success, and for this, we are eternally grateful.

We would also like to extend our heartfelt thanks to Andrew Thomsson, our supervisor at Telefonaktiebolaget LM Ericsson. His expert guidance and invaluable advice have been instrumental in shaping our research. His commitment to excellence and his willingness to share his profound knowledge have greatly enriched our learning experience.

Additionally, we are immensely thankful to Niklas Lavesson, our supervisor at Blekinge Institute of Technology. His academic insight and steadfast support have significantly contributed to our research and personal development. His constructive feedback and encouragement have been crucial in helping us navigate the challenges of our thesis work.

We extend our sincere appreciation to Parisa Yousefi and Björn Ringberg for adopting the idea of this project and for their pivotal role in supporting the initiative to make our collaboration possible. Their encouragement and belief in the project's potential have been invaluable.

To all mentioned, and to those unmentioned who have contributed to our journey, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Software engineering (SE) is a discipline which integrates principles of engineering to cover every stage of software production, from design to maintenance, balancing technical and managerial aspects within budget and time constraints [1]. In this context, there is an emerging need to enhance software engineering processes through the integration of cutting-edge technologies. One of which is artificial intelligence (AI), with a specific emphasis on generative AI and large language models (LLMs). Generative AI, utilizing machine learning (ML) and neural network techniques, specializes in creating new content, such as text and images, by identifying patterns within existing content represented as training data [2, 3]. In addition, LLMs are capable of processing and generating text that resembles human writing based on the learning from vast amounts of textual data [4]. An essential evolution in this domain is the attention mechanism, applied to a specific neural network architecture known as transformers [5]. This mechanism enables the model to selectively concentrate on relevant segments of the input sequence during each phase of its output generation. This approach significantly enhances the model's proficiency in managing lengthy input sequences and contextual understanding by dynamically adjusting the importance assigned to different input sections [5].

Furthermore, a generative pre-trained transformer (GPT) is a type of LLM that utilizes the transformer architecture designed to generate human-like text by predicting the next word in an output sequence given all the previous words in an input sequence. GPT models are pre-trained on a vast corpus of text, enabling such models to perform a wide range of natural language processing (NLP) tasks. In addition, multiple techniques have been explored to optimize the output of LLMs and extend their proficiency across a broader yet targeted spectrum of tasks. Fine-tuning is one of these methods, where LLMs are further trained on targeted datasets. This process aims to preserve the model's foundational abilities while refining it to accommodate more specialized applications [6]. Prompt engineering is another technique that involves crafting and refining instructions (prompts) to be used as part of the input, guiding the LLM towards a desired output [7]. The quality and specificity of these prompts play a crucial role in enhancing the model's output quality and the relevance of its answers [8]. Depending on the application and use case, multiple prompt engineering tactics can be employed to steer the output of LLMs. Among these tactics is zero-shot prompting, where the model receives instructions without any prior context or examples. This approach is effective for simple tasks, as well as general knowledge

inquiries [8]. In contrast, few-shot prompting introduces one or more input-output examples to the LLM, assisting it in understanding the task better and yielding more precise results. This method is particularly useful for tasks that need contextual understanding and specific types of output [8]. Lastly, the chain-of-thought prompting strategy entails dividing complex tasks into simpler subtasks. This tactic guides the model to follow a logical progression to arrive at a solution by transforming complex queries into multi-step tasks that can be addressed sequentially [8]. However, despite being pre-trained on huge amounts of textual data, and even with clear instructions provided as prompts, LLMs encounter challenges like outdated knowledge or lack of domain-specific knowledge [9]. These facts often lead to hallucinations as the model tries to perform predictions based solely on internal parameters [9, 10]. A technique called Retrieval Augmented Generation (RAG) overcomes these challenges by aiding the LLM with knowledge from external sources. This approach allows for continuous updating of knowledge, independent of what the model learned during the training process which enhances the accuracy and reliability of its outputs for tasks that require additional knowledge [9, 10]. The RAG process in its original form consists of three primary steps: (a) text embedding, (b) knowledge retrieval, (c) prompt augmentation and generation [9, 10].

(a) **Text embedding:** This step involves extracting text from knowledge source artifacts such as PDF documents and converting it into vector representations using an embedding model. These vectors are stored in a database, which enables subsequent similarity-based search queries.

(b) **Knowledge retrieval:** This step converts a plain user query into its vector representation using the same embedding model and then conducts a similarity search within the vector database. The retrieved results are then ranked where the most relevant documents are returned based on their similarity score.

(c) **Promt augmentation and generation:** The final step augments the search results into a comprehensive prompt, providing the LLM with all the necessary information to produce the desired output. The provided prompt usually has instructions that prohibit the model from using its own knowledge, making the model rely solely on the retrieved documents, reducing the model's hallucination.

In general, AI technologies have demonstrated considerable potential in supporting software engineering activities by automating routine tasks and facilitating data analysis [11]. Such automation empowers software practitioners to devote more time to creative and innovative tasks [12]. Furthermore, AI technologies have the potential to facilitate rapid prototyping and enhance the coding process. Specifically, generative AI and LLMs have shown remarkable promise in advancing software engineering processes [11]. The potential of generative AI spans across automating repetitive software tasks [11–13], code generation and completion [12], and assistance in debugging and test suite creation [12, 14]. Employing LLMs across various development phases not only simplifies the process but also contributes to the acceleration and improvement of software production quality [4, 14].

In the context of SE, software documentation refers to creating and maintaining documents that explain and support the software development process and the

software product itself. This includes user manuals, design specifications, technical guides, and inline comments within codebases [15]. In that manner, effective software documentation is a foundation of successful software development projects [16], playing a critical role in facilitating comprehension, maintenance, and evolution of codebases and software products [15, 17]. Furthermore, documentation not only aids developers in understanding and modifying existing code but also serves as an essential resource for onboarding new team members [18, 19]. Moreover, software documentation contributes to software sustainability [20], where well-documented software facilitates knowledge transfer, allowing new developers to easily understand and contribute to a project, thereby extending software lifespan and adaptability.

In its most fundamental form, productivity can be understood as the measurement of output relative to input. Across various disciplines, it signifies the efficiency with which resources are converted into desired outcomes. In contrast, measuring developer productivity goes beyond a simple output-to-input ratio. While this definition applies in many fields, software engineering presents unique challenges. Software outputs are often intangible, like well-designed code or intellectual contributions [21]. To get a more complete picture, we need to consider various dimensions [22]:

- **Effectivness:** This dimension reflects how well the outputs meet the project's goals and align with the desired standards. In other words, this dimension measures the correctness of the output in relation to the desired output.
- **Velocity:** This measures the speed of development, focusing on how quickly developers can achieve the desired output.
- **Quality:** This dimension goes beyond simple functionality. It considers the overall caliber of the work produced as it focus on the degree to which the software development process produces deliverables that meet predefined standards and requirements.

This thesis focuses on studying the effects of integrating generative AI and LLMs in software documentation tasks. The main goal is to create and analyze a documentation system driven by an LLM, and assess how this system influences developer productivity.

1.2 Problem Statement

The process of creating, maintaining, and retrieving high-quality documentation is highly challenging [15–17, 23, 24]. In reality, many software projects suffer from inadequate or outdated documentation [25], which can lead to increased maintenance costs, slower progress, decreased developer productivity and decreased time efficiency [26–28]. The underlying cause of these challenges is that traditional documentation methods often rely on manual efforts where developers or technical writers produce and maintain documentation artifacts [28]. Such a method struggles to keep pace with the dynamic nature of software projects' life cycle [24, 27]. Thus, software developers are frequently caught between the demands of coding and the need to maintain comprehensive documentation [26]. In contrast, automated documentation

tools such as Swagger and Javadoc are increasingly being used to mitigate these problems. However, such tools have their limitations as they usually depend on code annotations, and tend to generate specific types of documentation, such as code summaries, release notes, and code comments [28]. In addition, LLMs provide a more sophisticated approach by leveraging extensive training data to generate comprehensive and contextually rich documentation, thus addressing the limitations of previous methods.

Moreover, onboarding members in software projects often struggle to locate and comprehend relevant documentation within complex projects [29] making the onboarding process resource-intensive and time inefficient. This isn't exclusive to new members; it's a common challenge faced by existing team members in software projects who can spend a significant amount of time searching for and understanding documentation using manual methods [30, 31].

1.3 Aims And Objectives

The aim of this thesis is to study the effects of integrating generative AI and LLMs in tasks related to software documentation, focusing on developers' productivity, which includes the creation and utilization of documentation. To investigate this, we aim to conduct a controlled experiment followed by a survey to identify the effects of using an LLM-powered documentation system on these variables. The controlled experiment is designed to examine developers' velocity and effectiveness by testing the developers' ability to produce and comprehend software documentation with the help of the proposed system. Additionally, the survey is designed to evaluate the developers' perception of the quality of documentation generated by the proposed system. The primary objective of this research is to identify the impacts of this system on developers' productivity in relation to producing software documentation, finding relevant documentation, and accomplishing tasks that necessitate comprehension of existing documentation.

1.4 Scope

Based on the study's aims and objectives, we intend to implement a documentation system designed to offer the following functionalities:

- Automated generation of development documentation: This involves the system generating documentation entirely on its own, based solely on the provided code or project context, without any further instructions or clarifications.

- Semi-automated generation of development documentation: This involves a collaborative approach where the system assists in generating documentation but requires input instructions, corrections, or additional context from the developer to refine and complete the documentation.

- Summarization of documentation content.

- Providing answers to queries about software projects using existing documentation.

Software documentation encompasses various types, including component documentation, development documentation, and end-user documentation. Development documentation is created for communication among software engineers working on a specific software component [32]. Component documentation targets software engineers who use a component, typically by integrating it with other components in broader software systems [32]. End-user documentation targets users who are generally not software developers, though they may possess varying degrees of technical proficiency [32].

In our study, we intend to tailor our system to primarily focus on the generation of development documentation, while excluding component and end-user documentation. However, for the purposes of summarizing and question-answering functions, our system will consider all types of documentation.

1.5 Research Questions

To achieve the objectives and aim of this thesis, the following research question has been formulated:

- **RQ$_1$.** *How does an LLM-powered documentation system capable of generating documentation, summarizing documentation, and answering documentation-related questions affect developer productivity?*

As highlighted in Section 1.2, producing high-quality documentation presents significant challenges. Therefore, it is essential to evaluate the impact of employing a tool powered by an LLM in generating documentation on developers' proficiency in accomplishing tasks related to documentation that meet specific requirements. Furthermore, most developers find it challenging to find and understand documentation. Based on this, it is important to assess whether the system aids in simplifying comprehension and, as a result, enhance the efficiency of task completion. Moreover, the speed at which tasks are completed, whether they are development tasks involving understanding documentation or tasks directly related to documentation, is a critical factor in developer productivity. Measuring this can offer important insights into whether a documentation system powered by an LLM has the capability to enhance the developer's overall throughput. Finally, understanding developers' perspectives on the comprehensibility, completeness, and readability of LLM-generated documentation is crucial for evaluating the practical utility of such a system. Given that the primary use of development documentation is for communication among developers, gaining insights into their views on the quality of the generated documentation is essential. Based on these perspectives, **RQ$_1$.** is divided into the following subquestions:

- **RQ$_{1.1}$.** *How does an LLM-powered documentation system with the ability to generate development documentation affect the effectiveness of the developer when creating development documentation?*

- **$RQ_{1.2}$.** *How does an LLM-powered documentation system with the ability to summarize documentation and answer documentation-related questions affect the effectiveness of the developer when completing tasks that require documentation comprehension?*

- **$RQ_{1.3}$.** *How does an LLM-powered documentation system with the ability to summarize documentation and answer documentation-related questions affect the velocity of completing development-related tasks that require documentation comprehension?*

- **$RQ_{1.4}$.** *How does an LLM-powered documentation system with the ability to generate development documentation affect the velocity of creating development documentation?*

- **$RQ_{1.5}$.** *What is the software developers' assessment of the quality of documentation generated by LLM-powered system when compared to documentation generated by human developers in terms of comprehensibility, completeness, and readability?*

1.6 Outline

The structure of this thesis is organized to comprehensively explore the integration of generative AI and LLMs in enhancing software documentation and its impact on developer productivity. Chapter 1 lays the groundwork with a detailed discussion on the challenges and necessities of modern software documentation, framing the problem within the context of existing documentation deficiencies and their effects on productivity. Chapter 2 reviews relevant literature, establishing the theoretical foundation by linking previous studies to our research objectives and identifying gaps that this thesis aims to fill. Chapter 3 outlines the methodology, explaining our choice of a controlled experimental design and quantitative survey to measure the effects of an LLM-driven documentation system on various dimensions of developer productivity. Chapter 4 presents the results of the experiment and the survey, analyzing the data to draw conclusions about the effectiveness, velocity, and quality of the LLM-assisted method compared to the manual method. Chapter 5 presents discussions that synthesize our findings into a coherent narrative. Finally, the thesis concludes with Chapter 6 summarizing our work highlighting the practical implications for software development practices, and suggesting directions for future research.

Chapter 2

Related Work

This section presents a review of relevant literature and research papers that were selected through a comprehensive literature search. It focuses on presenting and discussing publications that are relevant to our research, examining their key findings to understand their implications for our study. The objective is to identify how these previous works connect to our research and to pinpoint any existing gaps in the literature, providing a foundation for our study to contribute to, and expand the current knowledge base within the software engineering domain.

Concerning the concept of employing chatbots to support software engineers in diverse software-related tasks, several publications have highlighted the potential of this technology to enhance and streamline various aspects of a software engineering role. Multiple studies have proposed and suggested the implementation of AI-driven chatbot solutions to facilitate the onboarding process of new members in software projects. The complexity and resource-intensive nature of the software onboarding process, which traditionally involves experienced developers acting as mentors is recognized by multiple studies [18, 19, 29, 33, 34]. To alleviate this, Dominic et al. in [33] propose a conversational voice assistant designed to onboard newcomers to software projects reducing the burden on experienced programmers. Capable of interacting in natural language, the proposed bot in the paper assists newcomers by providing guidance, scanning external resources like Stack Overflow for common solutions, and suggesting relevant documentation links. In a different publication, Dominic et al. suggest a similar approach to address a specific problem, namely the difficulties that newcomers encounter when joining open-source software projects [34].

A study conducted by L. K. Kivinen in [18] indicates that newcomers find documentation often poorly written and hard to comprehend. Furthermore, the study showed a strong agreement among the majority of survey respondents that AI-driven chatbots can facilitate the process of finding and comprehending related documentation.

Numerous studies have advanced the concept of conversational chatbots and assistants in software engineering practices. These studies have moved beyond proposing the implementation of such systems where they actually develop the systems and conduct the research based on these practical implementations. Abdellatif et al.'s research [35] explored the use of bots to overcome a common problem faced by many project stakeholders, namely the lack of expertise in mining and extracting useful information from their repositories effectively. The findings revealed that bots are not only useful and efficient but also accurate in providing answers to frequently asked questions, significantly surpassing manual methods in finding these answers.

Melo et al. in [36] investigated the potential of employing a chatbot assistant in software development, particularly looking into whether such an assistant could enhance the software development experience. Their research revealed a keen interest among developers in using chatbots as supportive tools in software development. It demonstrated that this technology could contribute positively by saving time and increasing efficiency. In [37], E. Aghajani designed and assessed the ADANA framework, which produces detailed code comments by leveraging similar, well-documented code snippets. The system proved the viability of generating beneficial code comments.

In contrast, the results of these studies suggest room for improvements in the capabilities of such systems, particularly in enabling deeper analysis and compensating for user errors like typos [35], and enriching the system's knowledge base with more context which can significantly increase its utility [36, 37].

However, we found that other proposed AI-driven chatbot solutions that integrate LLMs in other publications are not subject to these limitations. Ross et al. in [38] created and assessed a conversational programming assistant, powered by an LLM, aimed at supporting software engineers with code-related tasks through dialogue-based interactions. The research primarily focused on evaluating the type of interactions between the participants and the assistant, as well as the quality of the assistant's responses. Findings revealed that the assistant was capable of generating high-quality responses, including the ability to write code, explain code, respond to general programming queries, and even address broader knowledge questions. Additionally, the study found that the assistant played a significant role in enhancing developer productivity and supporting a range of coding activities. Similar results were found by Ziegler et al. in [39], which examined how GitHub Copilot affects developers' productivity. GitHub Copilot is a coding assistant built upon Codex, which is an LLM developed by OpenAI and trained on a vast corpus of open-source code [39]. A survey conducted among the participants of the study indicated that the tool enhances various aspects of productivity. For instance, it increases developer satisfaction by decreasing cognitive strain through the automation of repetitive tasks. Moreover, developers using GitHub Copilot were able to complete tasks quicker and with greater accuracy compared to those who did not have access to Copilot.

Most related to our field of study are publications that focus on utilizing different LLMs and GPT models for documentation and summarization purposes. Khan et al. investigated the application of Codex in producing code examples for documentation [40], and in the automatic generation of documentation for source code [41]. Their studies demonstrated that the approach is not only viable but also efficient. Additionally, the studies propose that this method could substantially assist in automating the generation of valuable enriched documentation. Furthermore, the utilization of the attention mechanism found in GPT models shows positive indications in multiple publications. In [42], Wang et al. focused on creating and assessing an AI tool that integrates the attention mechanism intended to improve the documentation process in data science computational notebooks. The findings revealed a notable reduction in the time needed for creating documentation and that the quality of documentation generated by the system was comparable with that produced by the data scientists themselves. Similarly, Haque et al. in [43] utilized the attention mechanism to create a context-aware model aimed at enhancing the automatic summarization of subroutines in software projects. The findings of the study indicated that the use of

the attention mechanism and thereby the inclusion of file context markedly enhances the quality of automatic subroutine summarizing, particularly when compared to baseline methods.

In [44], Sovrano et al. conducted a research on the feasibility of incorporating ChatGPT to design a system capable of converting static text and documentation into more interactive and intelligent forms. Their study revealed that this system can effectively transform conventional explanatory materials, such as textbooks and user manuals, into more dynamic and interactive formats. Such a transformation has the potential to significantly improve the learning and comprehension experience for users, rendering complex information more approachable and simpler to understand.

Research gap: Based on our literature review, numerous studies cover the integration of AI technologies including generative AI in different software development practices. While these technologies have been investigated for software documentation purposes with a focus on the feasibility of these technologies, we found that the body of literature still lacks a concrete examination of how these technologies affect developer productivity. Furthermore, we found that the research on integrating LLMs for documentation purposes mainly focuses on the output quality and performance of these sophisticated models. While these factors are important and might lead to increased productivity among developers, we found that there is still a lack of evidence and concrete conclusions that could justify and motivate the industry to integrate these technologies for documentation solutions. To this extent, our research can significantly advance the current understanding by exploring the impact of incorporating LLMs in documentation solutions on developer productivity. This study also lays the groundwork for future research inquiries, such as how software engineers adapt to using LLMs in their daily workflows and what learning curve is associated with these tools. From an industrial standpoint, our finding can demonstrate the practical potential of LLMs to enhance software engineers' workflows and reduce documentation costs, thereby encouraging the industry to adopt this cutting-edge technology and highlight the possible return of investment.

Chapter 3

Method

3.1 Research Method Selection

To address the research questions proposed by this thesis, we have selected two research methods, namely controlled experiment and survey. The survey method is chosen to address $\mathbf{RQ_{1.5}}$, while the controlled experiment method is chosen to address the remaining subquestions under $\mathbf{RQ_1}$.

3.1.1 Controlled experiment

A controlled experiment is a structured research method designed to examine a testable hypothesis by identifying cause-effect relationships between independent and dependent variables. This research method involves applying various combinations of independent variables (treatments) to at least two groups, namely an experimental group and a control group. The experimental group receives the treatment under investigation, whereas the control group does not receive the treatment while being kept under the same conditions. This ensures that any differences observed between the groups are due to the treatment itself rather than other variables [45].

We have opted for a controlled experiment approach to address all subquestions under $\mathbf{RQ_1}$ excluding $\mathbf{RQ_{1.5}}$. These subquestions examine the differences in developers' productivity in terms of velocity and effectiveness between the usage of two documentation methods: One where developers manually write the documentation and another where the documentation is fully or partially generated by an LLM. Conducting a controlled experiment enables precise measurements of performance metrics such as time, accuracy, and velocity. Furthermore, this setup allows for direct comparison between groups using software tools, thereby providing clear evidence of any productivity enhancement linked to the usage of an LMM-powered documentation system. Additionally, a controlled experiment creates standardized tasks that can be uniformly applied across all participants, allowing for reliable and measurable outcomes. Moreover, controlled experiments ensure replicability under similar conditions, a critical aspect for validating findings in our scientific research. To this extent, we have carefully designed and implemented software tools for collecting the required data, including the time taken to complete all types of tasks undertaken during the experiment.

While other methods such as surveys and interviews were considered, they were deemed less suitable due to their reliance on subjective individual perceptions and their inadequacy in accurately measuring productivity metrics such as effectiveness

and velocity. Furthermore, we also considered performing a literature review but determined that it is less appropriate due to the scarcity of relevant studies to support conclusions in our research.

3.1.2 Quantitative Survey

The quantitative survey is a systematic research method for gathering information from a specific group of people (sample) to understand the characteristics of a larger population. Quantitative surveys often rely on questionnaires or interviews that present participants with questions designed to elicit their opinions, beliefs, or experiences [46]. The quantitative survey's answers are typically based on individual judgments and perceptions, which makes it a perfect fit for $\mathbf{RQ_{1.5}}$ as it seeks subjective evaluations of individual perceptions of comprehensibility, completeness, and readability of documentation. To this extent, a survey questionnaire is chosen as a data collection method where a close-ended question is crafted to capture developers' perceptions of the documentation quality attributes.

We considered using alternative methods for this question, such as incorporating the question into the experiment. However, we determined that an experimental approach might not effectively capture subjective evaluations and the wide range of developers' opinions. Additionally, we considered using interviews, but this method is time-consuming and less efficient for collecting data from a large number of participants.

3.2 Hypotheses

In this part, we present the hypotheses for our statistical analysis based on the study's research questions in 1.5. The aim is to formally state the hypotheses and define the metrics and measurement scale required for their evaluations. The objective of the study is to empirically test if the use of an LLM-powered documentation system can enhance developers' productivity. In consequence, the following hypotheses are formulated.

1. **Hypothesis related to developer's productivity for documentation creation ($\mathbf{RQ_{1.1}}$, $\mathbf{RQ_{1.4}}$)**

   - Null hypothesis, $H_{0.Gen}$: There is no significant improvement in developer productivity when using an LLM-powered documentation system for producing development documentation compared to manual documentation methods, in which documentations are manually written by developers.

     - $H_{0.Gen.a}$:VEL_{Gen}(LLM assisted method) $\leq VEL_{Gen}$(manual method)
     - $H_{0.Gen.b}$:EFF_{Gen}(LLM assisted method) $\leq EFF_{Gen}$(manual method)

   - Alternative hypothesis, $H_{1.Gen}$:

     - $H_{1.Gen.a}$:VEL_{Gen}(LLM assisted method) $> VEL_{Gen}$(manual method)
     - $H_{1.Gen.b}$:EFF_{Gen}(LLM assisted method) $> EFF_{Gen}$(manual method)

- Required measures: Developer's effectiveness, and developer's velocity.

2. **Hypothesis related to developer's productivity for documentation comprehension ($\mathbf{RQ_{1.2}}$, $\mathbf{RQ_{1.3}}$, $\mathbf{RQ_{1.5}}$)**

   - Null hypothesis, $H_{0,Comp}$: There is no significant improvement in developer productivity when using an LLM-powered documentation system for documentation comprehension compared to manual methods, in which developers manually search for relevant information by reading static documentation artifacts (PDF, HTML, Markdown, etc).

     - $H_{0,Comp,a}$:VEL_{Comp}(LLM assisted method) $\leq VEL_{Comp}$(manual method)
     - $H_{0,Comp,b}$:EFF_{Comp}(LLM assisted method) $\leq EFF_{Comp}$(manual method)
     - $H_{0,Comp,c}$:$QLTY_{Comp}$(LLM assisted method) $\leq QLTY_{Comp}$(manual method)

   - Alternative hypothesis, $H_{1,Comp}$:

     - $H_{1,Comp,a}$:VEL_{Comp}(LLM assisted method) $> VEL_{Comp}$(manual method)
     - $H_{1,Comp,b}$:EFF_{Comp}(LLM assisted method) $> EFF_{Comp}$(manual method)
     - $H_{1,Comp,c}$:$QLTY_{Comp}$(LLM assisted method) $> QLTY_{Comp}$(manual method)

   - Required measures: Developer's effectiveness, developer's velocity, and developers' assessment of documentation quality.

In consequence, the following data is required to be gathered to test these hypotheses:

- Developers' effectiveness is quantified by the correctness of completed tasks by the developers (ratio scale).
- Developers' velocity is measured by the number of correctly completed tasks within a time frame (ratio scale).
- Documentation quality assessment is determined using Likert scale, where grades are given based on the following quality aspects: (a) comprehensibility, (b) completeness, and (c) readability.

3.3 Research Design

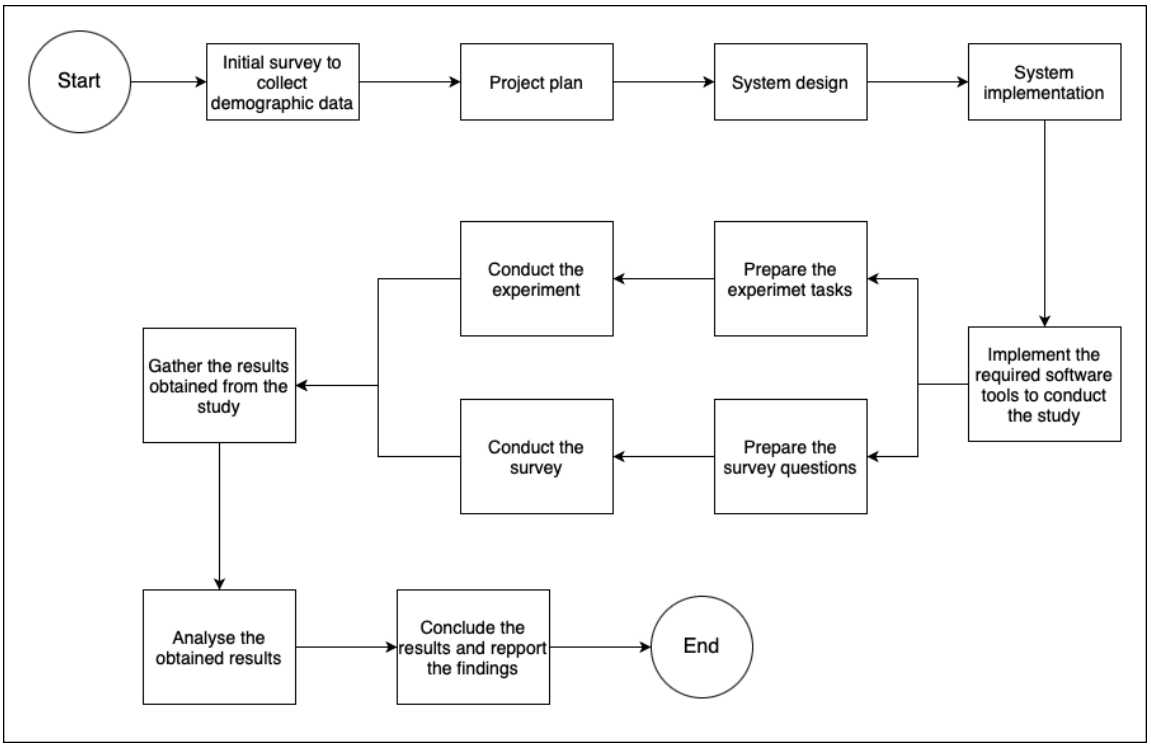
The study is conducted in an industrial environment at Telefonaktiebolaget LM Ericsson, with professional software developers as participants. This setting is chosen over academic or simulated environments to establish results that are directly applicable to real-world software development scenarios. This context supports the primary goal of evaluating the impact of an LLM-powered documentation system on the developers' productivity.

An overview of the research design is shown in Figure 3.1. As depicted in Figure 3.1, the research begins with an initial questionnaire designed to collect demographic data about the developer population at Telefonaktiebolaget LM Ericsson. Additionally, the research design encompasses preparatory activities for both the controlled

experiment and the survey. Lastly, the system implementation is followed by an implementation of custom software tools necessary for gathering the required measurements.

Figure 3.1.

Research design



Note. This figure outlines a roadmap of the study, depicting the various stages from the beginning to the conclusion.

3.4 Experiment Design

In this study, the controlled experiment is used to understand and gather data about the impact of using the LLM-powered documentation system on different dimensions of developer productivity. Specifically, the experiment is designed to evaluate the developers' effectiveness and velocity in a controlled environment. These dimensions are taken from the literature and chosen to assess the developers' productivity from these aspects.

3.4.1 Variables

The focus of our study involves four objects: documentation artifacts, source code, coding tasks, and documentation tasks. Documentation artifacts refer to the documentation that developers need to comprehend in order to complete related coding tasks. Additionally, the source code serves as a basis for requesting developers to

apply one of the documentation methods in order to complete related documentation tasks. The variables included in the study are summarized in Table 3.1.

Table 3.1

Summary of Experiment Variables

	Name	Values	Description
Independent variables	Documentation Method (DM)	{LLM, M}	One of two documentation methods is applied by each subject: LLM-assisted method or Manual method.
Controlled variables	Experience	Integer	The developers' experience is measured on a ratio scale and used to categorize developers into different groups.
	Complexity	Ordinal	The complexity of coding and documentation tasks are standardized across all participants.
	Environment	Nominal	The set of tools will be identical for all participants to perform the required tasks in the experiment.
Dependent variables	Time	Integer	The time taken by each participant to finish a task is recorded for all subjects, with seconds as the unit of measurement (ratio scale).
	Effectiveness	Integer	The number of correctly solved tasks is recorded for all subjects.
	Velocity	Integer	The number of correctly solved tasks per a predefined time interval is recorded for all subjects.

Note: This table lists and summarizes all the variables related to the controlled experiment.
DM. Documentation Method
LLM. LLM-assisted method
M. Manual method

3.4.2 Design and Operation

In this study, developer experience refers to the general programming experience of the participants, including their overall years of experience in software development across various programming languages, environments, and tools. It is not limited

to their tenure at Telefonaktiebolaget LM Ericsson or proficiency with a specific programming language or software development tools.

Furthermore, the tasks used in the controlled experiment were selected by the research team. The tasks were chosen to represent common documentation and coding challenges in professional software development environments, ensuring their relevance and appropriateness for evaluating the impact of the LLM-powered documentation system on developer productivity.

To evaluate the study's hypotheses, an experiment of one factor with two treatment design is used. The factor being tested is the documentation method used to complete the experiment tasks, with the two treatments being either an LLM-assisted method or a manual method.

Participants for the experiment were selected based on their availability within the company, Telefonaktiebolaget LM Ericsson. This non-probabilistic convenience sampling approach was chosen due to the time constraints and the unavailability of the participants within the limited time of conducting the study. To ensure that the results of the experiment are influenced by the treatment rather than differences in developers' work experience, a systematic blocking design is used. This design aims to include a sufficient number of participants in each block while minimizing the variation in work experience among them. To this end, we carried out an initial questionnaire asking developers at Telefonaktiebolaget LM Ericsson about their work experience and willingness to participate in our study. The results of the conducted questionnaire are presented in figures A.1 to A.2 in the appendix, as a supplement to our experiment. Based on the questionnaire responses, we formed six distinct experiment blocks presented in Table 3.2.

Table 3.2

Experimental block design.

Experiment block	Description
$Experience \leq 5$	Developers with less than or equal to 5 years of experience.
$10 < Experience \leq 15$	Developers with 6 to 10 years of experience.
$15 < Experience \leq 20$	Developers with 16 to 20 years of experience.
$20 < Experience \leq 25$	Developers with 21 to 25 years of experience.
$25 < Experience \leq 30$	Developers with 26 to 30 years of experience.
$Experience > 30$	Developers with over 30 years of experience.

Note. This table presents the formed experiment blocks that group participants into different groups based on their work experience.

The experiment's questionnaire initially requires the participants to grade their work experience. Based on their self-assessment, they are grouped into one of the above-mentioned groups.

After assigning participants to their respective blocks, each participant is placed into a treatment group based on the number of participants in each treatment group. If the participation level for one treatment group is higher than the other within each block, the subsequent participation is assigned to the less populated group. This approach is done to achieve a more balanced distribution of participants across treatment groups within each block. Furthermore, each participant carries out a combination of randomly selected tasks from a standardized task pool shared across all groups. This methodology ensures that the tasks are equally distributed and standardized across all groups, thus minimizing the likelihood of any uncontrolled variable.

The experiment involves participants' engagement in two distinct phases, performed in the following order:

1. Coding tasks: Within a limited time frame, participants are guided to complete a set of coding assignments that require a solid understanding of relevant documentation. This phase focuses on assessing the participants' productivity in terms of both velocity and effectiveness for documentation comprehension.

2. Documentation tasks: Participants are guided to document a collection of code snippets within a specified time limit. This phase aims to evaluate both the velocity and effectiveness dimensions for documentation production.

Furthermore, to ensure that the participants in the control group perform all experiment tasks without using the developed LLM-powered documentation system, a software component called the study component is designed and implemented as part of the system. The study component manages both the execution of the study and data collection. In addition, the component limits access to the LLM system features for developers assigned to the control group, ensuring they perform the tasks manually.

3.4.3 Data Gathering

To enhance data collection and prevent data loss, we have utilized the study component for data collection and study execution.

Velocity
The velocity dimension is determined by the number of tasks that are completed accurately, divided by the time taken to complete them. During the experiment, the velocity of developers is measured by recording the time taken to complete each task, including documentation and coding tasks. The coding solutions and produced documentation of participants are persisted by the study component. Both the time measures and produced artifacts are stored locally in the file system and then pushed to a database with the help of the study component to prevent data loss.

Effectiveness
The effectiveness dimension is measured by the number of tasks correctly completed, encompassing both coding and documentation tasks. The answers from all participants are collected by the study component and uploaded to a database for further

analysis. Each programming task comprises smaller generic subtasks, each contributing to the overall correctness of the implementation. Researchers manually evaluate participants' solutions by assessing the correctness of each subtask to determine the overall correctness score for each completed task. The overall effectiveness score is thus measured as the number of completed subtasks divided by the total number of subtasks. similarly, the correctness of the submitted solutions for the documentation tasks are manually evaluated by the researchers. The evaluation is based on whether the provided solution accurately describes the main functionalities of the associated code snippet.

3.4.4 Data Analysis

The purpose of this controlled experiment is to determine if the use of an LLM-powered documentation system, significantly enhances both the effectiveness and velocity dimensions of developer productivity. The experiment's design involves a single factor with two treatments and the measures for the hypotheses are on a ratio scale. Initially, the Shapiro-Wilk test will be used to assess the data's normality to identify the appropriate statistical test. In the case of normally distributed data, a parametric test, specifically a one-tailed t-test, will be used to compare the means of the two independent groups. However, if the data deviates from normality, the Mann-Whitney test will serve as an alternative to the t-test. This approach aims to determine whether the use of the LLM-assisted method results in a statistically significant improvement.

3.5 Suvery Questionnaire Design

The survey questionnaire entails a series of sequential steps, which aim to ensure the reliability and validity of the data collected. The steps involve the identification of objectives, identification of the population and sampling strategy, survey design, survey validation, data collection, and data analysis. These steps adhere to the guidelines provided by Kitchenham et al. in [46].

3.5.1 Survery Objectives

The survey objective is to collect data and evaluate the effects of utilizing the LLM-powered documentation system on the quality dimension of developers' productivity. The primary objective of this survey is to accumulate the necessary data to perform statistical analysis for the formulated hypotheses, with particular emphasis on the quality dimension of productivity ($RQ_{1.5}$). As emphasized in 3.1.2, quality is a subjective metric, making it essential to acquire developers' opinions concerning the quality aspects of the produced documentation. In particular, the survey questionnaire focuses on assessing the comprehensibility, completeness, and readability aspects of the documentation's quality.

3.5.2 Population Selection and Sampling Strategy

Software developers have been chosen as the target population for this survey. Given their frequent engagement with the creation and evaluation of documentation artifacts, they possess a solid base for evaluating the quality of these artifacts. Specifically, their insights are invaluable in determining the readability, completeness, and comprehensibility of documentation artifacts from a developer's perspective.

The time limitations to conduct the study led to the adoption of a a non-probabilistic convenience sampling strategy. This approach is practical when dealing with a highly specific target population with limited availability. The convenience sampling strategy was conducted in this study by sending an invitation email to all software developers at Telefonaktiebolaget LM Ericsson located in Karlskron, Sweden who were available and willing to participate in the study.

3.5.3 Survey Design

The survey adopts a longitudinal design and encompasses a questionnaire with a single close-ended question, which is further subdivided into five subquestions. Participants are requested to evaluate the quality of documentation across three aspects for five different pairs of documentation, which include both human-generated and LLM-generated documents. The respondents provide their answers to the close-ended questions by choosing from the options provided on a Likert scale. This method yields ordinal scale data. The Likert scale is a widely used approach for measuring attitudes, opinions, or perceptions, offering a simple, yet effective way to quantify subjective data. It allows respondents to express the degree of their agreement or disagreement with a given statement, providing more nuanced data than binary (yes/no) responses [47]. In the context of our survey, the Likert scale is ideal for capturing the subjective evaluations of developers regarding the quality aspects of the documentation (comprehensibility, completeness, and readability).

The survey questionnaire incorporates the same study component used in the controlled experiment. Our study design requires participants to first complete the experiment and then proceed to answer the survey questionnaire.

3.5.4 Survey Validation

The survey questionnaire has been carefully crafted by the research team. In addition the questionnaire was refined based on the feedback from the research supervisors to ensure it comprehensively addresses the survey's objective. The survey question is formulated with simple wording to ensure clarity and avoid ambiguity. The questionnaire includes one primary question:

- For each quality aspect (comprehensibility, completeness, and readability), assign a score from 1 to 5, with 1 indicating poor quality and 5 indicating excellent quality in each respective aspect.

Participants in the survey evaluate the quality of documentation based on its comprehensibility, completeness, and readability. This evaluation is done for five pairs of documentation artifacts, each consisting of a document written by a developer

and a document generated by the LLM-powered system. Both documents in each pair describe the same code. Additionally, the developer-written documentation adheres to the same structure provided in the prompt to the LLM to ensure that the structure does not influence the participants' perception of the documentation content. Furthermore, participants are unaware of whether the artifacts were created by a developer or generated by the LLM-powered documentation system. The study component randomizes the order of the artifacts, placing the LLM-generated content on either the right or left side, with the developer's content on the opposite side.

3.5.5 Data Gathering

The survey uses the same study component used in the controlled experiment, which facilitates survey participation and collects responses from participants. Upon submission by participants, their assessments of all quality aspects are uploaded to a database to prevent data loss.

3.5.6 Data Analysis

In this study, we analyze survey responses in which participants evaluate three aspects of documentation quality: comprehensibility, completeness, and readability on a Likert scale ranging from 1 to 5. Given the ordinal nature of this scale, we employ the Mann-Whitney statistical test to examine the hypotheses. The Mann-Whitney test serves as a non-parametric substitute for the t-test when comparing means between two independent samples. In our analysis, we will use it to compare the average total quality score between the two samples (developer-written and LLM-generated documentation).

3.6 System Description

This section offers a brief overview of the design and implementation aspects of the documentation system created for the sake of this study. The system mainly features two distinct functionalities: (a) generating development documentation based on code repositories and (b) responding to documentation-related queries expressed by developers in natural language. Furthermore, the system was specifically designed and implemented for use by the personnel at Telefonaktiebolaget LM Ericsson. Consequently, many design decisions were tailored to meet the needs and enhance the usability for Telefonaktiebolaget LM Ericsson' developers.

3.6.1 Architectural Overview

The system adopts a modular design, utilizing a microservice architecture to support separation of concerns, maintainability, and adaptability. Figure B.1 in the appendix section, provides an overview of the system's architecture, illustrating all its microservices.

3.6.2 System Components

In this section, we provide a brief description of each system microservice, detailing its responsibilities and how it interacts with other components.

- **frontend**: A web server responsible for serving the system's front end.
- **auth**: This component is responsible for authenticating the users and authorizing operations in the system. The microservice uses the authDB database which stores user credentials.
- **chatbot**: A component that acts as an abstraction layer between the system front end and the core microservice. The component contains logic related to the question-answering capability of the system, with a main responsibility to gather chat history and forward it with the current question to the core component.
- **chatHistory**: A component responsible for retrieving and saving user queries and system responses from the chatHistoryDB database.
- **documentationDecoder**: A component responsible for converting documentation documents into Markdown format.
- **knowledgeBase**: This component acts as an abstraction layer between the system frontend and the embedding service. It receives embedding requests from the frontend component and forwards them to the embedding component. The component is also responsible for performing some operations against the vectorStore database such as querying the available projects saved in the database.
- **embedding**: This component contains the embedding model responsible for converting plain text into vector representations.
- **core**: This component is the heart of the system, it contains logic for generating answers for user queries and for generating development documentation. The component formats all incoming requests and augments the requests in proper prompts that are sent to the LLM for processing.

3.6.3 The Used LLM

Operating an LLM requires significant resources and is quite expensive, which limits our ability to experiment with various LLMs. For this project, we utilize an LLM provided by Telefonaktiebolaget LM Ericsson through an API endpoint. Specifically, we use the mistralai/Mixtral-8x7B-Instruct-v0.1 model, which has a context length of 4,000 tokens. Details about the parameter configuration for this LLM are presented in Figure B.4.

3.6.4 Embedding

The system utilizes the RAG method to generate responses to documentation-related queries. The retrieval aspect of this method involves accessing a vectorized database containing stored knowledge, which is retrieved as vector representations and subsequently converted into text format. To enable this functionality, the system requires an embedding model capable of translating raw text into vector representations. To fulfill this need, we incorporated an embedding component housing an open-source embedding model called nomic-ai/nomic-embed-text-v1. This component serves two primary purposes: embedding of existing documentation for storage in the vector database to serve as a knowledge base, and embedding of user queries for similarity searches against the stored documents.

3.6.5 Documentation Generator

To enable the generation of development documentation from code bases, we choose to implement this part of the system as an extension for Visual Studio Code (VSCode). This decision was influenced by VSCode's popularity among developers, as evidenced by a Stack Overflow survey where over 73% of respondents indicated it as their preferred integrated development environment (IDE) [48].

The documentation generation process begins when the user selects and stages code snippets for documentation within the IDE (see Figure B.2 in the appendix section). These snippets are then displayed in a dedicated window (see Figure B.3 in the appendix section), allowing the user to review what code has been prepared for documentation. Optionally, the user can provide additional instructions in natural language for individual snippets or the entire documentation project, enhancing the context for the LLM. This step is optional, enabling both automatic and semi-automatic documentation generation. Once satisfied, the developer can launch the generation process with a single click, triggering a request to the core component to start creating the documentation.

The request sent to the core component includes details about each code snippet, such as the file name, the file path, and the programming language of the file. This, along with the code content and any user-provided instructions, is augmented in a comprehensive prompt that instructs the LLM in generating the development documentation. To ensure precise and practical outcomes, we employ two prompting strategies: (a) few-shot and (b) chain-of-thought. The prompts thus include one or more examples of how the documentation should be structured. In addition, the instructions are provided as smaller, generic instructions that the LLM can process in isolation. Once processing is finished, the generated documentation is displayed to the user, who can then edit, regenerate, or save the documentation locally.

3.6.6 Documentation Assistant

The question-answering and documentation summarization capabilities have been implemented in the form of a web-based chatbot interface. Developers initiate interactions by selecting a project from a menu that lists all projects integrated into the system. Once a project is chosen, the developer provides a documentation-related

question in natural language through the chatbot interface. The developer's query is then sent to the chatbot component that first checks for any chat history based on the current user session. If any exists, the chat history, together with the developer's query is sent to the core component for processing. When the core component receives the request, it starts by converting the query to a vector representation using the embedding component. As part of the RAG process, the embedding result is used to conduct a context-aware and semantic similarity search within the vector store which contains existing documentation related to the project selected by the developer. Together with the chat history and the results obtained from the search process, the developer query is augmented in a detailed prompt that includes all information necessary for the LLM to be able to generate an answer to the question. The prompt includes instructions that force the LLM to solely rely on the provided data, preventing it from generating uninformed or inaccurate responses. For more precise answers, a few-shot prompting strategy is used where the prompt also includes an example demonstrating how the LLM should respond to queries and use the contextual information from the search results and previous chat.

Chapter 4

Results and Analysis

4.1 Results

In this section, we provide the results and data from the controlled experiment and survey we conducted. At this point, the data is presented in its raw format, subsequent analysis of the data will be discussed in the Analysis section.

Experiment results:
The purpose of the conducted controlled experiment was to address all subquestions related to $\mathbf{RQ_1}$, excluding $\mathbf{RQ_{1.5}}$. This experiment investigates how developers' productivity varies between two documentation methods: one where developers write the documentation manually, and another where it is fully or partially generated by an LLM. Specifically, the experiment seeks to examine how these methods impact the velocity and effectiveness of developers' productivity.

Tables C.1 and C.2 (see Appendix C) depict the results of participants' submitted solutions for the programming and documentation subtasks, respectively. The tables depict the number of subtasks successfully completed by participants, alongside the total number of subtasks assigned, as well as the overall time spent on completing the tasks. The results are sorted in ascending order based on the experience of the participants, and grouped to follow the experimental block design, where participants that belong to the same block are grouped together.

Derived from the results in Tables C.1 and C.2, the participants' effectiveness and velocity results for programming and documentation tasks were calculated and are presented in Tables C.3 and C.4 in the appendix, respectively. The following formula was used to perform the effectiveness calculations:

$$\textit{Effectiveness} = \textit{Number of Correctly Completed Tasks} \;/\; \textit{Total Number of Tasks}$$

Whereas the velocity was calculated in terms of the developers' throughput in one hour using the following formula:

$$\textit{Velocity} = (\textit{Number of Correctly Completed Tasks} \;/\; \textit{Time}) \cdot 3600$$

Survey results:
The conducted survey aimed to gather developers' perceptions of documentation quality, comparing documentation artifacts written manually by developers to those generated by an LLM, as specified in $\mathbf{RQ_{1.5}}$. The evaluation focused on assessing the readability, comprehensibility, and completeness of the documentation. The participants' quality evaluations are presented in Table C.5 (see Appendix C).

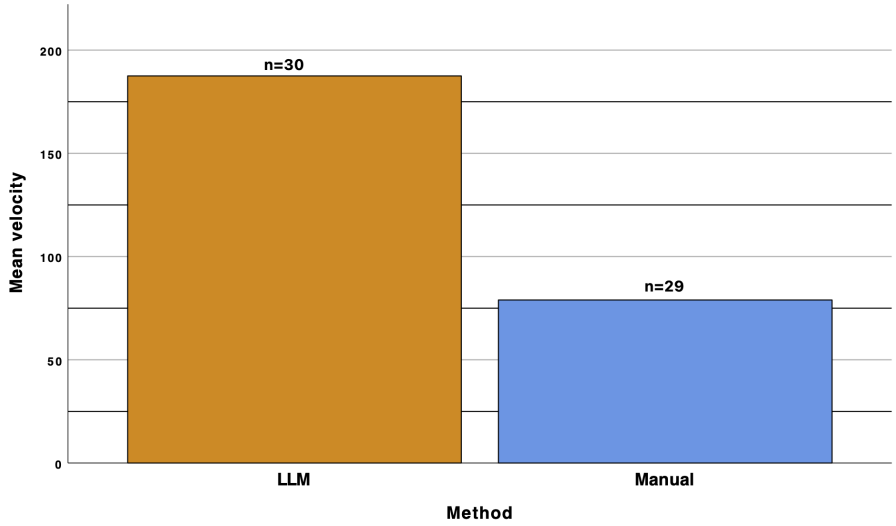
4.2 Analysis

4.2.1 Descriptive Statistics

In this section, we present a comprehensive descriptive analysis of the data collected in our study. The primary aim is to summarize the main characteristics of the dataset through statistical measures and visual representations. This analysis serves as the groundwork for further inferential analysis in subsequent sections.

Figure 4.1.

Developers' velocity for programming tasks across all experience groups



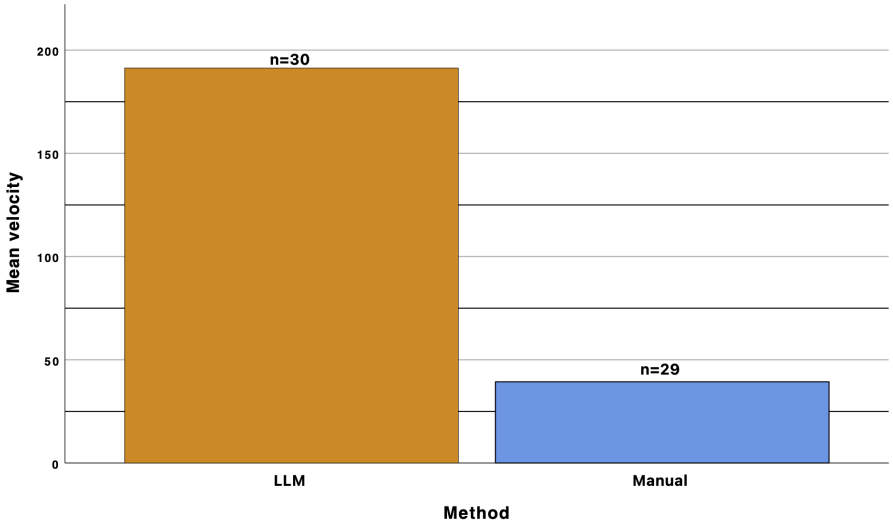
Note. This figure illustrates the differences in mean velocity between developers who used the LLM-assisted method for comprehending necessary documentation to complete programming tasks and those who employed the manual method.
n. Denotes the number of participants.

Developers' velocity: Figures 4.1 and 4.2 show the developers' mean velocity for the two methods across all the experience blocks, for programming and documentation tasks, respectively. Figure 4.1 shows an indication that developers utilizing the LLM-assisted method exhibit increased velocity in completing programming tasks that require documentation comprehension. Likewise, Figure 4.2 shows a similar indication in which developers also achieve greater velocity in producing documentation when using the LLM-assisted method.

Given that developer experience is a controlled variable, it is necessary to perform the comparison of the two documentation methods within each experimental block. Figures 4.3 and 4.4 depict the developer's velocity, segmented by experience level, for programming and documentation tasks, respectively. Figures 4.3 and 4.4 indicate that developers within all experience blocks exhibit increased velocity in both programming and documentation tasks when utilizing the LLM-assisted method. Additionally, it is noticeable that the difference in developers' velocity between the two methods is more pronounced for documentation tasks than for programming

Figure 4.2.

Developers' velocity for documentation tasks across all experience groups



Note. This figure illustrates the differences in mean velocity between developers who used the LLM-assisted method for producing development documentation and those who employed the manual method.
n. Denotes the number of participants.

tasks.

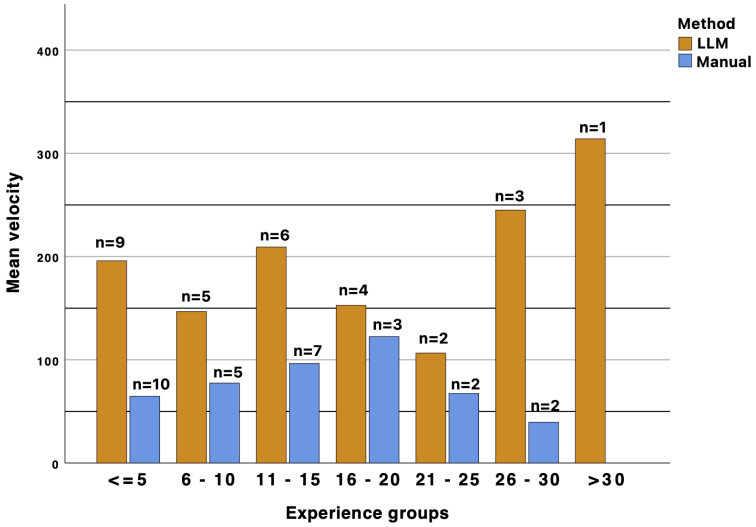
Developers' effectiveness: Figures 4.5 and 4.6 illustrate the mean effectiveness of developers using the two different methods across all experience blocks for programming and documentation tasks, respectively. From figures 4.5 and 4.6, it is possible to see that developers who use the LLM-assisted method exhibit a higher level of effectiveness on average in both programming tasks that require documentation comprehension and in generating documentation.

Figures 4.7 and 4.8 illustrate the mean effectiveness scores for programming and documentation tasks, respectively, based on different experience levels. Based on Figures 4.7 and 4.8, it is notable that developers demonstrate better effectiveness levels when using the LLM-assisted method as compared to the manual method for both programming and documentation tasks.

Documentation quality: Figure 4.9 illustrates the differences in the means of total scores for documentation quality between documentation created by software developers and that generated by the LLM-powered documentation system. Figures 4.10, 4.11, and 4.12 present the mean scores for readability, comprehensibility, and completeness, respectively. These figures illustrate developers' perceptions of documentation quality in these aspects for documentation created by developers versus documentation generated by the LLM-powered system. Figure 4.9 suggests that developers perceive the documentation generated by the LLM-powered system as being of higher quality when compared to documentation manually written by developers. However, the indication of disparity between the two methods does not seem to be

Figure 4.3.

Developers' velocity for programming tasks within experience groups



Note. This figure compares the mean velocity of developers within each experience group, distinguishing between those who used an LLM-assisted method and those who used the manual method to comprehend the necessary documentation for completing programming tasks.
n. Denotes the number of participants.

significant. This can be due to the comparable levels of perceived readability between the documentation produced by the LLM-powered system and that manually written by developers, as illustrated in Figure 4.10. However, Figures 4.11, and 4.12 indicate that developers perceive the comprehensibility and completeness of the documentation generated by the LLM-powered system to be superior to that of the manually written documentation.

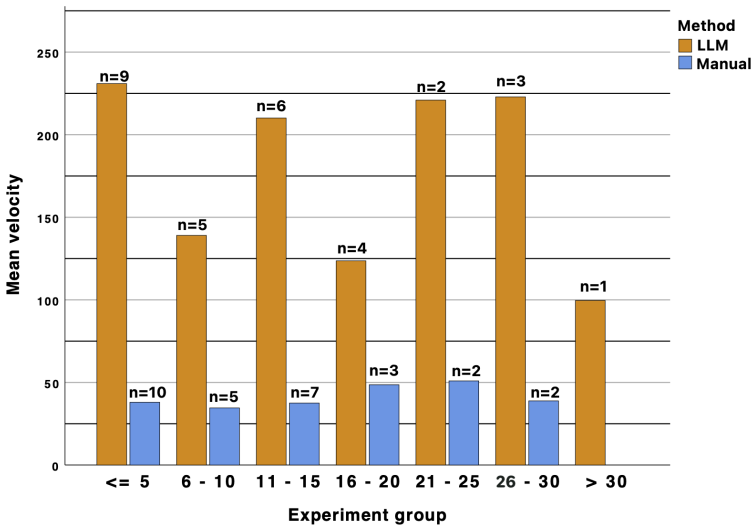
4.2.2 Normality Testing

In this section of the analysis, we investigate the assumption of normality for our dataset, which is fundamental for the valid application of the parametric test we intend to use for our hypotheses testing. To assess the normality assumption, the Shapiro-Wilk test is employed with a critical P-value of 0.05.

The Shapiro-Wilk test was initially applied to all experimental blocks across the two documentation methods used in the programming tasks. As indicated in Table 4.1, the Shapiro-Wilk results for the velocity dimension reveal significant values for both documentation methods, surpassing the critical P-value of 0.05. This suggests that the velocity data recorded for the programming tasks are normally distributed. Conversely, the results for the effectiveness dimension of the LLM-assisted method reveal a significant value below the critical P-value. This suggests that the effectiveness data for the LLM-assisted method is not normally distributed. Table 4.2 shows similar findings regarding normality. The velocity data is normally distributed across

Figure 4.4.

Developers' velocity for documentation tasks within experience groups



Note. This figure compares the mean velocity of developers within each experience group, distinguishing between those who used an LLM-assisted method and those who used the manual method to produce development documentation in documentation tasks.
n. Denotes the number of participants.

Table 4.1

Tests of Normality for Programming Tasks

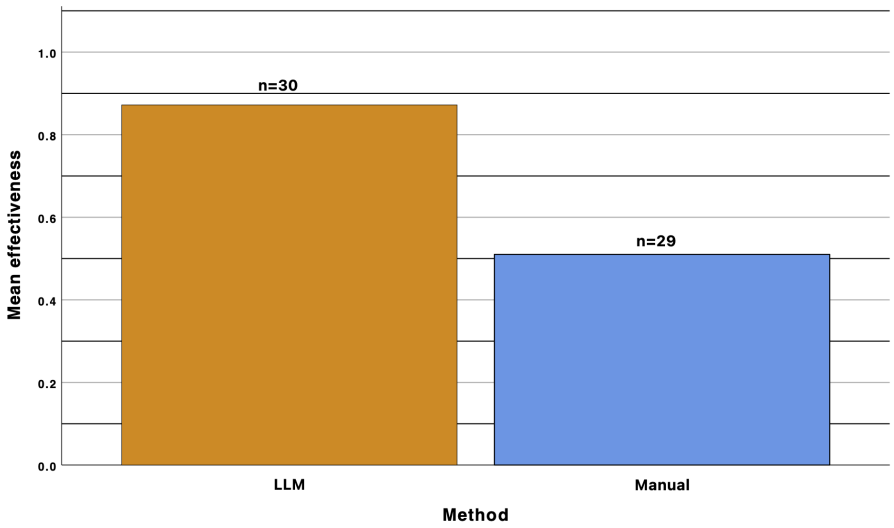
		Shapiro-Wilk Test		
Variable	Method[a]	Statistic[b]	df[c]	Sig.[d]
Velocity	LLM	0.960	30	0.316
	Manual	0.951	29	0.197
Effectiveness	LLM	0.758	30	<0.001
	Manual	0.970	29	0.556

Note. This table presents the results of the normality test conducted on the entire dataset related to developers' effectiveness and velocity, which was collected from programming tasks completed by developers during the controlled experiment.
[a] The method used for comprehending the required documentation to complete the programming task, either manual method or LLM-assisted method.
[b] Measure how well the ordered and standardized sample quantiles fit the standard normal quantiles.
[c] Degrees of freedom.
[d] Significance level.

both documentation methods for the documentation tasks. However, the effectiveness data from the LLM-assisted method does not exhibit a normal distribution.

Figure 4.5.

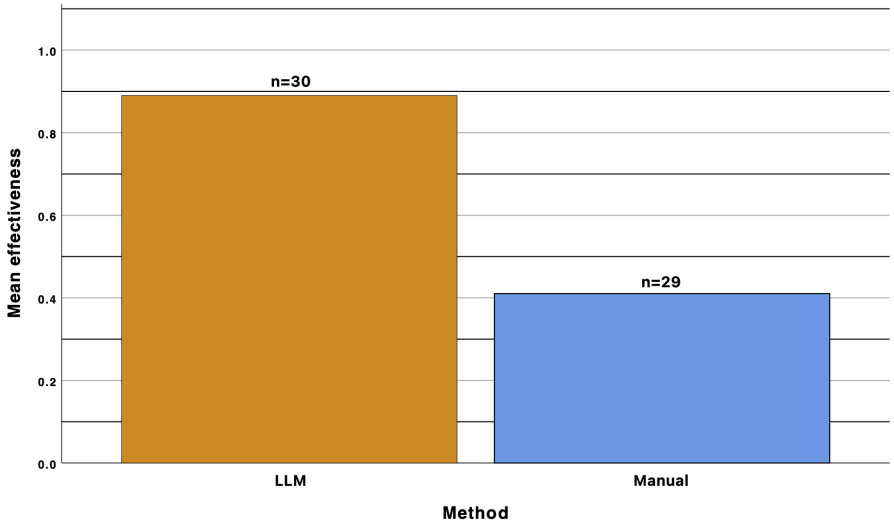
Developers' effectiveness for programming tasks across all experience groups



Note. This figure illustrates the differences in mean effectiveness between developers who used the LLM-assisted method for comprehending necessary documentation to complete programming tasks and those who employed the manual method.
n. Denotes the number of participants.

Figure 4.6.

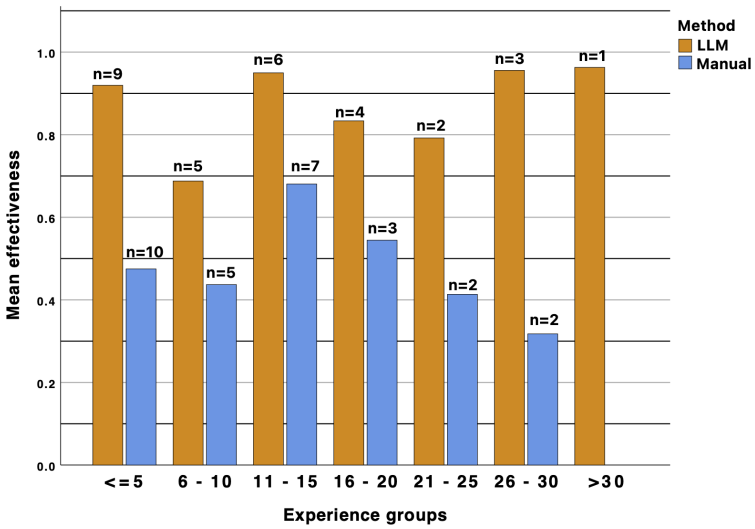
Developers' effectiveness for documentation tasks across all experience groups



Note. This figure illustrates the differences in mean effectiveness between developers who used the LLM-assisted method for producing development documentation and those who employed the manual method.
n. Denotes the number of participants.

Figure 4.7.

Developers' effectiveness for programming tasks within experience groups



Note. This figure compares the mean effectiveness of developers within each experience group, distinguishing between those who used an LLM-assisted method and those who used the manual method to comprehend the necessary documentation for completing programming tasks.
n. Denotes the number of participants.

Given the adopted experimental design in which the comparison between the pro-

Table 4.2

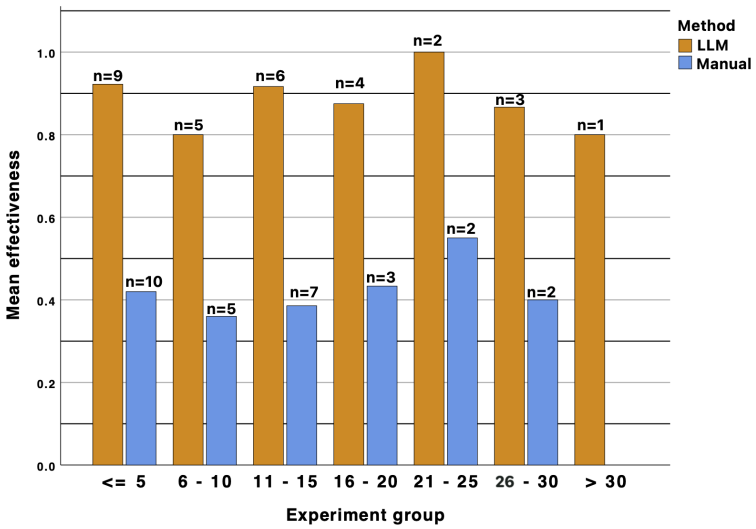
Tests of Normality for Documentation Tasks

		Shapiro-Wilk Test		
Variable	Method[a]	Statistic[b]	df[c]	Sig.[d]
Velocity	LLM	0.975	30	0.683
	Manual	0.974	29	0.659
Effectiveness	LLM	0.803	30	<0.001
	Manual	0.942	29	0.112

Note. This table presents the results of the normality test conducted on the entire dataset related to developers' effectiveness and velocity, which was collected from documentation tasks completed by developers during the controlled experiment.
[a] The method used for producing the documentation, either manual method or LLM-assisted method.
[b] Measure how well the ordered and standardized sample quantiles fit the standard normal quantiles.
[c] Degrees of freedom.
[d] Significance level.

Figure 4.8.

Developers' effectiveness for documentation tasks within experience groups



Note. This figure compares the mean effectiveness of developers within each experience group, distinguishing between those who used an LLM-assisted method and those who used the manual method to produce development documentation in documentation tasks.
n. Denotes the number of participants.

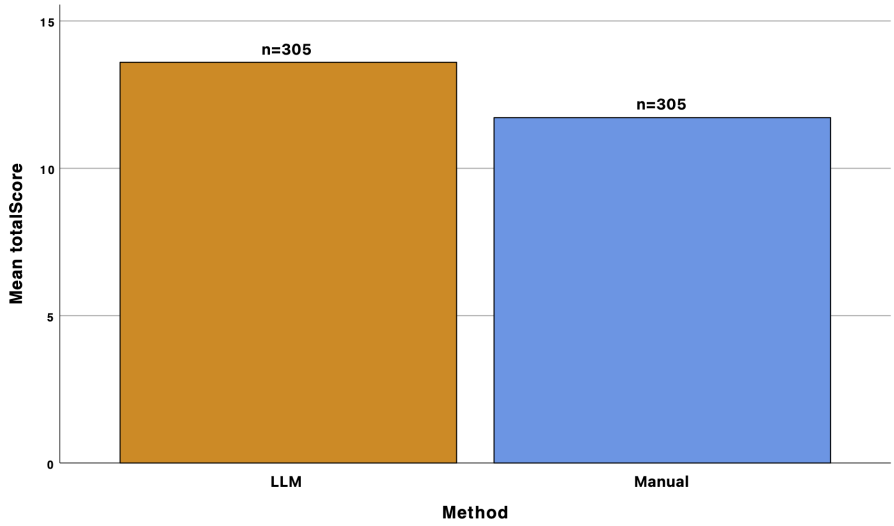
ductivity groups is made within each experimental block, further normality testing is conducted within each design block. The Shapiro-Wilk test is used to assess the normality of both velocity and effectiveness data within each block to select the appropriate statistical test for hypotheses testing. Our initial review of the dataset showed that the experience groups of $21 - 25$, $26 - 30$, and over 30 years lacked enough data points to perform the Shapiro-Wilk test. Consequently, we decided to consolidate these groups into a single category, comprising all participants with more than 20 years of work experience.

The results, presented in Table 4.3, show that the velocity data of developers performing programming tasks for both documentation methods are normally distributed in all experience groups, as indicated by significant values exceeding the critical P-value of 0.05. In contrast, the significance values for the effectiveness data in the LLM-assisted method fall below the critical P-value in group ≤ 5 and group *11 - 15*. Based on the findings presented in Table 4.3, the t-test with two independent samples is selected to test the velocity dimension within each experience group. Meanwhile, the Mann-Whitney test is selected to test the effectiveness dimension.

The results, presented in Table 4.4, show that the velocity data of developers for documentation tasks for both documentation methods are normally distributed in all experience groups, as indicated by significant values exceeding the critical P-value of 0.05. In contrast, the significance values for the effectiveness data in the LLM-assisted method fall below the critical P-value in group ≤ 5 and group *16 - 20*. Based on the findings presented in Table 4.4, the t-test with two independent samples is

Figure 4.9.

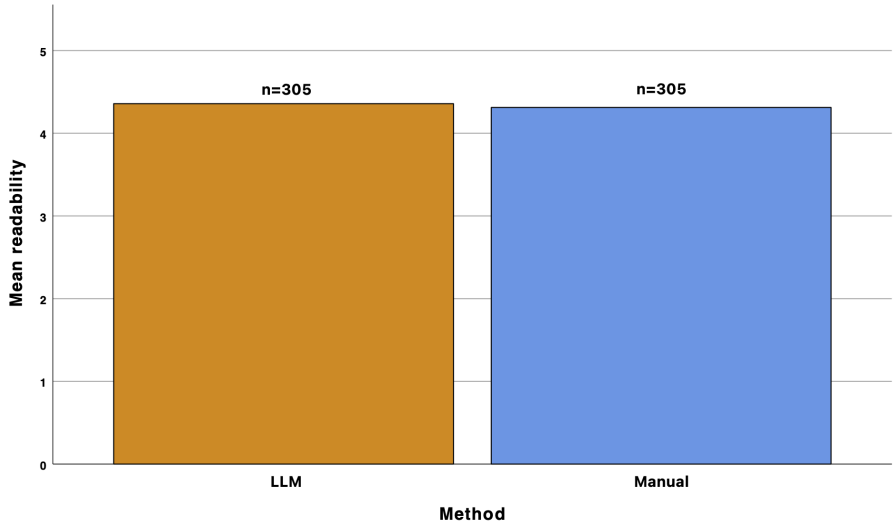
Mean total quality scores for manually created vs. LLM-generated documentation



Note. This figure illustrates the mean cumulative total quality scores assigned by developers to documentation manually created by developers compared to documentation generated by the LLM-powered documentation system.
n. Denotes the number of rated documentation artifacts.

Figure 4.10.

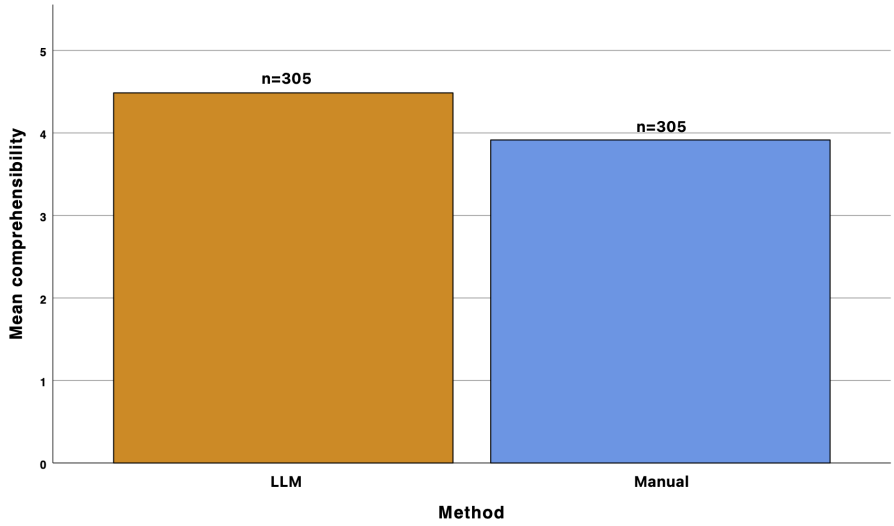
Readability Scores: Manual vs. LLM-Generated Documentation



Note. This figure illustrates the mean readability scores assigned by developers for documentation manually created by developers vs. documentation generated by the LLM-powered documentation system.
n. Denotes the number of rated documentation artifacts.

Figure 4.11.

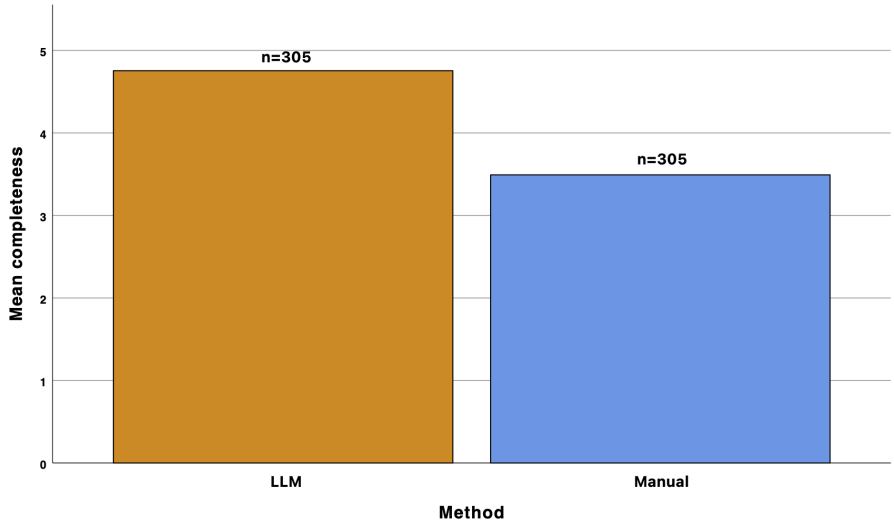
Comprehensibility Scores: Manual vs. LLM-Generated Documentation



Note. This figure illustrates the mean comprehensibility scores assigned by developers for documentation manually created by developers vs. documentation generated by the LLM-powered documentation system.
n. Denotes the number of rated documentation artifacts.

Figure 4.12.

Completeness Scores: Manual vs. LLM-Generated Documentation



Note. This figure illustrates the mean completeness scores assigned by developers for documentation manually created by developers vs. documentation generated by the LLM-powered documentation system.
n. Denotes the number of rated documentation artifacts.

Table 4.3

Test of Normality for Programming Tasks within Each Experience Group

			Shapiro-Wilk Test		
Experience Group	Variable	Method[a]	Statistic[b]	df[c]	Sig.[d]
≤ 5	Velocity	llm	0.857	9	0.090
		manual	0.962	10	0.804
	Effectiveness	llm	0.791	9	0.016
		manual	0.894	10	0.186
6 - 10	Velocity	llm	0.981	5	0.940
		manual	0.845	5	0.179
	Effectiveness	llm	0.908	5	0.456
		manual	0.953	5	0.755
11 - 15	Velocity	llm	0.962	6	0.838
		manual	0.944	7	0.677
	Effectiveness	llm	0.791	6	0.049
		manual	0.934	7	0.588
16 - 20	Velocity	llm	0.947	4	0.696
		manual	0.893	3	0.365
	Effectiveness	llm	0.794	4	0.093
		manual	0.987	3	0.783
> 20	Velocity	llm	0.886	6	0.297
		manual	0.857	4	0.250
	Effectiveness	llm	0.784	6	0.042
		manual	0.982	4	0.914

Note. This table shows the results of normality tests on the effectiveness and velocity of developers from different experience groups, based on data collected from programming tasks completed during the controlled experiment.
[a] The method used for comprehending the required documentation to complete the programming task, either manual method or LLM-assisted method.
[b] Description for Statistic
[c] Degrees of freedom.
[d] Significance level.

selected to test the velocity dimension within each experience group. Meanwhile, the Mann-Whitney test is selected to test the effectiveness dimension. Additionally, the Mann-Whitney test is employed to compare the mean total quality scores between the two methods.

Table 4.4

Test of Normality for Documentation Tasks Within Each Experience Group

			Shapiro-Wilk Test		
Experiment Group	Variable	Method[a]	Statistic[b]	df[c]	Sig.[d]
≤ 5	Velocity	llm	0.936	9	0.543
		manual	0.848	10	0.056
	Effectiveness	llm	0.684	9	<0.001
		manual	0.835	10	0.038
6 - 10	Velocity	llm	0.996	5	0.996
		manual	0.863	5	0.237
	Effectiveness	llm	0.836	5	0.154
		manual	0.952	5	0.754
11 - 15	Velocity	llm	0.872	6	0.236
		manual	0.955	7	0.779
	Effectiveness	llm	0.866	6	0.212
		manual	0.894	7	0.294
16 - 20	Velocity	llm	0.874	4	0.315
		manual	0.967	3	0.653
	Effectiveness	llm	0.630	4	0.001
		manual	0.964	3	0.637
> 20	Velocity	llm	0.951	6	0.746
		manual	0.802	4	0.106
	Effectiveness	llm	0.831	6	0.110
		manual	0.630	4	0.001

Note. This table shows the results of normality tests on the effectiveness and velocity of developers from different experience groups, based on data collected from documentation tasks completed during the controlled experiment.

[a] The method used for producing the documentation, either manual method or LLM-assisted method.

[b] Measure how well the ordered and standardized sample quantiles fit the standard normal quantiles.

[c] Degrees of freedom.

[d] Significance level.

4.2.3 Hypotheses Testing

4.2.3.1 Hypothesis related to developer's productivity for documentation creation ($\mathbf{H_{0.Gen}}$)

The null hypothesis states that: "There is no significant improvement in developer productivity when using an LLM-powered documentation system for producing development documentation compared to manual documentation methods, in which documentation are manually written by developers.". Consequently, to test this hypothesis, it's essential to assess improvements in both the velocity and effectiveness of developers in documentation tasks. Thus, the alternative hypothesis is formulated as follows:

- $H_{1.Gen,a}$:VEL_{Gen}(LLM assisted method) > VEL_{Gen}(manual method)
- $H_{1.Gen,b}$:EFF_{Gen}(LLM assisted method) > EFF_{Gen}(manual method)

As stated in Normality Testing, hypothesis testing is conducted using the independent-samples t-test on experimental groups that adhere to a normal distribution. For groups that do not follow a normal distribution, the Mann-Whitney test is used. Both tests are carried out with a significance level set at $\alpha = 0.05$.

Testing developers' velocity:
A Levene test for equality of variances was performed across all experience groups to assess whether variance equality could be assumed for the conducted t-test. The results of the Levene test are presented in Table 4.5. The Levene test result for the experience group ≤ 5 was not significant, with a value of 0.01, which is below $\alpha = 0.05$, indicating that variance equality could not be assumed for this group. However, for the remaining groups, since the Levene test results exceeded the critical alpha value of 0.05, variance equality was assumed. Given that the velocity data for

Table 4.5

Levene's Test for Equality of Variances of velocity for documentation tasks

	Levene Test	
Experience group	F-value[a]	Sig.[b]
≤ 5 years	8.414	0.010
6-10 years	4.030	0.080
11-15 years	2.797	0.123
16-20 years	2.341	0.187
> 20 years	3.399	0.102

Note. This table shows the results from the Levene test conducted to assess the equality of variances in developer velocities for documentation tasks. The test compares the variances between two method groups within each experience group.
[a] Tests if the variances are equal. When the F value and the significance level is small, the hypothesis of equal variances can be rejected
[b] The conditional probability that a relationship as strong as the one observed in the data would be present, if the null hypothesis were true.

documentation tasks follow a normal distribution across all experimental groups, an independent-sample t-test is conducted for each group. The results of conducting the independent-sample t-test within each experiment group are presented in Table 4.6. For the experience group ≤ 5 an independent-samples t-test with unequal variances was conducted to compare the developers' velocity for producing documentation between the two documentation method groups. The analysis revealed a significant difference ($t(9.58) = 13.70, p < 0.001$), indicating that the group using the LLM-assisted method outperformed the manual method in terms mean velocity. The magnitude of the difference in means (Mean difference = 192.957, 95% Confidence Interval(CI)= 161.38 to 224.53) was significant. Thus, the null hypothesis that there is no significant improvement in developers' velocity when using the LLM-assisted method for producing documentation compared to the manual method ($H_{0.Gen,a}$) was rejected for the experience group ≤ 5. Table 4.6 also shows similar restuls for the remaining experimental groups, where the null hypothesis can be rejected for these groups.

Furthermore, conducting the t-test with unequal variance for all the experience groups collectively reveals a significant difference ($t(11.49) = 31.52, p < 0.001$), indicating that the group using the LLM-assisted method surpassed the manual method in terms of mean velocity. The magnitude of the difference in means (Mean difference = 151.96, 95% Confidence Interval(CI)= 125 to 178.92) was significant. Thus, the null hypothesis that there is no significant improvement in developers' velocity when using the LLM-assisted method for producing documentation compared to the manual method ($H_{0.Gen,a}$) was also rejected when analyzing all the samples collectively.

Testing developers' effectiveness:
Given that the effectiveness data in multiple experience groups is not normally distributed, the Mann-Whitney U test is selected to test the hypothesis. Table 4.7 shows the results of conducting the Mann-Whitney U test for developers' effectiveness in documentation tasks, in which the null hypothesis ($H_{0.Gen,b}$) is rejected for all experience groups except the $16-20$ group. Figures C.1 to C.5 in the appendix show the ranking results for the effectiveness of the two documentation methods. The results indicates that in experience groups where the null hypothesis was rejected, developers employing the LLM-assisted method demonstrated higher effectiveness compared to those using manual methods. This is evident as the average rankings are higher in the groups rejecting the null hypothesis than those for the manual method.

Furthermore, Table 4.8 presents the results of conducting the Mann-Whitney U test for all the experience groups collectively. As depicted in Table 4.8, the null hypothesis that the two documentation methods' effectiveness means for documentation tasks do not differ ($H_{0.Gen,b}$) was rejected. The ranking results can be seen in Figure C.6 in the appendix, where developers using the LLM-assisted method have higher effectiveness compared to those using the manual method.

4.2.3.2 Hypothesis related to developer's productivity for documentation comprehension ($H_{0.Comp}$)

The null hypothesis states that "There is no significant improvement in developer productivity when using an LLM-powered documentation system for documentation

Table 4.6

Independent Samples t-test of Velocity Data for Documentation Tasks

Experience group	Equal variances	t[a]	df[b]	Sig. One-Sided p[c]	Mean Diff.[d]	95% CI of the Difference[e]	
						Lower	Upper
<=5	true	14.326	17	< 0.001	192.96	164.54	221.37
	false	13.698	9.576	< 0.001	192.96	161.38	224.53
6-10	true	3.211	8	0.006	104.45	29.44	179.46
	false	3.211	4.614	0.013	104.45	18.68	190.21
11-15	true	7.880	11	< 0.001	172.58	124.37	220.78
	false	7.305	5.477	< 0.001	172.58	113.41	231.75
16-20	true	2.174	5	0.041	75.03	−13.69	163.75
	false	2.515	3.556	0.037	75.03	−12.04	162.10
>20	true	3.462	8	0.004	156.81	52.37	261.26
	false	4.268	5.470	0.003	156.81	64.76	248.87

Note. This table summarizes the outcomes of independent samples t-tests performed to evaluate the null hypothesis related to developers' velocity in documentation tasks across different experience groups.

[a] The t-statistics under the two different assumptions: equal variances and unequal variances. These are the ratios of the mean of the differences to the standard errors of the difference under the two different assumptions.

[b] The degrees of freedom.

[c] The p-value where the observed direction is predicted in advance (The mean difference is higher than zero in this case).

[d] This is the difference between the means.

[f] These are the lower and upper bounds of the confidence interval for the mean difference. A confidence interval for the mean specifies a range of values within which the unknown population parameter may lie.

comprehension compared to manual methods, in which developers manually search for relevant information by reading static (PDF, HTML, Markdown, etc) documentation artifacts." Thus, to test this hypothesis, it's essential to assess improvements in both the velocity and effectiveness of developers in programming tasks that re-

Table 4.7

The results of Mann-Whitney U Test for developers' effectiveness in documentation tasks within each experience block

Experience group	Null Hypothesis	Sig.[a]	Decision
≤ 5	The distribution of effectiveness is the same across categories of method.	< 0.001[b]	Reject the null hypothesis.
$6 - 10$	The distribution of effectiveness is the same across categories of method.	0.032[b]	Reject the null hypothesis.
$11 - 15$	The distribution of effectiveness is the same across categories of method.	0.001[b]	Reject the null hypothesis.
$16 - 20$	The distribution of effectiveness is the same across categories of method.	0.057[b]	Retain the null hypothesis.
> 20	The distribution of effectiveness is the same across categories of method.	0.010[b]	Reject the null hypothesis.

Note. This table reports the results of the Mann-Whitney U test assessing differences in effectiveness across different experience groups for documentation tasks. The null hypotheses are rejected if $Sig. \leq 0.05$.
[a] The significance level is 0.050.
[b] Exact significance is displayed for this test.

Table 4.8

The results of Mann-Whitney U Test for developers' effectiveness in documentation tasks

	Mann-Whitney U Test	
Null Hypothesis	Sig.[a]	Decision
The distribution of effectiveness is the same across categories of method.	< 0.001[b]	Reject the null hypothesis

Note. This table summarizes the results of the Mann-Whitney U test evaluating differences in developer effectiveness across documentation methods. The null hypotheses are rejected if $Sig. \leq 0.05$.
[a] The significance level is 0.050.
[b] Exact significance is displayed for this test.

quire documentation comprehension, and developers' perception of the documentation quality. Hence, the alternative hypothesis is formulated as follows:

- $H_{1.Comp,a}$:VEL_{Comp}(LLM assisted method) $> VEL_{Comp}$(manual method)
- $H_{1.Comp,b}$:EFF_{Comp}(LLM assisted method) $> EFF_{Comp}$(manual method)

- $H_{1.Comp,c}$:$QLTY_{Comp}$(LLM assisted method) > $QLTY_{Comp}$(manual method)

As stated in Normality Testing, hypothesis testing is conducted using the independent-samples t-test on experimental groups that adhere to a normal distribution. For groups that do not follow a normal distribution, the Mann-Whitney U test is used. Both tests are carried out with a significance level set at $\alpha = 0.05$.

Testing developers' velocity:
A Levene test for equality of variances was performed across all experience groups to assess whether variance equality could be assumed for the conducted t-test. The results of the Levene test are presented in Table 4.9. The Levene test results for the experience groups ≤ 5 and > 20 were not significant, with values of 0.01 and 0.005 respectively, which is below $\alpha = 0.05$, indicating that variance equality can not be assumed for these groups. However, for the remaining groups, since the Levene test results exceeded the critical alpha value of 0.05, variance equality was assumed.

Table 4.9

Levene's Test for Equality of Variances of velocity for programming tasks

	Levene Test	
Experience group	F-value[a]	Sig.[b]
≤ 5 years	8.445	0.010
6-10 years	0.946	0.359
11-15 years	0.124	0.731
16-20 years	0.472	0.523
> 20 years	14.601	0.005

[a] F-value: Statistic value of the F-test for equal variances.
[b] Sig.: Significance level, indicating the probability that the observed differences in variances are due to chance.

Given that the velocity data for programming tasks follow a normal distribution across all experimental groups, an independent-sample t-test is conducted for each group. The results of conducting the independent-sample t-test within each experience group are presented in Table 4.10. For each of the experience groups, an independent-sample t-test with either unequal or equal variances was conducted to compare the developers' velocity for completing programming tasks that require documentation comprehension between the two documentation method groups. The results presented in Table 4.10 show that the null hypothesis ($H_{0.Comp,a}$) can be rejected for experience groups ≤ 5, $11 - 15$, and > 20, whereas it should be retained for the $6 - 10$ and $16 - 20$ groups.

Furthermore, the t-test with unequal variance for all the experience groups collectively was conducted. The results reveal a significant difference ($t(49) = 6.79, p < 0.001$), indicating that the group using the LLM-assisted method surpassed the manual method in terms of mean velocity. The magnitude of the difference in means (Mean difference = 108.52, 95% Confidence Interval(CI)= 76.40 to 140.64) was significant. Thus, the null hypothesis that there is no significant improvement in developers' velocity when using the LLM-assisted method for programming tasks

Table 4.10

Independent Samples t-test of Velocity Data for Programming Tasks

Experience group	Equal variances	t[a]	df[b]	Sig. One-Sided p[c]	Mean Diff.[d]	95% CI of the Difference[e]	
						Lower	Upper
<=5	true	5.184	17	< 0.001	131.19	77.80	184.59
	false	5.056	13.060	< 0.001	131.19	75.16	187.22
6-10	true	1.791	8	0.056	69.30	−19.91	158.51
	false	1.791	6.026	0.062	69.30	−25.26	163.86
11-15	true	3.368	11	0.003	112.79	39.08	186.49
	false	3.314	9.740	0.004	112.79	36.67	188.90
16-20	true	0.612	5	0.284	30.24	−96.76	157.23
	false	0.632	4.883	0.278	30.24	−93.61	154.08
>20	true	3.297	8	0.005	156.93	47.18	266.67
	false	4.033	5.741	0.004	156.93	60.66	253.20

Note. This table summarizes the outcomes of independent samples t-tests performed to evaluate the null hypothesis related to developers' velocity in programming tasks across different experience groups.

[a] The t-statistics under the two different assumptions: equal variances and unequal variances. These are the ratios of the mean of the differences to the standard errors of the difference under the two different assumptions.

[b] The degrees of freedom.

[c] The p-value where the observed direction is predicted in advance (The mean difference is higher than zero in this case).

[d] This is the difference between the means.

[f] These are the lower and upper bounds of the confidence interval for the mean difference. A confidence interval for the mean specifies a range of values within which the unknown population parameter may lie.

compared to the manual method ($H_{0.Comp,a}$) was also rejected when analyzing all the samples collectively.

Testing developers' effectiveness:
Given that the effectiveness data in multiple experience groups is not normally distributed, the Mann-Whitney U test is selected to test the hypothesis. Table 4.11 shows the results of conducting the Mann-Whitney U test for developers' effectiveness in programming tasks, in which the null hypothesis ($H_{0.Comp,b}$) is rejected for the groups ≤ 5 and > 20 while retained for the rest of the experience groups. Figures C.7 to C.11 in the appendix show the ranking results for the effectiveness of the two documentation methods. The results indicates that in experience groups where the null hypothesis was rejected, developers employing the LLM-assisted method demonstrated higher effectiveness compared to those using the manual method. This is evident as the mean rankings are higher in the groups rejecting the null hypothesis than those for the manual method. This trend is also present in the experience groups where the null hypothesis was retained, where the mean effectiveness rankings are still higher for developers who used the LLM-assisted method compared to those who used the manual method.

Table 4.11

The results of Mann-Whitney U Test for developers' effectiveness in programming tasks within each experience block

Experience group	Null Hypothesis	Sig.[a]	Decision
≤ 5	The distribution of effectiveness is the same across categories of method.	0.001[b]	Reject the null hypothesis.
$6 - 10$	The distribution of effectiveness is the same across categories of method.	0.151[b]	Retain the null hypothesis.
$11 - 15$	The distribution of effectiveness is the same across categories of method.	0.073[b]	Retain the null hypothesis.
$16 - 20$	The distribution of effectiveness is the same across categories of method.	0.229[b]	Retain the null hypothesis.
> 20	The distribution of effectiveness is the same across categories of method.	0.010[b]	Reject the null hypothesis.

Note. This table reports the results of the Mann-Whitney U test assessing differences in effectiveness across different experience groups. The null hypotheses are rejected if $Sig. \leq 0.05$.
[a] The significance level is 0.050.
[b] Exact significance is displayed for this test.

Furthermore, Table 4.12 presents the results of conducting the Mann-Whitney U test for all the experience groups collectively. As depicted in Table 4.12, the null

hypothesis that the two documentation methods' effectiveness means do not differ for programming tasks ($H_{0.Comp,b}$) was rejected. The ranking results can be seen in Figure C.12 in the appendix, where developers using the LLM-assisted method have higher effectiveness compared to those using the manual method.

Table 4.12

The results of Mann-Whitney U Test for developers' effectiveness in programming tasks

	Mann-Whitney U Test	
Null Hypothesis	Sig.[a]	Decision
The distribution of effectiveness is the same across categories of method.	< 0.001[b]	Reject the null hypothesis

Note. This table summarizes the results of the Mann-Whitney U test evaluating differences in developer effectiveness across programming methods. The null hypotheses are rejected if $Sig. \leq 0.05$.
[a] The significance level is 0.050.
[b] Exact significance is displayed for this test.

Testing documentation quality:
Given that the quality measures (readability, completeness, and comprehensibility) are on Likert scale (ordinal scale), we have made the assumption that the total quality score also is ordinal in nature. Thus, the Mann-Whitney U test has been chosen to test the null hypothesis related to documentation quality ($H_{0.Comp,c}$). As depicted in Table 4.13, the null hypothesis that the two documentation methods' total quality score means do not differ ($H_{0.Comp,c}$) was rejected. The ranking results can be seen in

Table 4.13

The results of Mann-Whitney U Test for the documentation quality evaluation

	Mann-Whitney U Test	
Null Hypothesis	Sig.[a]	Decision
The distribution of total quality score is the same across categories of method.	< 0.001[b]	Reject the null hypothesis

Note. This table summarizes the results of the Mann-Whitney U test evaluating differences in documentation quality across different methods. The null hypotheses are rejected if $Sig. \leq 0.05$.
[a] The significance level is 0.050.
[b] Exact significance is displayed for this test.

Figure C.13 in the appendix, where documentations produced by the LLM-powered documentation system have higher total quality scores compared to those that are manually written by developers. These results have an indication that developers might perceive documentation generated by the LLM-powred documentation system to have higher quality scores compared to documentation written manually by developers.

Chapter 5

Discussion

Our thesis explores the impact of incorporating generative AI and LLMs into software documentation processes and their effects on developer productivity. Numerous studies have examined the integration of AI technologies, including generative AI, into various software development practices. While these technologies have been investigated for software documentation purposes with a focus on their feasibility and performance, there is still a notable gap in the literature regarding the specific influence of these technologies on developer productivity. Therefore, our research aims to determine whether a documentation system powered by an LLM can improve tasks related to software documentation in software development. To achieve this, we carried out a controlled experiment followed by a survey to assess the impact of an LLM-powered documentation system on developers' productivity. We specifically evaluated the velocity, effectiveness, and quality dimensions of productivity. In this section, we will discuss the findings of our study and provide a detailed interpretation of the results outlined in Results and Analysis.

5.1 Developers' Effectiveness ($RQ_{1.1}$ and $RQ_{1.2}$)

In our study, we assessed developer effectiveness by calculating the ratio of correctly completed tasks to the total number of tasks. This evaluation was applied across two types of tasks: (a) programming tasks that require documentation comprehension, and (b) documentation tasks in which developers produce development documentation.

For programming tasks, our analysis revealed that for multiple experience groups, a statistically significant improvement in effectiveness can't be inferred for developers using the LLM-powered documentation system. However, the results indicated a statistically significant improvement in effectiveness for developers with either less than or equal to five years of experience or more than twenty years, when employing the LLM-assisted method. Despite these variations, our general observation from the data analysis suggests that the use of the LLM-assisted method tended to be more effective in comprehending documentation compared to the manual method, although this trend was not uniformly statistically significant across all experience groups. We propose that this observed disparity in significant statistical results among certain groups could stem from an insufficient number of participants in those groups. This trend is supported by hypothesis testing on the overall effectiveness of developers using the LLM-assisted method across all experience groups combined, where the results suggest a general enhancement in developer effectiveness. Further

support for this observation can be seen in the data visualizations, which depict the LLM-assisted method as performing better than the manual method across each experience group and collectively (refer to figures 4.5 and 4.7). Additionally, even in experience groups where no statistically significant improvement was noted, the mean effectiveness rankings for developers utilizing the LLM system were higher than those who did not (see figures C.7-C.11).

In examining documentation tasks, our data analysis suggests that there is a statistically significant improvement in developers' effectiveness when employing the LLM-powered documentation system across various experience levels, with the exception of developers with 16 to 20 years of experience. As indicated in programming tasks, it appears that developers across all experience groups tend to be more effective when utilizing the LLM-powered documentation system. This observation is supported by hypothesis testing across all experience groups, which reveals a statistically significant improvement in effectiveness with the use of this system for producing development documentation. Additionally, it is noted that the sample size within the experience group where the null hypothesis was not rejected was relatively small. Nevertheless, even in this group, the mean effectiveness ratings for developers using the LLM-powered documentation system were generally higher than those for developers who did not use the system. Data visualizations also provide some indications that, in general, the LLM-assisted method tends to surpass the manual method in terms of effectiveness.

5.2 Developers' Velocity ($RQ_{1.3}$ and $RQ_{1.4}$)

We assessed the developers' velocity by measuring the ratio of successfully completed tasks to the time spent on these tasks, expressing this metric in tasks per hour. This approach was employed for both programming and documentation tasks. Regarding programming tasks, our analysis suggested a notable, statistically significant association between the utilization of the LLM-powered documentation system and increased developer velocity across various experience levels. However, this observation was not consistent across all experience groups; developers with 6-10 and 16-20 years of experience did not exhibit statistically significant improvements in velocity for programming tasks requiring documentation comprehension. While these specific groups did not exhibit statistically significant improvements, our overall analysis implies a positive trend. Hypothesis testing across the full dataset indicated a statistically significant suggestion that the LLM-assisted approach might enhance developer velocity in programming tasks requiring documentation comprehension. This observation is further supported by data visualization (see figures 4.1 and 4.3), which hints that the LLM-assisted method may surpass the manual method in velocity.

In examining the impact of the LLM-powered documentation system on developers' velocity in documentation tasks, our analysis suggests that there is a statistically significant indication that such a system could potentially enhance developer velocity in these tasks. This trend appears to be consistent across all experience levels, suggesting that the LLM-assisted method may offer notable benefits over the manual method. This suggestion is supported by hypothesis testing performed on the complete dataset, where the LLM-powered system seems to increase developers'

documentation velocity with statistical significance. Additionally, data visualization analyses further corroborate that the LLM-assisted method generally surpasses the manual approach in the velocity of producing development documentation (see figures 4.2 and 4.4).

5.3 Documentation Quality ($RQ_{1.5}$)

To evaluate the quality of documentation, we opted for a survey approach where developers themselves assess the documentation's quality. This approach is justified by the notion that developers, as the primary users of development documentation, are best positioned to evaluate its quality. In this survey, developers rated the documentation on several key aspects: readability, comprehensibility, and completeness. These aspects were rated on a scale from 1 to 5, where 1 signifies poor quality and 5 denotes excellent quality. The aggregate of these scores provided a cumulative measure of the documentation's overall quality.

Our analysis suggests a trend where developers perceive the documentation generated by the LLM-powred documentation system to be of higher quality. This observation is particularly supported by hypothesis testing, which indicated that documentation created by the LLM-powered system is perceived as higher quality, with statistical significance. Additionally, data visualization supports this finding, especially concerning documentation readability, where it appears that LLM-generated documentation is as readable as that authored by developers (see Figure 4.10).

However, there are notable differences in the perceptions of comprehensibility and completeness. For comprehensibility, there is a subtle preference for the LLM-generated documentation, which developers found more understandable (see Figure 4.11). The most significant disparity, however, appears in the completeness of the documentation (see Figure 4.12). Visual data analysis suggests that developers perceive LLM-generated documentation to be more complete compared to developer-written documentation.

We believe these findings are in line with the capabilities of modern advanced LLMs, which can produce text comparable to human-written content. This similarity in readability and comprehensibility between the two sources supports this view. The higher completeness scores for LLM-generated documentation might be attributed to the system's ability to integrate a broader context, which allows for more detailed documentation. This detail-oriented approach, facilitated by the prompts provided to the LLM, is likely appreciated by developers and may address aspects that are sometimes overlooked in manually produced documentation.

5.4 The overall effect on developer productivity (RQ_1)

Based on our analysis, our findings indicate an improvement in developers' overall productivity when using the LLM-powered documentation system. This indication is due to the fact that there are indications of enhancements across three productivity dimensions when using an LLM-powered documentation system. However, we must be cautious in asserting that these findings are universally applicable to all developers.

This restraint stems from several factors: the study was exclusively conducted within a single company, specifically Telefonaktiebolaget LM Ericsson, and limited to a single country, Sweden. These conditions could restrict the broader application of our results to the global developer community. Additionally, the sample size of 59 developers, though adequate for initial insights, may not provide a robust basis for broader generalizations. Nevertheless, the observed indications of productivity improvements are promising and could serve as useful preliminary data for future research in this area.

5.5 Reflecting on potential impacts and cost considerations of the LLM-powered documentation system

In reflecting on this study, it is apparent that integrating LLMs for documentation purposes may offer suggestive evidence of enhanced developer productivity in documentation-related tasks. This study was carried out on a software system unfamiliar to all participating developers, effectively positioning them as newcomers to the project. This context allows us to infer that the use of an LLM-powered documentation system might support the onboarding process for newcomers by potentially decreasing the time and effort required to begin meaningful contributions to a software project. Additionally, the implication that enhanced productivity in documentation tasks may lead to more efficient software engineering processes is worthy of consideration. Since effective documentation is integral to facilitating various other software processes such as development and maintenance, improvements in this area could indirectly contribute to overall efficiencies.

The potential for an LLM-powered documentation system to reduce costs is another aspect touched upon by this study. By diminishing the time developers spend on documentation tasks, there could be a natural reduction in associated costs. This saved time could be reallocated to other critical activities such as feature implementation, bug fixes, or system maintenance, which in turn might reduce the overall time spent on comprehending documentation and allow for more direct engagement with development tasks. However, it is crucial to remain cautious about these implications. The potential cost reductions must be balanced against the expenses involved in operating, maintaining, and implementing the system. Although this study did not factor in the initial cost of system implementation, the ongoing operational costs of the LLM are significant, given the substantial hardware resources required.

Furthermore, while the LLM used in the study was a pre-trained, open-source model that incurred no acquisition costs, the broader financial implications of training and operating such a system cannot be overlooked. The potential returns from investment in the LLM system, theoretically realized as reductions in documentation costs, should be rigorously evaluated. The scalability of these benefits, particularly how they relate to the number of developers utilizing the system, remains a critical variable in this equation.

5.6 Study Limitations

Regarding developers' experience, the interpretation of the term might not effectively reflect accurate relationships between the results of the study and reality. The experience of software developers is broad and includes a wide range of programming languages, software tools, experience in different companies, experience in specific sub-organizations, or experience with specific types of products or solutions. This variability in developer experience could introduce biases that limit the generalizability of our findings. For instance, a developer's familiarity with a particular tool or programming language might disproportionately influence their productivity when using the LLM-powered documentation system, thus skewing the results.

Furthermore, due to time constraints, we did not investigate specific types of programming tasks or documentation tasks to understand where exactly the LLM-powered documentation system most significantly increased developers' productivity. This study broadly assessed the productivity impact without delving into the nuances of different task types. For example, it remains unclear whether the developers' documentation primarily lacked usage examples or input-output type documentation. Similarly, identifying the specific coding tasks that were most challenging for developers was beyond the scope of this research. Addressing these detailed aspects might reveal more about the effectiveness of LLM-powered systems in various contexts and task-specific improvements, offering valuable directions for future research.

Future research should aim to address these limitations by including a more diverse sample of developers with varying levels of experience across different domains and programming languages. Additionally, investigating the impact of LLM-powered documentation systems on specific types of programming and documentation tasks can provide deeper insights into their effectiveness. Expanding the scope to include longitudinal studies in naturalistic settings would also help in understanding the long-term benefits and real-world applicability of these systems. Finally, incorporating more objective measures of documentation quality, alongside subjective evaluations, could enhance the robustness and reliability of the findings.

Chapter 6

Threats to Validity

6.1 Internal Validity

In experimental research, internal validity is crucial for ensuring that the observed effects are indeed due to the interventions applied rather than extraneous variables [49]. The following presents the identified threats to internal validity, detailing each threat and the corresponding countermeasures taken to mitigate their effects.

Threat: Participants may face difficulties during the study due to the use of custom-made study software, potentially impacting the study outcomes because of their unfamiliarity with this software.
Countermeasure: Participants are provided with detailed video tutorials at the start of the study. These tutorials thoroughly explain and demonstrate how to use the custom software, ensuring participants can effectively complete the study tasks.

Threat: In the familiarization phase, where participants view instructional videos, there is a potential issue with participants memorizing the example tasks shown, which could affect the study results.
Countermeasure: The video guides have been carefully crafted to only explain the general process of the study without revealing specific tasks that participants will later undertake, thereby preventing any unintentional memorization that could affect the outcomes.

Threat: The familiarity of some participants with the software system used in the study could potentially influence the results.
Countermeasure: The software system selected for this study was previously employed only in a student project, and its developers were the students themselves. This ensures that no participant has a familiarity advantage regarding the system's code or documentation.

Threat: Participants in the control group who do not have access to the LLM-powered documentation system might feel disengaged and less motivated during the study. This lack of engagement could potentially influence the outcomes of the research.
Countermeasure: All participants are offered access to the LLM-powered documentation system. This approach aims to ensure a more enjoyable and engaging experience for everyone involved, potentially reducing any negative impacts on the

study results.

Threat: There is a risk that researchers might exhibit bias when assessing the correctness of tasks completed by participants, particularly if the tasks involved the use of the LLM-powered documentation system, which could affect the study results in favor of a particular outcome.
Countermeasure: A custom software component has been developed for evaluating tasks. This component conceals the method used by participants during the assessment phase, preventing researchers from knowing whether the LLM-powered documentation system was employed, thus minimizing potential evaluation bias. **Threat:** There is a potential threat to the internal validity of the study due to participants in the control group possibly using external resources, such as ChatGPT, when completing the experiment tasks. This access to additional assistance outside the scope of the provided study materials could lead to a confounding variable that may influence their performance, thereby affecting the comparison between the control group and the experimental group.
Countermeasure: To mitigate this threat, participants were informed explicitly at the start of the experiment about the rules prohibiting the use of any external resources except the documentation for the project.

6.2 External Validity

External validity refers to the extent to which research findings can be applied to other populations and settings beyond the scope of the study [49]. The study was conducted in an industrial setting involving professional software developers who have diverse work experiences. Although the software system utilized in the study was originally developed by students, it has been implemented internally as an actual subsystem, integrating and interacting with other production subsystems. Consequently, the findings are relevant to real software projects and can be generalized to a wider context.

6.3 Construct Validity

Construct validity is crucial for ensuring that experimental designs and outcomes accurately reflect theoretical constructs [49]. During our study, we recognized the difficulties involved in measuring the productivity of developers. To ensure accurate and precise assessments, we followed established methodologies outlined in the literature that emphasize the need to measure various dimensions of productivity. We also used well-tested software components to minimize the risk of measurement errors resulting from human factors.

6.4 Conclusion Validity

In experimental research, conclusion validity concerns the accuracy of drawing relationships between treatment and outcome [49]. Our study design involved a con-

trolled experiment with a sample size determined by the availability of participants within Telefonaktiebolaget LM Ericsson. While efforts were made to ensure sufficient statistical power, the relatively focused participant pool might limit our ability to generalize findings across all software development contexts, potentially increasing the risk of Type II errors. In addition, the LLM-powered documentation system's consistency in providing assistance was critical. Any variability in the performance or output of the LLM could introduce inconsistencies in how the treatment was administered across different testing sessions.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis explored the integration of Genearative AI and LLMs into the software documentation process, specifically focusing on the impact of this technology on developer productivity. Through a rigorous controlled experiment and subsequent survey, it was established that LLM-driven documentation tools have the potential to enhance the effectiveness and velocity of developers in both creating and comprehending software documentation.

The findings from our study confirm that the application of LLMs in software documentation tasks could address several of the productivity challenges faced by developers. Firstly, the LLM-powered documentation system demonstrated a marked improvement in the speed (velocity) with which developers completed tasks related to documentation creation and comprehension. This increase in velocity suggests that LLMs can effectively reduce the time developers spend on documentation tasks, thereby allowing more time for core development activities.

Moreover, the quality of the documentation produced by LLMs, as assessed by the developers, generally met or exceeded that of manually produced documentation. Developers reported improvements in comprehensibility and completeness, which are critical factors in effective software documentation. This not only supports the utility of LLMs in automating documentation tasks but also highlights the potential of these models to enhance the overall quality of documentation output.

7.2 Future Work

While the LLM-driven system has shown promising results, the research exposed new questions about the optimal integration of AI in software engineering practices. It remains to be seen how these systems perform across different environments and types of software projects. The potential for LLMs to adapt to the nuanced needs of diverse development scenarios is an area ripe for further exploration.

This research contributes to the growing body of knowledge in software engineering by demonstrating the practical benefits of applying AI, particularly LLMs, in enhancing developer productivity. Future studies could investigate the long-term impacts of these systems in real-world settings and explore the possibilities of further integrations of AI technologies in software development. Furthermore, questions related to the cost-effectiveness of LLMs in software documentation, such as return on

investment and the balance between upfront costs versus long-term gains in productivity, require thorough exploration. This could help organizations make informed decisions about adopting these technologies.

By confirming the hypotheses proposed, this study not only advances our understanding of the capabilities and benefits of LLMs in software development but also lays the groundwork for further research in this area, potentially setting the stage for widespread adoption of AI-driven tools in software engineering.

References

[1] I. Sommerville, *Software engineering*, 10th ed. Boston: Pearson, 2016.

[2] K.-B. Ooi, G. W.-H. Tan, M. Al-Emran, M. A. Al-Sharafi, A. Capatina, A. Chakraborty, Y. K. Dwivedi, T.-L. Huang, A. K. Kar, V.-H. Lee, X.-M. Loh, A. Micu, P. Mikalef, E. Mogaji, N. Pandey, R. Raman, N. P. Rana, P. Sarker, A. Sharma, C.-I. Teng, S. F. Wamba, and L.-W. Wong, "The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions," *Journal of Computer Information Systems*, vol. 0, no. 0, pp. 1–32, 2023.

[3] F. García-Peñalvo and A. Vázquez-Ingelmo, "What Do We Mean by GenAI? A Systematic Mapping of The Evolution, Trends, and Techniques Involved in Generative AI," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 8, no. 4, p. 7, 2023.

[4] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large Language Models for Software Engineering: A Systematic Literature Review," Sep. 2023, arXiv:2308.10620.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is All you Need," vol. 30, 2017.

[6] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, Jan. 2021.

[7] M. Aljanabi, M. G. Yaseen, A. H. Ali, and M. A. Mohammed, "Prompt Engineering: Guiding the Way to Effective Large Language Models," *Iraqi Journal for Computer Science and Mathematics*, vol. 4, no. 4, Nov. 2023.

[8] H. Ratnayake and C. Wang, "A Prompting Framework to Enhance Language Model Output," in *AI 2023: Advances in Artificial Intelligence*, T. Liu, G. Webb, L. Yue, and D. Wang, Eds. Singapore: Springer Nature, 2024, pp. 66–81.

[9] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 2024, arXiv:2312.10997.

[10] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," Feb. 2020, arXiv:2002.08909.

[11] M. Barenkamp, J. Rebstadt, and O. Thomas, "Applications of AI in classical software engineering," *AI Perspectives*, vol. 2, no. 1, p. 1, Dec. 2020.

[12] C. Ebert and P. Louridas, "Generative AI for Software Practitioners," *IEEE Software*, vol. 40, no. 4, pp. 30–38, Jul. 2023.

[13] Y. Su, C. Wan, U. Sethi, S. Lu, M. Musuvathi, and S. Nath, "HotGPT: How to Make Software Documentation More Useful with a Large Language Model?" in *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*. Providence RI USA: ACM, Jun. 2023, pp. 87–93.

[14] C. Qian, X. Cong, W. Liu, C. Yang, W. Chen, Y. Su, Y. Dang, J. Li, J. Xu, D. Li, Z. Liu, and M. Sun, "Communicative Agents for Software Development," Aug. 2023, arXiv:2307.07924.

[15] E. Aghajani, C. Nagy, O. L. Vega-Marquez, M. Linares-Vasquez, L. Moreno, G. Bavota, and M. Lanza, "Software Documentation Issues Unveiled," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. Montreal, QC, Canada: IEEE, May 2019, pp. 1199–1210.

[16] V. Chomal and J. Saini, "Significance of Software Documentation in Software Development Process," *International Journal of Engineering Innovation & Research*, vol. 3, pp. 410–416, Jul. 2014.

[17] C. J. Stettina and W. Heijstek, "Necessary and neglected?: an empirical study of internal documentation in agile software development teams," in *Proceedings of the 29th ACM international conference on Design of communication*. Pisa Italy: ACM, Oct. 2011, pp. 159–166.

[18] L. K. Kivinen, "AI-driven chatbot as a support tool for developers during the onboarding process," Master's thesis, Haaga-Helia University of Applied Science, 2023. [Online]. Available: http://www.theseus.fi/handle/10024/802817

[19] J. Buchan, S. G. MacDonell, and J. Yang, "Effective team onboarding in Agile software development: techniques and goals," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Sep. 2019, pp. 1–11, iSSN: 1949-3789.

[20] A. Imran and T. Kosar, "Software Sustainability: A Systematic Literature Review and Comprehensive Analysis," Oct. 2019, arXiv:1910.06109.

[21] S. Wagner and F. Deissenboeck, "Defining Productivity in Software Engineering," in *Rethinking Productivity in Software Engineering*, C. Sadowski and T. Zimmermann, Eds. Berkeley, CA: Springer Nature, 2019, pp. 29–38.

[22] C. Sadowski, M.-A. Storey, and R. Feldt, "A Software Development Productivity Framework," in *Rethinking Productivity in Software Engineering*, C. Sadowski and T. Zimmermann, Eds. Berkeley, CA: Springer Nature, 2019, pp. 39–47.

[23] L. Briand, "Software documentation: how much is enough?" in *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.* Benevento, Italy: IEEE Comput. Soc, 2003, pp. 13–15.

[24] T. Theunissen, U. van Heesch, and P. Avgeriou, "A mapping study on documentation in Continuous Software Development," *Information and Software Technology*, vol. 142, 2022.

[25] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in *Proceedings of the 2002 ACM symposium on Document engineering*. McLean Virginia USA: ACM, Nov. 2002, pp. 26–33.

[26] Y. Shmerlin, D. Kliger, and H. Makabee, "Reducing technical debt: Using persuasive technology for encouraging software developers to document code," *Lecture Notes in Business Information Processing*, vol. 178 LNBIP, pp. 207–212, 2014, iSBN: 9783319078687.

[27] L. Silva, M. Unterkalmsteiner, and K. Wnuk, "Towards identifying and minimizing customer-facing documentation debt," in *2023 ACM/IEEE International Conference on Technical Debt (TechDebt)*. Melbourne, Australia: IEEE, May 2023, pp. 72–81.

[28] S. Wu, Y. Zhou, and X. Wang, "Exploring User Experience of Automatic Documentation Tools," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. Yokohama Japan: ACM, May 2021, pp. 1–6.

[29] G. Matturro, K. Barrella, and P. Benitez, "Difficulties of Newcomers Joining Software Projects Already in Execution," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2017, pp. 993–998.

[30] K. A. de Graaf, P. Liang, A. Tang, and H. van Vliet, "The impact of prior knowledge on searching in software documentation," in *Proceedings of the 2014 ACM symposium on Document engineering*, ser. DocEng '14. New York, NY, USA: Association for Computing Machinery, Sep. 2014, pp. 189–198.

[31] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring Program Comprehension: A Large-Scale Field Study with Professionals," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 951–976, Oct. 2018.

[32] L. Wilsson, "Automating and increasing efficiency of component documentation maintenance: A case study."

[33] J. Dominic, C. Ritter, and P. Rodeghero, "Onboarding Bot for Newcomers to Software Engineering," in *Proceedings of the International Conference on Software and System Processes*. Seoul Republic of Korea: ACM, Jun. 2020, pp. 91–94.

[34] J. Dominic, J. Houser, I. Steinmacher, C. Ritter, and P. Rodeghero, "Conversational Bot for Newcomers Onboarding to Open Source Projects," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. Seoul Republic of Korea: ACM, Jun. 2020, pp. 46–50.

[35] A. Abdellatif, K. Badran, and E. Shihab, "MSRBot: Using bots to answer questions from software repositories," *Empirical Software Engineering*, vol. 25, no. 3, pp. 1834–1863, May 2020.

[36] G. Melo, E. Law, P. Alencar, and D. Cowan, "Understanding User Understanding: What do Developers Expect from a Cognitive Assistant?" in *2020 IEEE International Conference on Big Data (Big Data)*. Atlanta, GA, USA: IEEE, Dec. 2020, pp. 3165–3172.

[37] E. Aghajani, "Context-Aware Software Documentation," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Madrid: IEEE, Sep. 2018, pp. 727–731.

[38] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, "The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development," in *Proceedings of the 28th International Conference on Intelligent User Interfaces*. Sydney NSW Australia: ACM, Mar. 2023, pp. 491–514.

[39] A. Ziegler, E. Kalliamvakou, X. A. Li, A. Rice, D. Rifkin, S. Simister, G. Sittampalam, and E. Aftandilian, "Productivity assessment of neural code completion," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, ser. MAPS 2022. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 21–29.

[40] J. Y. Khan and G. Uddin, "Combining Contexts from Multiple Sources for Documentation-Specific Code Example Generation," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Taipa, Macao: IEEE, Mar. 2023, pp. 683–687.

[41] ——, "Automatic Code Documentation Generation Using GPT-3," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. Rochester MI USA: ACM, Oct. 2022, pp. 1–6.

[42] A. Y. Wang, D. Wang, J. Drozdal, M. Muller, S. Park, J. D. Weisz, X. Liu, L. Wu, and C. Dugan, "Documentation Matters: Human-Centered AI System to Assist Data Science Code Documentation in Computational Notebooks," *ACM Transactions on Computer-Human Interaction*, vol. 29, no. 2, pp. 1–33, Apr. 2022.

[43] S. Haque, A. LeClair, L. Wu, and C. McMillan, "Improved Automatic Summarization of Subroutines via Attention to File Context," in *Proceedings of the 17th International Conference on Mining Software Repositories*. Seoul Republic of Korea: ACM, Jun. 2020, pp. 300–310.

[44] F. Sovrano, K. Ashley, and A. Bacchelli, "Toward Eliminating Hallucinations: GPT-based Explanatory AI for Intelligent Textbooks and Documentation."

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experiment Process," in *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Science & Business Media, 2012, pp. 73–81.

[46] B. Kitchenham and S. L. Pfleeger, "Personal Opinion Surveys," in *Guide to advanced empirical software engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer, 2008, pp. 63–91.

[47] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22 140, pp. 55–55, 1932.

[48] "Stack Overflow Developer Survey 2023," accessed at Mars. 18, 2024. [Online]. Available: https://survey.stackoverflow.co/2023/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2023

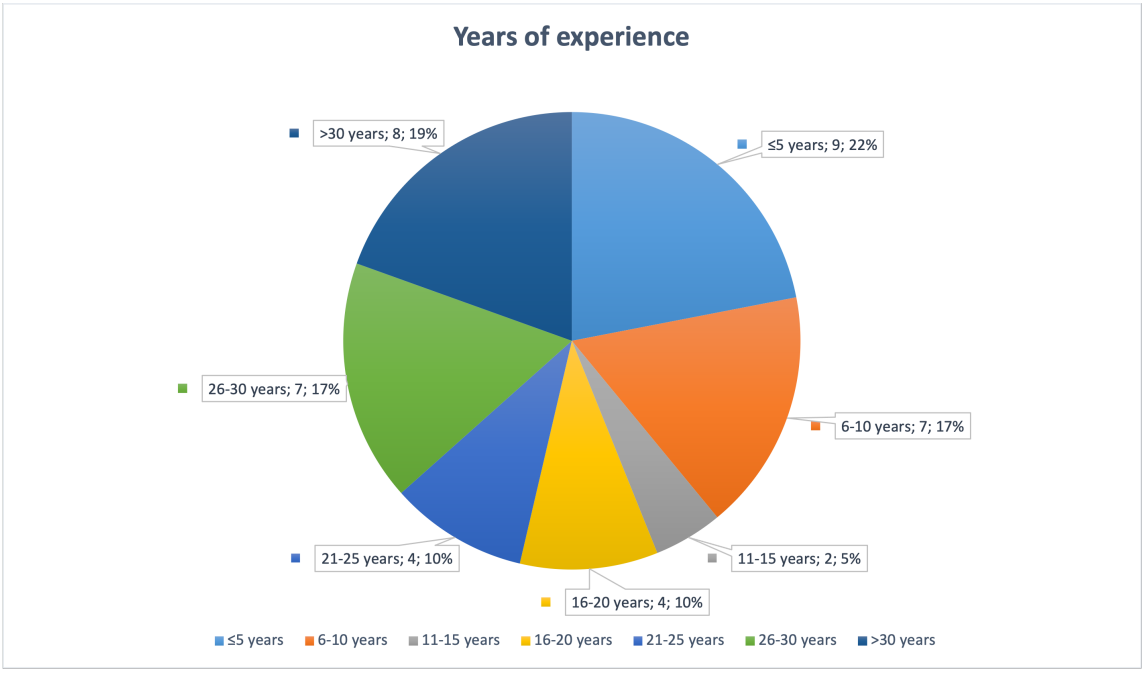
[49] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Planning," in *Experimentation in Software Engineering.* Berlin, Heidelberg: Springer Science & Business Media, 2012, pp. 73–81.

Appendix A

Supplemental Information

Figure A.1.

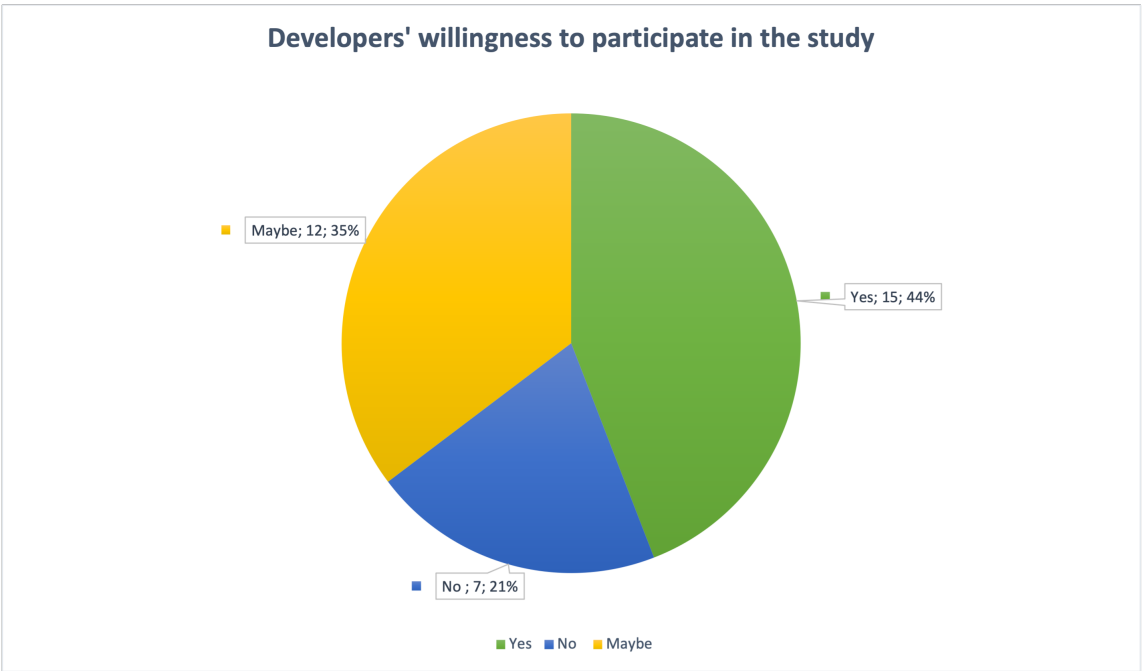
Developer Experience Levels



Note. Displayed in this figure are the experience levels of developers who participated in the pre-survey performed to examine the demographic characteristics of developers at Telefonaktiebolaget LM Ericsson.

Figure A.2.

Developers' willingness to participate in the study



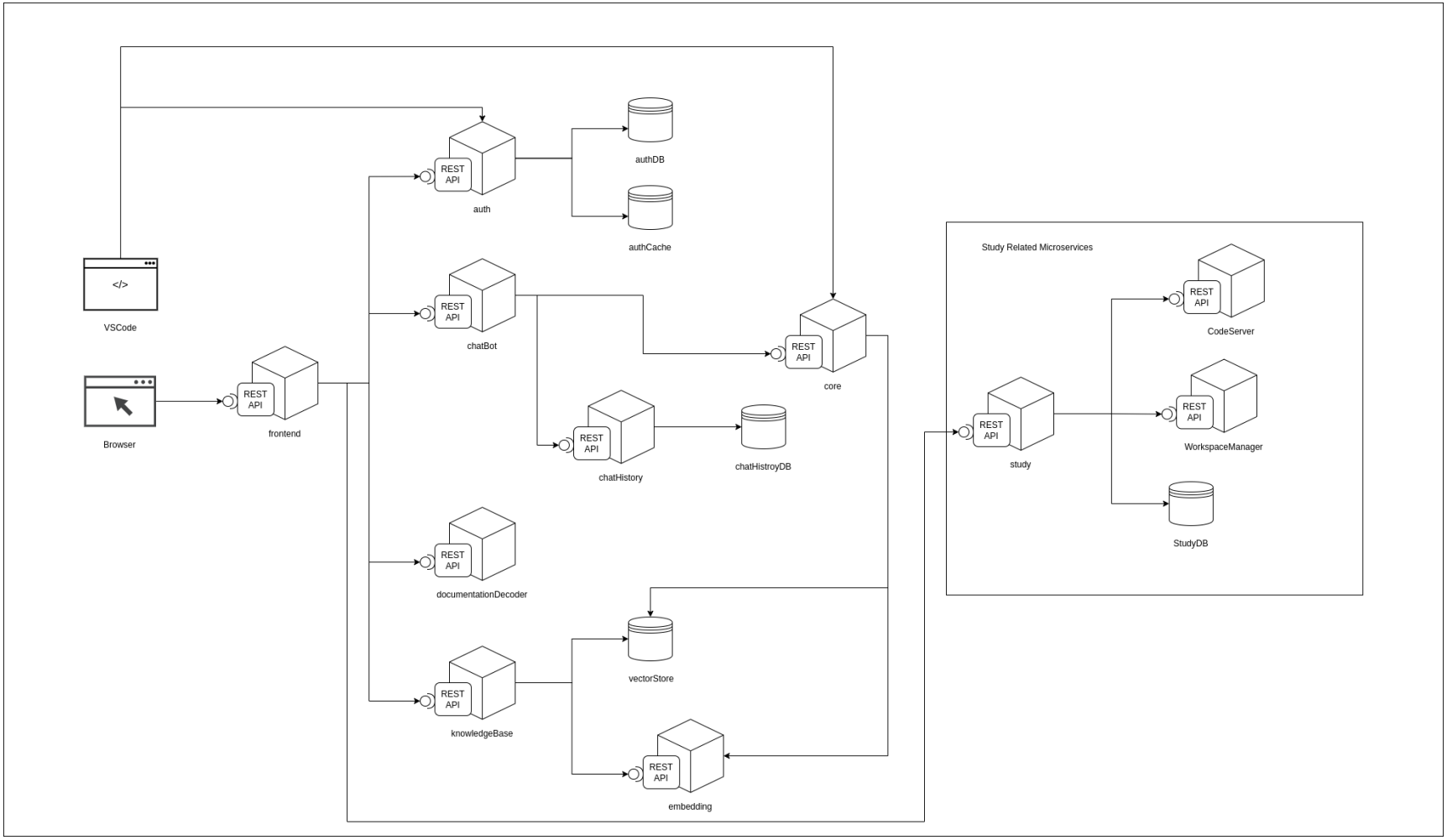
Note. This figure represents the willingness of developers who participated in the pre-survey to engage in the study.

Appendix B

System Components

Figure B.1.

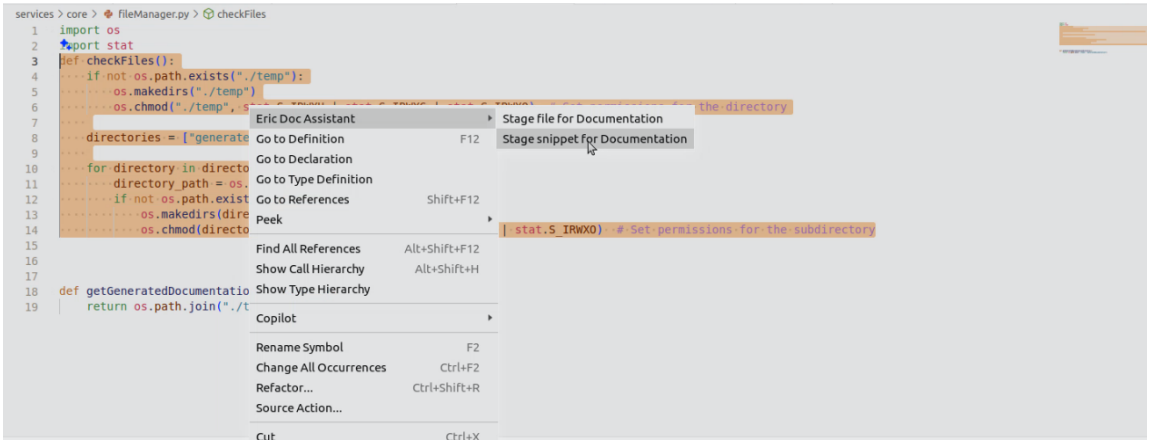
Architecture Overview of the System's Microservices



Note. This figure provides an architectural overview of the system developed for this study, illustrating the microservices as system components and detailing the interconnections among these components.

Figure B.2.

Staging a code snippet for documentation



Note. This figure shows the process of staging a code snippet for documentation in the implemented documentation generator.

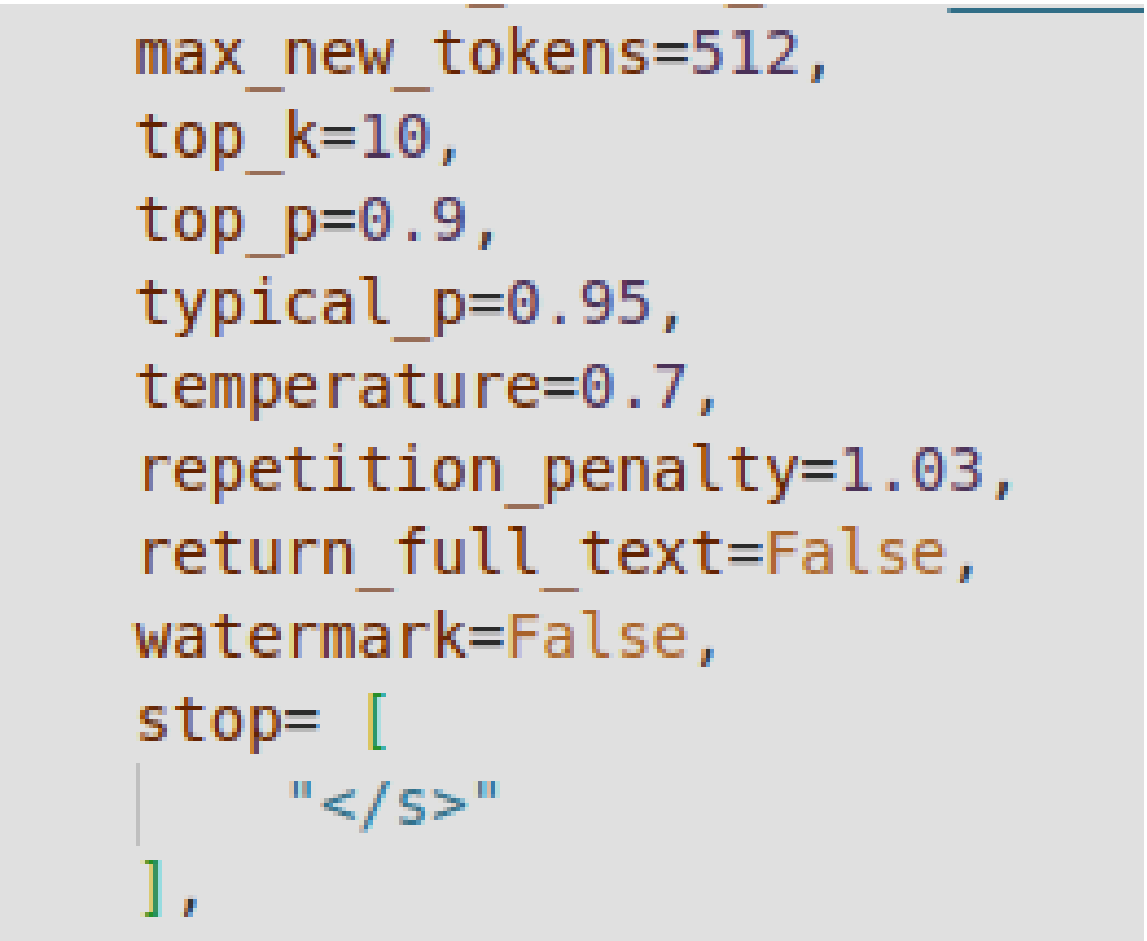
Figure B.3.

Staged code snippets window

Note. This figure shows a custom window in the implemented documentation generator in which the staged code for documentation is showed in.

Figure B.4.

LLM parameter configuration



```
max_new_tokens=512,
top_k=10,
top_p=0.9,
typical_p=0.95,
temperature=0.7,
repetition_penalty=1.03,
return_full_text=False,
watermark=False,
stop= [
    "</s>"
],
```

Note. This figure shows the parameters configuration used for the mistralai/Mixtral-8x7B-Instruct-v0.1 model.

Appendix C

Results

Table C.1

Programming task results.

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
		Experience ≤ 5			
6f7bebbc0b87	Manual	1	27	6	420
6108d268cf92	Manual	1	10	1	386
e1c3f6047de3	LLM	1	10	10	245
37c100e61b9f	LLM	1	17	17	240
bf2a59f9975d	Manual	2	17	11	288
c8d72c49e97b	Manual	2	10	9	420
2a3874a15c56	LLM	2	17	16	206
92d7088440cf	Manual	2	10	8	382
cca0a4439525	LLM	3	16	15	217
c70b82a89813	Manual	3	20	3	294
71fd688a3e93	Manual	3	17	0	420
0807045b421c	LLM	3	17	13	348
72d8891b0514	LLM	3	17	17	251
739f8cc0c7b4	LLM	4	10	10	307
8d4fc24e4ee6	Manual	4	11	10	391
ccc6512918cc	LLM	4	16	12	390
90e0b9f0263c	Manual	5	16	11	420
3b172ce10d3d	Manual	5	24	8	396
a8438a69a5a1	LLM	5	17	15	239

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
			$5 < Experience \leq 10$		
824061e0d50d	LLM	6	17	16	226
ed38e8c594d8	LLM	6	17	5	420
e807d09f553f	LLM	7	10	7	206
1b915b3d493c	LLM	7	17	14	300
4c8a2c0b6573	Manual	8	23	10	420
a387a3e6443f	Manual	8	11	1	355
fe62732ed5df	Manual	9	24	9	420
fc603662666c	LLM	9	25	17	420
5ac9c784ea39	Manual	9	19	13	420
38bf317558a7	Manual	10	20	12	420
			$6 < Experience \leq 15$		
f4b19dcd0828	Manual	11	17	4	420
f203f0e22e1d	LLM	11	17	15	320
35754d04a5ae	Manual	11	27	21	391
14a81b205773	LLM	11	17	17	208
f94a29cd4921	Manual	12	17	11	342
cc0521767afe	LLM	12	17	16	284
c049d125639b	Manual	13	9	5	395
c2ca9f6f682e	LLM	13	11	11	351
43ab770f9f01	Manual	13	10	9	390
a47d56db5a71	Manual	14	16	8	383
b727cc1aab59	LLM	14	16	14	186

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
b04f0f17e9ea	Manual	15	11	11	310
d21560690162	LLM	15	10	10	175
		$16 < Experience \leq 20$			
864002c6a0ad	LLM	16	17	8	420
be87d11b1f7a	LLM	16	22	19	309
1fd4ab5a1263	LLM	16	11	11	316
87aae6fd6ccf	Manual	17	7	12	318
18080cb00a91	LLM	18	10	10	184
03b706676536	Manual	19	17	9	321
f97339e44e9a	Manual	20	27	18	346
		$21 < Experience \leq 25$			
2a1891ca62bd	LLM	21	10	9	319
4ec3555870a8	LLM	22	19	13	420
c95f5372fe02	Manual	22	19	9	389
c9c51875a046	Manual	23	17	6	420
		$26 < Experience \leq 30$			
1d7933b2b5df	LLM	26	27	25	314

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
800db0baa67b	Manual	27	17	4	344
1f470084f7fe	LLM	27	17	16	216
f4ffe349c97b	LLM	28	21	21	416
64414c28876e	Manual	29	10	4	388
		$Experience > 30$			
4379a709aaf7	LLM	31	27	26	298

Note. This table presents the results of programming tasks completed by developers in the controlled experiment.

[a] The method used for comprehending the required documentation for completing a programming task. The available methods are a manual method or an LLM-assisted method.

[b] The total years of professional experience the developer has in the field of software engineering (measured in years).

[c] The total number of programming subtasks assigned to the developer during the experiment.

[d] The count of programming subtasks correctly solved by the developer during the experiment.

[e] The time taken by the developer to complete the programming tasks (measured in seconds).

Table C.2

Documentation task results.

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
		Experience ≤ 5			
6f7bebbc0b87	Manual	1	10	3	420
6108d268cf92	Manual	1	10	3	420
e1c3f6047de3	LLM	1	10	10	150
37c100e61b9f	LLM	1	10	8	156
bf2a59f9975d	Manual	2	10	6	420
c8d72c49e97b	Manual	2	10	3	420
2a3874a15c56	LLM	2	10	10	120
92d7088440cf	Manual	2	10	4	372
cca0a4439525	LLM	3	10	10	177
c70b82a89813	Manual	3	10	5	326
71fd688a3e93	Manual	3	10	3	420
0807045b421c	LLM	3	10	10	128
72d8891b0514	LLM	3	10	7	137
739f8cc0c7b4	LLM	4	10	10	162
8d4fc24e4ee6	Manual	4	10	7	416
ccc6512918cc	LLM	4	10	10	166
90e0b9f0263c	Manual	5	10	4	420
3b172ce10d3d	Manual	5	10	4	392
a8438a69a5a1	LLM	5	10	8	117

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
			$5 < Experience \leq 10$		
824061e0d50d	LLM	6	10	10	246
ed38e8c594d8	LLM	6	10	5	420
e807d09f553f	LLM	7	10	10	157
1b915b3d493c	LLM	7	10	9	308
4c8a2c0b6573	Manual	8	10	6	420
a387a3e6443f	Manual	8	10	4	346
fe62732ed5df	Manual	9	10	2	369
fc603662666c	LLM	9	10	6	126
5ac9c784ea39	Manual	9	10	1	411
38bf317558a7	Manual	10	10	5	349
			$10 < Experience \leq 15$		
f4b19dcd0828	Manual	11	10	3	420
f203f0e22e1d	LLM	11	10	9	150
35754d04a5ae	Manual	11	10	4	356
14a81b205773	LLM	11	10	8	127
f94a29cd4921	Manual	12	10	5	294
cc0521767afe	LLM	12	10	10	333
c049d125639b	Manual	13	10	4	420
c2ca9f6f682e	LLM	13	10	10	128
43ab770f9f01	Manual	13	10	5	420
a47d56db5a71	Manual	14	10	2	364
b727cc1aab59	LLM	14	10	9	158

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
b04f0f17e9ea	Manual	15	10	4	375
d21560690162	LLM	15	10	9	145
		$15 < Experience \leq 20$			
864002c6a0ad	LLM	16	10	8	374
be87d11b1f7a	LLM	16	10	9	347
1fd4ab5a1263	LLM	16	10	9	272
87aae6fd6ccf	Manual	17	10	6	348
18080cb00a91	LLM	18	10	9	158
03b706676536	Manual	19	10	3	338
f97339e44e9a	Manual	20	10	4	278
		$20 < Experience \leq 25$			
2a1891ca62bd	LLM	21	10	10	154
4ec3555870a8	LLM	22	10	10	173
c95f5372fe02	Manual	22	10	7	373
c9c51875a046	Manual	23	10	4	420
		$25 < Experience \leq 30$			
1d7933b2b5df	LLM	26	10	10	259

Continued on next page

Participant ID	Method[a]	Experience[b]	Total No. of Subtasks[c]	Correctly Completed Subtasks[d]	Time[e]
800db0baa67b	Manual	27	10	4	332
1f470084f7fe	LLM	27	10	9	92
f4ffe349c97b	LLM	28	10	7	142
64414c28876e	Manual	29	10	4	420
		$Experience > 30$			
4379a709aaf7	LLM	31	10	8	289

Note. This table presents the results of documentation tasks completed by developers in the controlled experiment.

[a] The method used for producing development documentation. The available methods are a manual method or an LLM-assisted method.

[b] The total years of professional experience the developer has in the field of software engineering (measured in years).

[c] The total number of documentation subtasks assigned to the developer during the experiment.

[d] The count of documentation subtasks correctly solved by the developer during the experiment.

[e] The time taken by the developer to complete the documentation tasks (measured in seconds).

Table C.3

Programming task velocity and effectiveness results.

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
		$Experience \leq 5$		
6f7bebbc0b87	manual	1	51.43	0.22
6108d268cf92	manual	1	9.33	0.10
e1c3f6047de3	llm	1	146.94	1.00
37c100e61b9f	llm	1	255.00	1.00
bf2a59f9975d	manual	2	137.50	0.65
c8d72c49e97b	manual	2	77.14	0.90
2a3874a15c56	llm	2	279.61	0.94
92d7088440cf	manual	2	75.39	0.80
cca0a4439525	llm	3	248.85	0.94
c70b82a89813	manual	3	36.73	0.15
71fd688a3e93	manual	3	0.00	0.00
0807045b421c	llm	3	134.48	0.76
72d8891b0514	llm	3	243.82	1.00
739f8cc0c7b4	llm	4	117.26	1.00
8d4fc24e4ee6	manual	4	92.07	0.91
ccc6512918cc	llm	4	110.77	0.75
90e0b9f0263c	manual	5	94.29	0.69
3b172ce10d3d	manual	5	72.73	0.33
a8438a69a5a1	llm	5	225.94	0.88
		$5 < Experience \leq 10$		
824061e0d50d	llm	6	254.87	0.94
ed38e8c594d8	llm	6	42.86	0.29
e807d09f553f	llm	7	122.33	0.70
1b915b3d493c	llm	7	168.00	0.82
4c8a2c0b6573	manual	8	85.71	0.43
a387a3e6443f	manual	8	10.14	0.09
fe62732ed5df	manual	9	77.14	0.38
fc603662666c	llm	9	145.71	0.68
5ac9c784ea39	manual	9	111.43	0.68
38bf317558a7	manual	10	102.86	0.60
		$10 < Experience \leq 15$		
f4b19dcd0828	manual	11	34.29	0.24

Continued on next page

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
f203f0e22e1d	llm	11	168.75	0.88
35754d04a5ae	manual	11	193.35	0.78
14a81b205773	llm	11	294.23	1.00
f94a29cd4921	manual	12	115.79	0.65
cc0521767afe	llm	12	202.82	0.94
c049d125639b	manual	13	45.57	0.56
c2ca9f6f682e	llm	13	112.82	1.00
43ab770f9f01	manual	13	83.08	0.90
a47d56db5a71	manual	14	75.20	0.50
b727cc1aab59	llm	14	270.97	0.88
b04f0f17e9ea	manual	15	127.74	1.00
d21560690162	llm	15	205.71	1.00
		$15 < Experience \leq 20$		
864002c6a0ad	llm	16	68.57	0.47
be87d11b1f7a	llm	16	221.36	0.86
1fd4ab5a1263	llm	16	125.32	1.00
87aae6fd6ccf	manual	17	79.25	0.44
18080cb00a91	llm	18	195.65	1.00
03b706676536	manual	19	100.93	0.53
f97339e44e9a	manual	20	187.283	0.67
		$20 < Experience \leq 25$		
2a1891ca62bd	llm	21	101.57	0.90
4ec3555870a8	llm	22	111.43	0.68
c95f5372fe02	manual	22	83.29	0.47
c9c51875a046	manual	23	51.43	0.35
		$25 < Experience \leq 30$		
1d7933b2b5df	llm	26	286.62	0.93
800db0baa67b	manual	27	41.86	0.24
1f470084f7fe	llm	27	266.67	0.94
f4ffe349c97b	llm	28	181.73	1.00
64414c28876e	manual	29	37.11	0.40
		$Experience > 30$		

Continued on next page

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
4379a709aaf7	llm	31	314.09	0.96

Note. This table presents the velocity and effectiveness of developers for programming tasks performed during the experiment.
[a] The method used for comprehending the required documentation: either manual or LLM-assisted.
[b] Professional experience of the participant in years.
[c] Velocity of completing tasks measured as tasks per hour.
[d] Effectiveness in completing tasks correctly (scale 0-1).

Table C.4

Documentation task velocity and effectiveness results.

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
		$Experience \leq 5$		
6f7bebbc0b87	manual	1	25.71	0.3
6108d268cf92	manual	1	25.71	0.3
e1c3f6047de3	llm	1	240.00	1.0
37c100e61b9f	llm	1	184.62	0.8
bf2a59f9975d	manual	2	51.43	0.6
c8d72c49e97b	manual	2	25.71	0.3
2a3874a15c56	llm	2	300.00	1.0
92d7088440cf	manual	2	38.71	0.4
cca0a4439525	llm	3	203.39	1.0
c70b82a89813	manual	3	55.21	0.5
71fd688a3e93	manual	3	25.71	0.3
0807045b421c	llm	3	281.25	1.0
72d8891b0514	llm	3	183.94	0.7
739f8cc0c7b4	llm	4	222.22	1.0
8d4fc24e4ee6	manual	4	60.58	0.7
ccc6512918cc	llm	4	216.87	1.0
90e0b9f0263c	manual	5	34.29	0.4
3b172ce10d3d	manual	5	36.73	0.4
a8438a69a5a1	llm	5	246.15	0.8
		$5 < Experience \leq 10$		
824061e0d50d	llm	6	146.34	1.0
ed38e8c594d8	llm	6	42.86	0.5
e807d09f553f	llm	7	229.30	1.0
1b915b3d493c	llm	7	105.19	0.9
4c8a2c0b6573	manual	8	51.43	0.6
a387a3e6443f	manual	8	41.62	0.4
fe62732ed5df	manual	9	19.51	0.2
fc603662666c	llm	9	171.43	0.6
5ac9c784ea39	manual	9	8.76	0.1
38bf317558a7	manual	10	51.58	0.5
		$10 < Experience \leq 15$		
f4b19dcd0828	manual	11	25.71	0.3

Continued on next page

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
f203f0e22e1d	llm	11	216.00	0.9
35754d04a5ae	manual	11	40.45	0.4
14a81b205773	llm	11	226.77	0.8
f94a29cd4921	manual	12	61.22	0.5
cc0521767afe	llm	12	108.11	1.0
c049d125639b	manual	13	34.29	0.4
c2ca9f6f682e	llm	13	281.25	1.0
43ab770f9f01	manual	13	42.86	0.5
a47d56db5a71	manual	14	19.78	0.2
b727cc1aab59	llm	14	205.06	0.9
b04f0f17e9ea	manual	15	38.40	0.4
d21560690162	llm	15	223.45	0.9
		$15 < Experience \leq 20$		
864002c6a0ad	llm	16	77.01	0.8
be87d11b1f7a	llm	16	93.37	0.9
1fd4ab5a1263	llm	16	119.12	0.9
87aae6fd6ccf	manual	17	62.07	0.6
18080cb00a91	llm	18	205.06	0.9
03b706676536	manual	19	31.95	0.3
f97339e44e9a	manual	20	51.80	0.4
		$20 < Experience \leq 25$		
2a1891ca62bd	llm	21	233.77	1.0
4ec3555870a8	llm	22	208.09	1.0
c95f5372fe02	manual	22	67.56	0.7
c9c51875a046	manual	23	34.29	0.4
		$25 < Experience \leq 30$		
1d7933b2b5df	llm	26	139.00	1.0
800db0baa67b	manual	27	43.37	0.4
1f470084f7fe	llm	27	352.17	0.9
f4ffe349c97b	llm	28	177.46	0.7
64414c28876e	manual	29	34.29	0.4
		$Experience > 30$		

Continued on next page

Participant ID	Method[a]	Experience[b]	Velocity[c]	Effectiveness[d]
4379a709aaf7	llm	31	99.65	0.8

Note. This table presents the velocity and effectiveness of developers for documentation tasks performed during the experiment.

[a] The method used for producing documentation: either manual or LLM-assisted.

[b] Professional experience of the participant in years.

[c] Velocity of completing tasks measured as tasks per hour.

[d] Effectiveness in completing tasks correctly (scale 0-1).

Table C.5

Documentation quality evaluation results.

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
6f7bebbc0b87	llm	5	3	5	13
6f7bebbc0b87	manual	3	2	1	6
6f7bebbc0b87	llm	5	5	5	15
6f7bebbc0b87	manual	5	5	5	15
6f7bebbc0b87	llm	4	5	5	14
6f7bebbc0b87	manual	4	4	3	11
6f7bebbc0b87	llm	5	5	5	15
6f7bebbc0b87	manual	4	4	3	11
6f7bebbc0b87	llm	5	5	5	15
6f7bebbc0b87	manual	4	4	3	11
6108d268cf92	llm	5	5	5	15
6108d268cf92	manual	5	4	2	11
6108d268cf92	llm	5	4	5	14
6108d268cf92	manual	5	2	5	12
6108d268cf92	llm	5	4	5	14
6108d268cf92	manual	5	4	4	13
6108d268cf92	llm	5	5	5	15
6108d268cf92	manual	5	5	5	15
6108d268cf92	llm	5	5	5	15
6108d268cf92	manual	5	3	5	13
e1c3f6047de3	llm	4	5	4	13
e1c3f6047de3	manual	4	4	3	11

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
e1c3f6047de3	llm	5	4	4	13
e1c3f6047de3	manual	5	5	5	15
e1c3f6047de3	llm	4	5	4	13
e1c3f6047de3	manual	5	5	4	14
e1c3f6047de3	llm	4	3	3	10
e1c3f6047de3	manual	5	5	4	14
e1c3f6047de3	llm	5	4	5	14
e1c3f6047de3	manual	4	3	2	9
37c100e61b9f	llm	4	5	5	14
37c100e61b9f	manual	5	3	4	12
37c100e61b9f	llm	3	3	4	10
37c100e61b9f	manual	4	4	4	12
37c100e61b9f	llm	4	4	5	13
37c100e61b9f	manual	4	4	3	11
37c100e61b9f	llm	4	5	5	14
37c100e61b9f	manual	4	3	3	10
37c100e61b9f	llm	4	4	5	13
37c100e61b9f	manual	3	3	2	8
bf2a59f9975d	llm	3	4	5	12
bf2a59f9975d	manual	5	4	3	12
bf2a59f9975d	llm	5	5	5	15
bf2a59f9975d	manual	5	4	3	12
bf2a59f9975d	llm	5	3	5	13
bf2a59f9975d	manual	5	2	1	8
bf2a59f9975d	llm	5	5	5	15
bf2a59f9975d	manual	5	5	5	15
bf2a59f9975d	llm	4	5	5	14

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
bf2a59f9975d	manual	5	3	4	12
c8d72c49e97b	llm	5	5	5	15
c8d72c49e97b	manual	4	4	3	11
c8d72c49e97b	llm	5	5	5	15
c8d72c49e97b	manual	5	5	4	14
c8d72c49e97b	llm	5	4	5	14
c8d72c49e97b	manual	5	4	4	13
c8d72c49e97b	llm	5	5	5	15
c8d72c49e97b	manual	5	4	2	11
c8d72c49e97b	llm	3	4	5	12
c8d72c49e97b	manual	5	5	3	13
2a3874a15c56	llm	5	4	4	13
2a3874a15c56	manual	4	3	3	10
2a3874a15c56	llm	2	3	3	8
2a3874a15c56	manual	4	3	3	10
2a3874a15c56	llm	4	5	5	14
2a3874a15c56	manual	5	3	4	12
2a3874a15c56	llm	4	5	5	14
2a3874a15c56	manual	3	3	1	7
2a3874a15c56	llm	4	4	4	12
2a3874a15c56	manual	3	3	4	10
92d7088440cf	llm	5	5	5	15
92d7088440cf	manual	5	4	4	13
92d7088440cf	llm	5	5	5	15
92d7088440cf	manual	5	4	4	13
92d7088440cf	llm	5	5	5	15
92d7088440cf	manual	5	5	4	14

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
92d7088440cf	llm	5	5	5	15
92d7088440cf	manual	5	5	5	15
92d7088440cf	llm	5	5	5	15
92d7088440cf	manual	5	5	3	13
cca0a4439525	llm	3	4	5	12
cca0a4439525	manual	3	4	1	8
cca0a4439525	llm	5	5	5	15
cca0a4439525	manual	4	4	2	10
cca0a4439525	llm	4	5	5	14
cca0a4439525	manual	3	1	1	5
cca0a4439525	llm	5	4	5	14
cca0a4439525	manual	5	5	4	14
cca0a4439525	llm	5	4	4	13
cca0a4439525	manual	3	3	2	8
c70b82a89813	llm	3	5	4	12
c70b82a89813	manual	5	3	3	11
c70b82a89813	llm	4	5	5	14
c70b82a89813	manual	5	4	5	14
c70b82a89813	llm	4	5	5	14
c70b82a89813	manual	5	5	4	14
c70b82a89813	llm	5	5	5	15
c70b82a89813	manual	4	3	4	11
c70b82a89813	llm	4	5	5	14
c70b82a89813	manual	4	5	5	14
71fd688a3e93	llm	4	4	5	13
71fd688a3e93	manual	5	3	3	11
71fd688a3e93	llm	4	3	5	12

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
71fd688a3e93	manual	5	4	3	12
71fd688a3e93	llm	5	5	5	15
71fd688a3e93	manual	5	4	2	11
71fd688a3e93	llm	4	5	5	14
71fd688a3e93	manual	5	3	4	12
71fd688a3e93	llm	5	5	5	15
71fd688a3e93	manual	4	4	5	13
0807045b421c	llm	4	4	5	13
0807045b421c	manual	5	5	4	14
0807045b421c	llm	4	4	5	13
0807045b421c	manual	3	4	2	9
0807045b421c	llm	5	4	5	14
0807045b421c	manual	4	3	3	10
0807045b421c	llm	5	4	4	13
0807045b421c	manual	5	5	4	14
0807045b421c	llm	5	5	4	14
0807045b421c	manual	4	3	4	11
72d8891b0514	llm	4	5	5	14
72d8891b0514	manual	5	4	3	12
72d8891b0514	llm	4	4	5	13
72d8891b0514	manual	5	4	3	12
72d8891b0514	llm	5	4	5	14
72d8891b0514	manual	4	2	2	8
72d8891b0514	llm	5	4	5	14
72d8891b0514	manual	4	4	4	12
72d8891b0514	llm	3	5	4	12
72d8891b0514	manual	5	5	4	14

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
739f8cc0c7b4	llm	4	5	5	14
739f8cc0c7b4	manual	5	4	3	12
739f8cc0c7b4	llm	4	4	4	12
739f8cc0c7b4	manual	4	2	2	8
739f8cc0c7b4	llm	4	5	5	14
739f8cc0c7b4	manual	5	3	2	10
739f8cc0c7b4	llm	5	5	5	15
739f8cc0c7b4	manual	4	4	3	11
739f8cc0c7b4	llm	4	5	4	13
739f8cc0c7b4	manual	5	3	3	11
8d4fc24e4ee6	llm	5	4	5	14
8d4fc24e4ee6	manual	4	3	3	10
8d4fc24e4ee6	llm	4	5	5	14
8d4fc24e4ee6	manual	4	5	4	13
8d4fc24e4ee6	llm	2	4	5	11
8d4fc24e4ee6	manual	5	4	3	12
8d4fc24e4ee6	llm	5	5	5	15
8d4fc24e4ee6	manual	3	5	3	11
8d4fc24e4ee6	llm	5	5	5	15
8d4fc24e4ee6	manual	5	5	5	15
ccc6512918cc	llm	4	4	5	13
ccc6512918cc	manual	4	3	3	10
ccc6512918cc	llm	4	4	4	12
ccc6512918cc	manual	5	4	3	12
ccc6512918cc	llm	4	5	5	14
ccc6512918cc	manual	5	4	4	13
ccc6512918cc	llm	4	3	4	11

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
ccc6512918cc	manual	5	4	3	12
ccc6512918cc	llm	4	5	4	13
ccc6512918cc	manual	3	3	3	9
90e0b9f0263c	llm	5	5	5	15
90e0b9f0263c	manual	5	4	3	12
90e0b9f0263c	llm	4	5	5	14
90e0b9f0263c	manual	3	3	2	8
90e0b9f0263c	llm	5	5	5	15
90e0b9f0263c	manual	4	5	3	12
90e0b9f0263c	llm	4	5	5	14
90e0b9f0263c	manual	2	3	2	7
90e0b9f0263c	llm	5	5	5	15
90e0b9f0263c	manual	3	2	1	6
3b172ce10d3d	llm	5	5	5	15
3b172ce10d3d	manual	5	2	5	12
3b172ce10d3d	llm	3	4	5	12
3b172ce10d3d	manual	3	3	4	10
3b172ce10d3d	llm	4	4	5	13
3b172ce10d3d	manual	3	4	1	8
3b172ce10d3d	llm	5	5	5	15
3b172ce10d3d	manual	5	3	4	12
3b172ce10d3d	llm	5	5	5	15
3b172ce10d3d	manual	5	4	5	14
a8438a69a5a1	llm	5	5	5	15
a8438a69a5a1	manual	5	3	3	11
a8438a69a5a1	llm	3	3	5	11
a8438a69a5a1	manual	5	3	3	11

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
a8438a69a5a1	llm	5	5	3	13
a8438a69a5a1	manual	5	5	5	15
a8438a69a5a1	llm	3	3	4	10
a8438a69a5a1	manual	4	5	5	14
a8438a69a5a1	llm	4	4	4	12
a8438a69a5a1	manual	4	5	4	13
824061e0d50d	llm	5	5	5	15
824061e0d50d	manual	4	3	2	9
824061e0d50d	llm	4	4	5	13
824061e0d50d	manual	4	2	2	8
824061e0d50d	llm	4	5	5	14
824061e0d50d	manual	2	1	1	4
824061e0d50d	llm	4	5	5	14
824061e0d50d	manual	3	3	4	10
824061e0d50d	llm	4	5	5	14
824061e0d50d	manual	2	1	2	5
ed38e8c594d8	llm	4	3	4	11
ed38e8c594d8	manual	3	2	2	7
ed38e8c594d8	llm	4	4	4	12
ed38e8c594d8	manual	3	2	2	7
ed38e8c594d8	llm	4	3	4	11
ed38e8c594d8	manual	4	3	3	10
ed38e8c594d8	llm	4	3	4	11
ed38e8c594d8	manual	3	2	2	7
ed38e8c594d8	llm	4	3	4	11
ed38e8c594d8	manual	3	2	1	6
e807d09f553f	llm	5	4	5	14

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
e807d09f553f	manual	4	4	3	11
e807d09f553f	llm	5	5	5	15
e807d09f553f	manual	2	3	2	7
e807d09f553f	llm	4	4	5	13
e807d09f553f	manual	4	2	2	8
e807d09f553f	llm	5	4	5	14
e807d09f553f	manual	4	4	4	12
e807d09f553f	llm	4	5	4	13
e807d09f553f	manual	5	5	4	14
1b915b3d493c	llm	4	4	5	13
1b915b3d493c	manual	3	3	3	9
1b915b3d493c	llm	4	5	5	14
1b915b3d493c	manual	3	3	4	10
1b915b3d493c	llm	5	5	5	15
1b915b3d493c	manual	5	5	5	15
1b915b3d493c	llm	5	5	4	14
1b915b3d493c	manual	3	3	4	10
1b915b3d493c	llm	5	5	5	15
1b915b3d493c	manual	4	5	5	14
4c8a2c0b6573	llm	4	4	5	13
4c8a2c0b6573	manual	4	3	2	9
4c8a2c0b6573	llm	5	5	5	15
4c8a2c0b6573	manual	4	3	5	12
4c8a2c0b6573	llm	4	5	5	14
4c8a2c0b6573	manual	4	4	4	12
4c8a2c0b6573	llm	4	5	5	14
4c8a2c0b6573	manual	5	3	2	10

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
4c8a2c0b6573	llm	5	5	5	15
4c8a2c0b6573	manual	5	4	5	14
a387a3e6443f	llm	5	5	5	15
a387a3e6443f	manual	5	4	3	12
a387a3e6443f	llm	5	4	5	14
a387a3e6443f	manual	5	4	3	12
a387a3e6443f	llm	4	5	5	14
a387a3e6443f	manual	5	5	5	15
a387a3e6443f	llm	3	5	5	13
a387a3e6443f	manual	5	5	4	14
a387a3e6443f	llm	5	5	5	15
a387a3e6443f	manual	5	5	5	15
fe62732ed5df	llm	5	5	5	15
fe62732ed5df	manual	5	5	4	14
fe62732ed5df	llm	5	4	5	14
fe62732ed5df	manual	5	4	4	13
fe62732ed5df	llm	5	3	5	13
fe62732ed5df	manual	3	3	2	8
fe62732ed5df	llm	5	5	5	15
fe62732ed5df	manual	5	5	4	14
fe62732ed5df	llm	4	5	5	14
fe62732ed5df	manual	5	5	3	13
fc603662666c	llm	5	5	5	15
fc603662666c	manual	5	4	5	14
fc603662666c	llm	5	5	4	14
fc603662666c	manual	4	5	5	14
fc603662666c	llm	4	4	4	12

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
fc603662666c	manual	4	4	4	12
fc603662666c	llm	4	5	5	14
fc603662666c	manual	3	2	4	9
fc603662666c	llm	4	4	4	12
fc603662666c	manual	5	5	4	14
5ac9c784ea39	llm	5	5	5	15
5ac9c784ea39	manual	5	5	3	13
5ac9c784ea39	llm	4	5	5	14
5ac9c784ea39	manual	5	5	4	14
5ac9c784ea39	llm	5	5	5	15
5ac9c784ea39	manual	3	4	5	12
5ac9c784ea39	llm	3	5	5	13
5ac9c784ea39	manual	5	5	3	13
5ac9c784ea39	llm	3	4	5	12
5ac9c784ea39	manual	5	5	3	13
38bf317558a7	llm	5	5	5	15
38bf317558a7	manual	5	5	4	14
38bf317558a7	llm	5	5	5	15
38bf317558a7	manual	5	5	5	15
38bf317558a7	llm	5	4	4	13
38bf317558a7	manual	5	3	4	12
38bf317558a7	llm	5	5	5	15
38bf317558a7	manual	5	4	3	12
38bf317558a7	llm	3	4	4	11
38bf317558a7	manual	5	4	3	12
f4b19dcd0828	llm	5	5	5	15
f4b19dcd0828	manual	4	4	5	13

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
f4b19dcd0828	llm	4	5	5	14
f4b19dcd0828	manual	4	4	5	13
f4b19dcd0828	llm	5	5	5	15
f4b19dcd0828	manual	4	3	3	10
f4b19dcd0828	llm	5	5	5	15
f4b19dcd0828	manual	4	3	3	10
f4b19dcd0828	llm	4	4	5	13
f4b19dcd0828	manual	4	5	3	12
f203f0e22e1d	llm	4	4	5	13
f203f0e22e1d	manual	4	5	4	13
f203f0e22e1d	llm	3	3	5	11
f203f0e22e1d	manual	4	3	2	9
f203f0e22e1d	llm	4	5	5	14
f203f0e22e1d	manual	4	2	1	7
f203f0e22e1d	llm	5	5	5	15
f203f0e22e1d	manual	5	3	3	11
f203f0e22e1d	llm	4	5	4	13
f203f0e22e1d	manual	4	5	4	13
35754d04a5ae	llm	5	5	5	15
35754d04a5ae	manual	5	5	4	14
35754d04a5ae	llm	5	5	4	14
35754d04a5ae	manual	5	5	5	15
35754d04a5ae	llm	4	5	5	14
35754d04a5ae	manual	5	5	4	14
35754d04a5ae	llm	5	5	5	15
35754d04a5ae	manual	5	4	3	12
35754d04a5ae	llm	2	4	4	10

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
35754d04a5ae	manual	5	5	4	14
14a81b205773	llm	4	4	5	13
14a81b205773	manual	4	3	3	10
14a81b205773	llm	4	3	4	11
14a81b205773	manual	3	3	4	10
14a81b205773	llm	4	5	5	14
14a81b205773	manual	4	4	4	12
14a81b205773	llm	5	4	4	13
14a81b205773	manual	3	4	4	11
14a81b205773	llm	4	4	4	12
14a81b205773	manual	4	4	4	12
f94a29cd4921	llm	3	4	5	12
f94a29cd4921	manual	5	4	3	12
f94a29cd4921	llm	4	5	5	14
f94a29cd4921	manual	5	5	4	14
f94a29cd4921	llm	5	5	5	15
f94a29cd4921	manual	5	4	3	12
f94a29cd4921	llm	5	4	5	14
f94a29cd4921	manual	5	3	4	12
f94a29cd4921	llm	5	5	5	15
f94a29cd4921	manual	4	4	5	13
cc0521767afe	llm	4	5	5	14
cc0521767afe	manual	3	5	4	12
cc0521767afe	llm	3	3	4	10
cc0521767afe	manual	4	3	3	10
cc0521767afe	llm	4	4	4	12
cc0521767afe	manual	4	4	4	12

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
cc0521767afe	llm	5	4	5	14
cc0521767afe	manual	3	4	2	9
cc0521767afe	llm	5	5	5	15
cc0521767afe	manual	4	4	4	12
c049d125639b	llm	5	4	5	14
c049d125639b	manual	5	3	4	12
c049d125639b	llm	5	5	5	15
c049d125639b	manual	5	4	5	14
c049d125639b	llm	5	5	5	15
c049d125639b	manual	5	5	5	15
c049d125639b	llm	5	5	5	15
c049d125639b	manual	4	5	5	14
c049d125639b	llm	4	3	5	12
c049d125639b	manual	5	5	5	15
c2ca9f6f682e	llm	4	4	5	13
c2ca9f6f682e	manual	4	2	2	8
c2ca9f6f682e	llm	4	5	5	14
c2ca9f6f682e	manual	5	4	5	14
c2ca9f6f682e	llm	5	5	4	14
c2ca9f6f682e	manual	4	4	3	11
c2ca9f6f682e	llm	5	5	5	15
c2ca9f6f682e	manual	4	4	4	12
c2ca9f6f682e	llm	3	3	4	10
c2ca9f6f682e	manual	4	4	3	11
43ab770f9f01	llm	3	4	5	12
43ab770f9f01	manual	5	5	3	13
43ab770f9f01	llm	5	5	5	15

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
43ab770f9f01	manual	5	5	4	14
43ab770f9f01	llm	5	5	5	15
43ab770f9f01	manual	5	5	4	14
43ab770f9f01	llm	5	5	5	15
43ab770f9f01	manual	5	5	5	15
43ab770f9f01	llm	5	5	5	15
43ab770f9f01	manual	5	5	4	14
a47d56db5a71	llm	5	5	5	15
a47d56db5a71	manual	5	5	3	13
a47d56db5a71	llm	5	4	5	14
a47d56db5a71	manual	5	4	4	13
a47d56db5a71	llm	5	5	5	15
a47d56db5a71	manual	5	4	5	14
a47d56db5a71	llm	4	5	5	14
a47d56db5a71	manual	5	5	4	14
a47d56db5a71	llm	5	5	5	15
a47d56db5a71	manual	5	5	3	13
b727cc1aab59	llm	4	4	5	13
b727cc1aab59	manual	3	3	3	9
b727cc1aab59	llm	4	4	5	13
b727cc1aab59	manual	4	3	4	11
b727cc1aab59	llm	5	5	5	15
b727cc1aab59	manual	5	4	5	14
b727cc1aab59	llm	4	4	5	13
b727cc1aab59	manual	4	4	4	12
b727cc1aab59	llm	5	5	5	15
b727cc1aab59	manual	5	5	5	15

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
b04f0f17e9ea	llm	5	5	5	15
b04f0f17e9ea	manual	5	5	4	14
b04f0f17e9ea	llm	4	5	5	14
b04f0f17e9ea	manual	4	4	3	11
b04f0f17e9ea	llm	3	4	5	12
b04f0f17e9ea	manual	5	4	4	13
b04f0f17e9ea	llm	5	5	5	15
b04f0f17e9ea	manual	5	5	4	14
b04f0f17e9ea	llm	5	5	5	15
b04f0f17e9ea	manual	5	5	3	13
d21560690162	llm	4	5	5	14
d21560690162	manual	4	4	4	12
d21560690162	llm	3	2	2	7
d21560690162	manual	2	2	3	7
d21560690162	llm	5	5	5	15
d21560690162	manual	5	5	5	15
d21560690162	llm	5	4	5	14
d21560690162	manual	3	3	4	10
d21560690162	llm	5	3	4	12
d21560690162	manual	3	3	4	10
864002c6a0ad	llm	2	3	5	10
864002c6a0ad	manual	5	5	3	13
864002c6a0ad	llm	2	2	5	9
864002c6a0ad	manual	5	5	5	15
864002c6a0ad	llm	2	2	5	9
864002c6a0ad	manual	4	4	2	10
864002c6a0ad	llm	4	4	5	13

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
864002c6a0ad	manual	5	5	2	12
864002c6a0ad	llm	1	1	5	7
864002c6a0ad	manual	5	5	2	12
be87d11b1f7a	llm	4	4	5	13
be87d11b1f7a	manual	3	4	4	11
be87d11b1f7a	llm	5	5	4	14
be87d11b1f7a	manual	4	4	5	13
be87d11b1f7a	llm	5	5	5	15
be87d11b1f7a	manual	5	4	5	14
be87d11b1f7a	llm	4	4	4	12
be87d11b1f7a	manual	4	4	4	12
be87d11b1f7a	llm	4	4	4	12
be87d11b1f7a	manual	3	4	4	11
1fd4ab5a1263	llm	5	5	4	14
1fd4ab5a1263	manual	3	3	4	10
1fd4ab5a1263	llm	4	4	4	12
1fd4ab5a1263	manual	3	2	3	8
1fd4ab5a1263	llm	4	4	4	12
1fd4ab5a1263	manual	4	4	4	12
1fd4ab5a1263	llm	5	5	4	14
1fd4ab5a1263	manual	4	4	5	13
1fd4ab5a1263	llm	5	5	5	15
1fd4ab5a1263	manual	4	4	4	12
87aae6fd6ccf	llm	5	5	5	15
87aae6fd6ccf	manual	5	5	4	14
87aae6fd6ccf	llm	5	5	5	15
87aae6fd6ccf	manual	5	4	3	12

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
87aae6fd6ccf	llm	5	5	5	15
87aae6fd6ccf	manual	5	5	5	15
87aae6fd6ccf	llm	5	5	5	15
87aae6fd6ccf	manual	4	4	5	13
87aae6fd6ccf	llm	5	5	5	15
87aae6fd6ccf	manual	5	5	3	13
18080cb00a91	llm	4	4	5	13
18080cb00a91	manual	3	3	4	10
18080cb00a91	llm	4	4	5	13
18080cb00a91	manual	3	3	2	8
18080cb00a91	llm	4	4	4	12
18080cb00a91	manual	5	5	3	13
18080cb00a91	llm	5	4	5	14
18080cb00a91	manual	5	3	4	12
18080cb00a91	llm	4	4	5	13
18080cb00a91	manual	5	4	2	11
03b706676536	llm	2	4	5	11
03b706676536	manual	5	4	3	12
03b706676536	llm	5	5	5	15
03b706676536	manual	5	5	4	14
03b706676536	llm	5	5	5	15
03b706676536	manual	5	5	3	13
03b706676536	llm	2	4	5	11
03b706676536	manual	5	3	3	11
03b706676536	llm	5	5	5	15
03b706676536	manual	5	5	4	14
f97339e44e9a	llm	5	5	5	15

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
f97339e44e9a	manual	4	4	4	12
f97339e44e9a	llm	3	3	5	11
f97339e44e9a	manual	5	5	4	14
f97339e44e9a	llm	4	5	5	14
f97339e44e9a	manual	5	4	4	13
f97339e44e9a	llm	5	5	5	15
f97339e44e9a	manual	5	5	4	14
f97339e44e9a	llm	5	5	5	15
f97339e44e9a	manual	5	5	5	15
2a1891ca62bd	llm	5	4	5	14
2a1891ca62bd	manual	4	3	3	10
2a1891ca62bd	llm	4	4	4	12
2a1891ca62bd	manual	4	4	4	12
2a1891ca62bd	llm	5	4	4	13
2a1891ca62bd	manual	4	4	5	13
2a1891ca62bd	llm	4	4	5	13
2a1891ca62bd	manual	3	2	2	7
2a1891ca62bd	llm	5	5	4	14
2a1891ca62bd	manual	4	4	3	11
4ec3555870a8	llm	4	5	5	14
4ec3555870a8	manual	3	2	3	8
4ec3555870a8	llm	5	5	5	15
4ec3555870a8	manual	4	3	1	8
4ec3555870a8	llm	5	5	5	15
4ec3555870a8	manual	4	4	2	10
4ec3555870a8	llm	5	4	5	14
4ec3555870a8	manual	5	4	2	11

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
4ec3555870a8	llm	5	4	5	14
4ec3555870a8	manual	4	3	2	9
c95f5372fe02	llm	5	5	5	15
c95f5372fe02	manual	5	3	1	9
c95f5372fe02	llm	5	5	5	15
c95f5372fe02	manual	5	5	4	14
c95f5372fe02	llm	5	5	5	15
c95f5372fe02	manual	5	5	5	15
c95f5372fe02	llm	5	5	5	15
c95f5372fe02	manual	5	5	5	15
c95f5372fe02	llm	5	5	5	15
c95f5372fe02	manual	5	5	5	15
c9c51875a046	llm	5	5	5	15
c9c51875a046	manual	4	4	2	10
c9c51875a046	llm	5	5	5	15
c9c51875a046	manual	4	5	4	13
c9c51875a046	llm	5	4	5	14
c9c51875a046	manual	2	3	2	7
c9c51875a046	llm	5	4	4	13
c9c51875a046	manual	3	3	1	7
c9c51875a046	llm	5	5	5	15
c9c51875a046	manual	4	4	3	11
1d7933b2b5df	llm	5	5	5	15
1d7933b2b5df	manual	5	4	4	13
1d7933b2b5df	llm	5	5	5	15
1d7933b2b5df	manual	5	5	4	14
1d7933b2b5df	llm	2	3	5	10

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
1d7933b2b5df	manual	5	4	3	12
1d7933b2b5df	llm	5	4	5	14
1d7933b2b5df	manual	5	4	4	13
1d7933b2b5df	llm	5	5	5	15
1d7933b2b5df	manual	5	5	3	13
800db0baa67b	llm	5	5	5	15
800db0baa67b	manual	5	4	5	14
800db0baa67b	llm	5	5	5	15
800db0baa67b	manual	5	4	4	13
800db0baa67b	llm	5	5	5	15
800db0baa67b	manual	5	4	3	12
800db0baa67b	llm	3	5	5	13
800db0baa67b	manual	5	5	4	14
800db0baa67b	llm	2	4	5	11
800db0baa67b	manual	5	5	3	13
1f470084f7fe	llm	4	5	4	13
1f470084f7fe	manual	3	4	4	11
1f470084f7fe	llm	4	4	4	12
1f470084f7fe	manual	4	5	2	11
1f470084f7fe	llm	4	4	5	13
1f470084f7fe	manual	3	3	2	8
1f470084f7fe	llm	5	5	5	15
1f470084f7fe	manual	4	4	5	13
1f470084f7fe	llm	4	5	5	14
1f470084f7fe	manual	4	4	3	11
f4ffe349c97b	llm	4	4	3	11
f4ffe349c97b	manual	4	4	5	13

Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
f4ffe349c97b	llm	5	5	4	14
f4ffe349c97b	manual	4	4	5	13
f4ffe349c97b	llm	4	5	5	14
f4ffe349c97b	manual	4	4	4	12
f4ffe349c97b	llm	3	3	3	9
f4ffe349c97b	manual	3	3	3	9
f4ffe349c97b	llm	5	4	5	14
f4ffe349c97b	manual	4	4	4	12
64414c28876e	llm	5	5	5	15
64414c28876e	manual	5	5	4	14
64414c28876e	llm	3	5	5	13
64414c28876e	manual	5	5	4	14
64414c28876e	llm	5	5	5	15
64414c28876e	manual	5	5	3	13
64414c28876e	llm	5	5	5	15
64414c28876e	manual	5	5	4	14
64414c28876e	llm	5	5	5	15
64414c28876e	manual	5	5	5	15
4379a709aaf7	llm	5	5	5	15
4379a709aaf7	manual	5	5	3	13
4379a709aaf7	llm	5	5	5	15
4379a709aaf7	manual	5	4	4	13
4379a709aaf7	llm	2	5	5	12
4379a709aaf7	manual	5	4	5	14
4379a709aaf7	llm	5	5	5	15
4379a709aaf7	manual	5	5	4	14
4379a709aaf7	llm	2	4	5	11

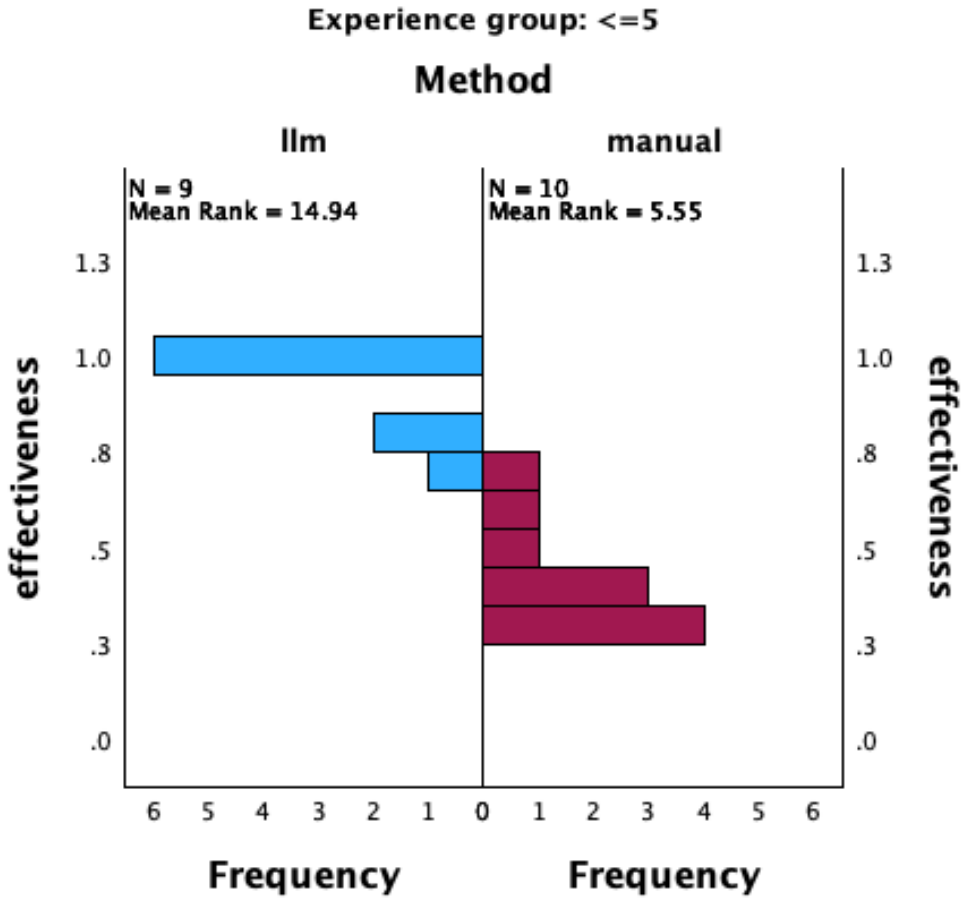
Continued on next page

Participant ID	Method[a]	Readability[b]	Comprehensibility[c]	Completeness[d]	Total quality score[e]
4379a709aaf7	manual	5	3	4	12

Note. This table presents the results of documentation quality evaluations performed by the developers.
[a] The method used for producing the documentation. The documentation is either manually produced or generated by an LLM.
[b] The readability score assigned by the developer.
[c] The comprehensibility score assigned by the developer.
[d] The completeness score assigned by the developer.
[e] The cumulative score of documentation quality (readability + comprehensibility + completeness).

Figure C.1.

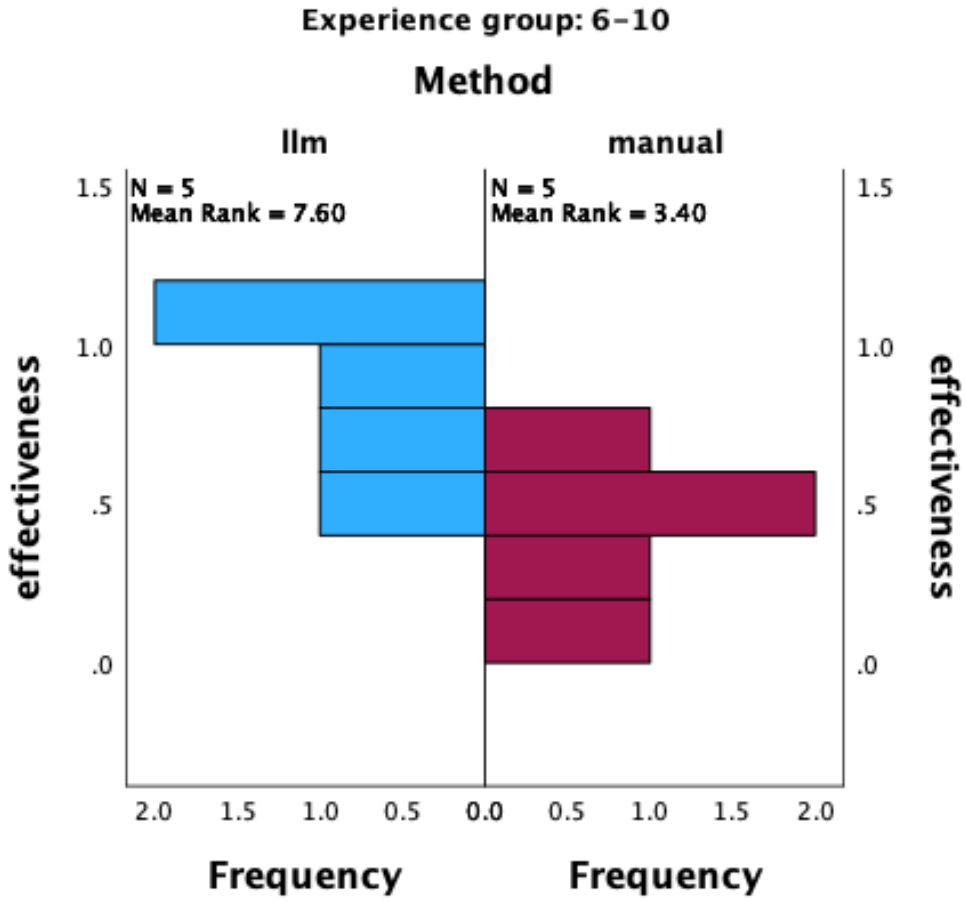
Mann-Whitney U test results for developers' effectiveness in documentation tasks for experience group ≤ 5



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers with less than 5 years of experience (group ≤ 5) in performing documentation tasks.

Figure C.2.

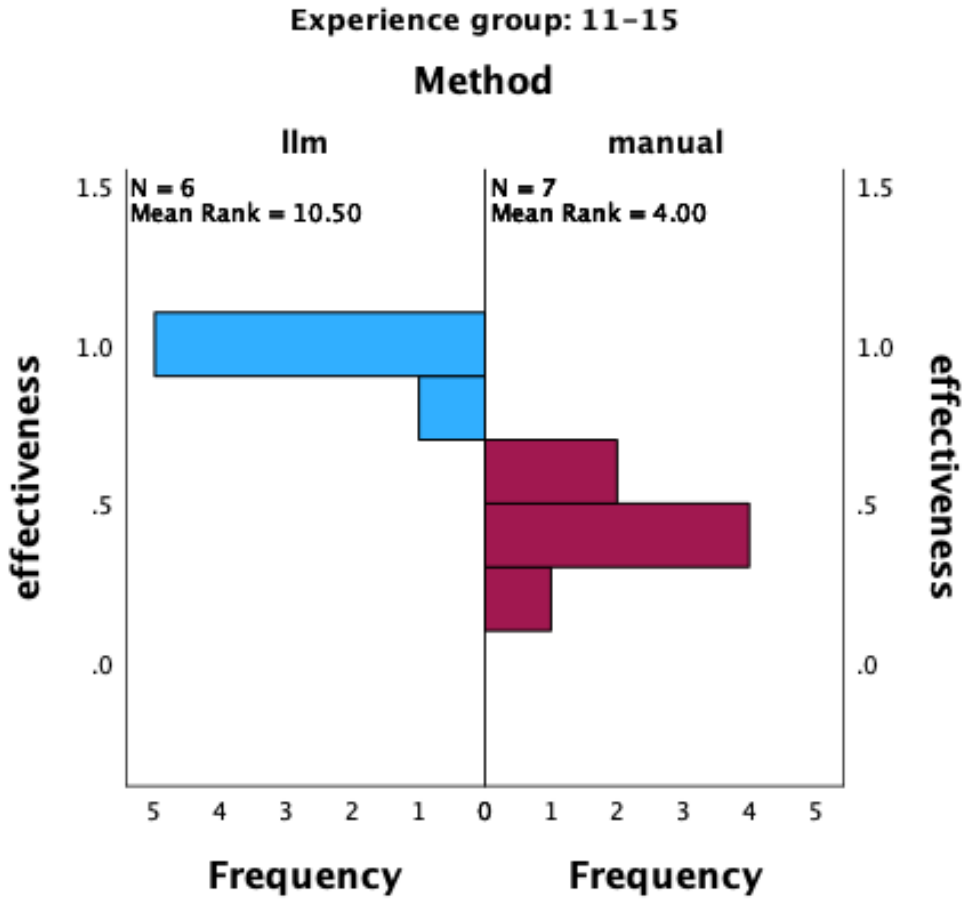
Mann-Whitney U test results for developers' effectiveness in documentation tasks for experience group $6 - 10$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $6 - 10$ in performing documentation tasks.

Figure C.3.

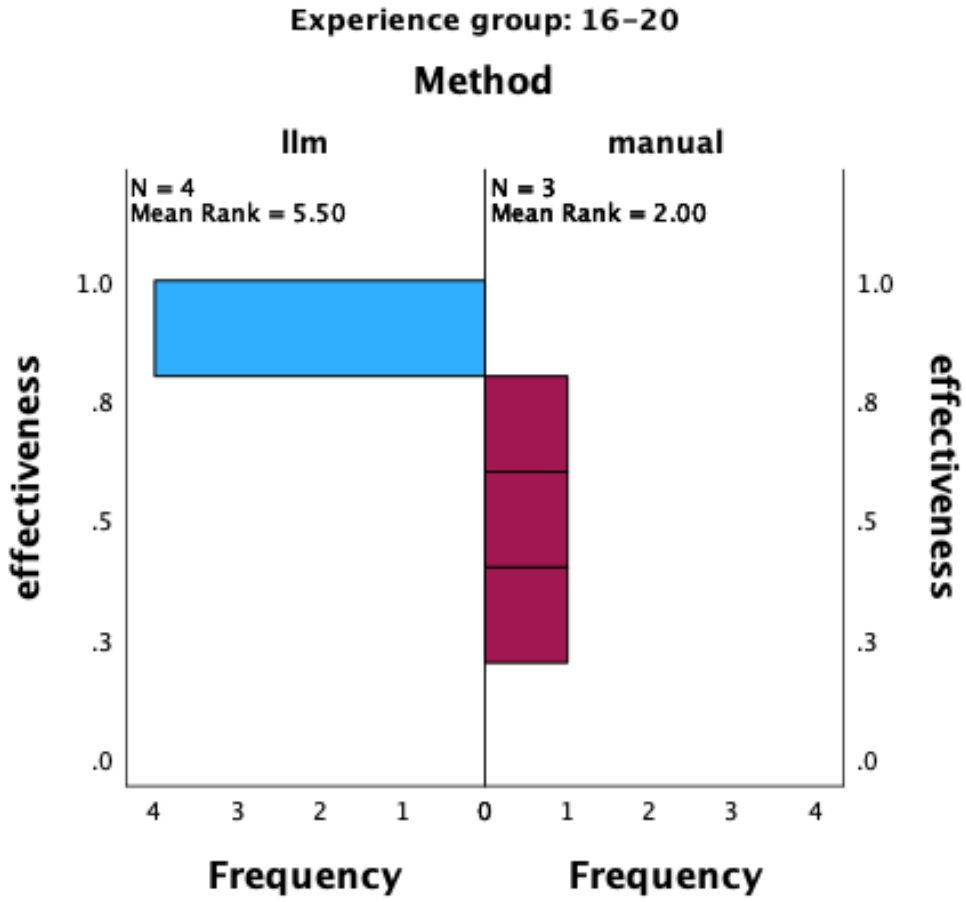
Mann-Whitney U test results for developers' effectiveness in documentation tasks for experience group $11 - 15$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $11 - 15$ in performing documentation tasks.

Figure C.4.

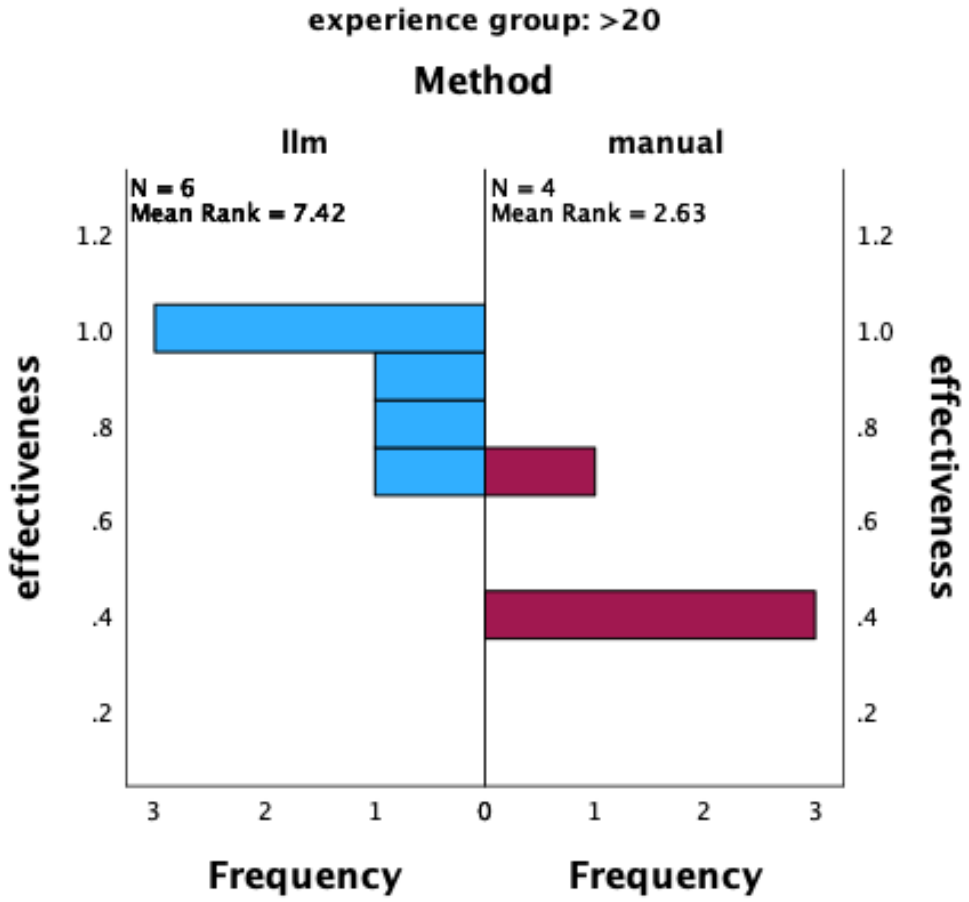
Mann-Whitney U test results for developers' effectiveness in documentation tasks for experience group $16 - 20$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $16 - 20$ in performing documentation tasks.

Figure C.5.

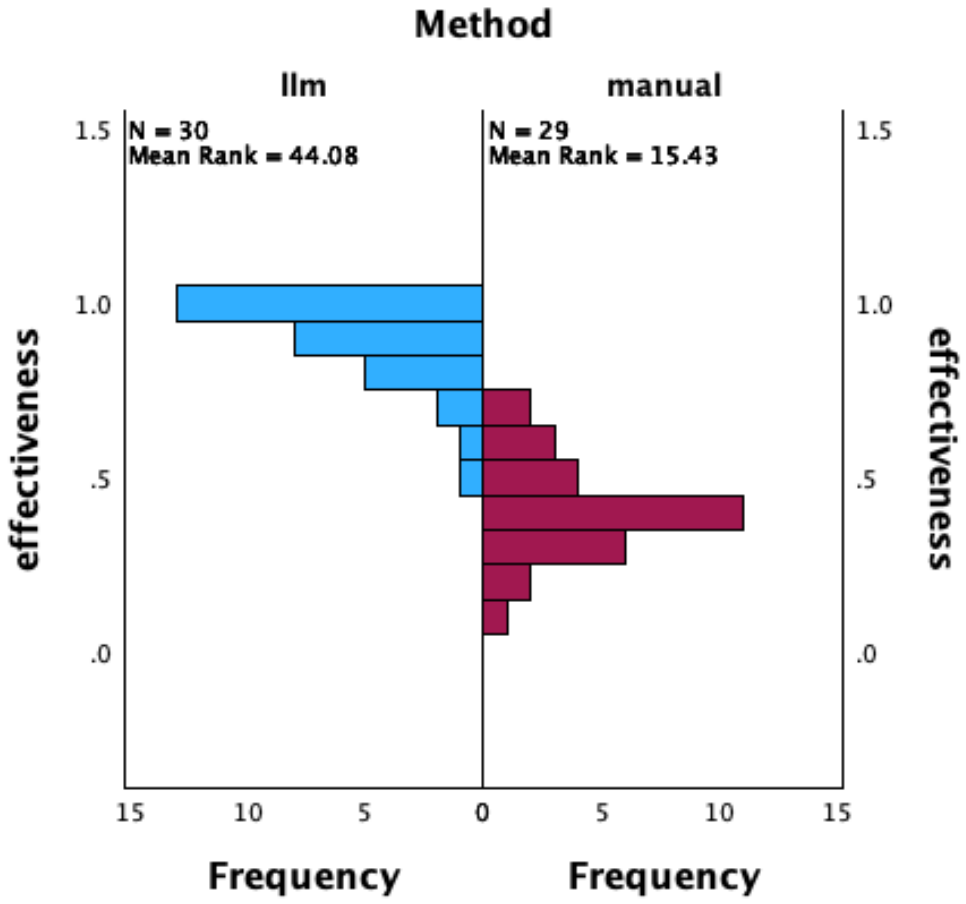
Mann-Whitney U test results for developers' effectiveness in documentation tasks for experience group > 20



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group > 20 in performing documentation tasks.

Figure C.6.

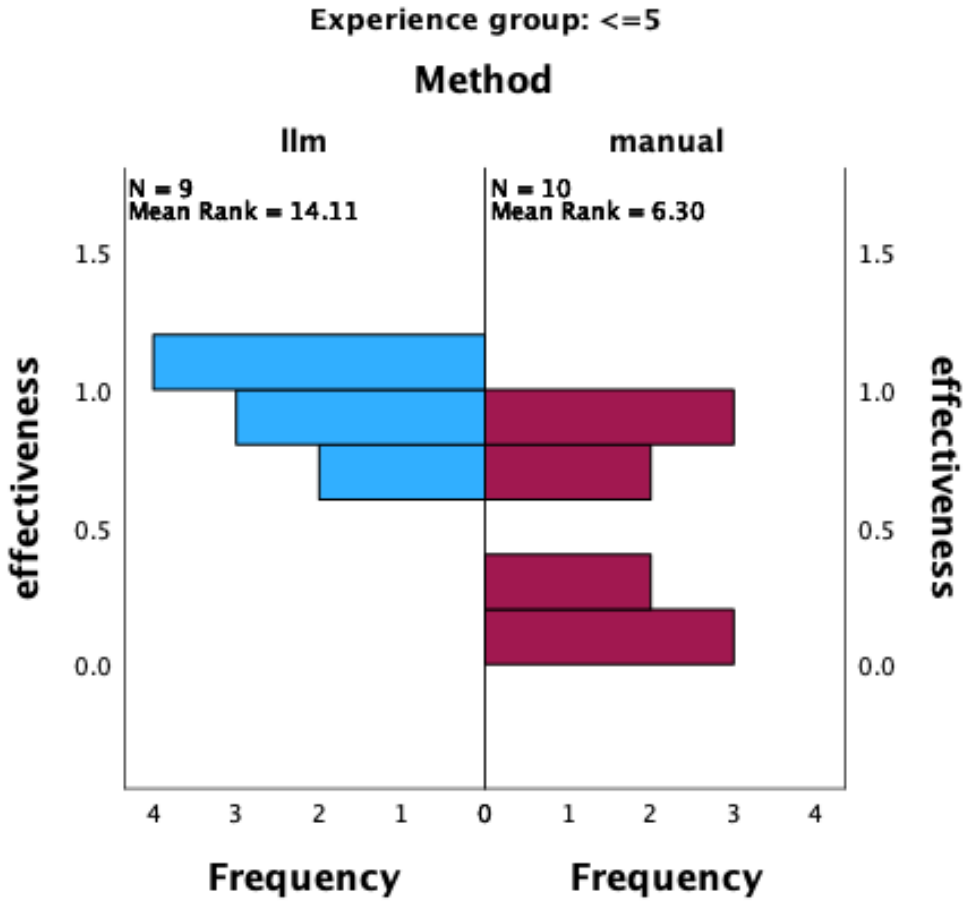
Mann-Whitney U test results for developers' effectiveness in documentation tasks for all experience groups collectively



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers in all experience groups collectively in performing documentation tasks.

Figure C.7.

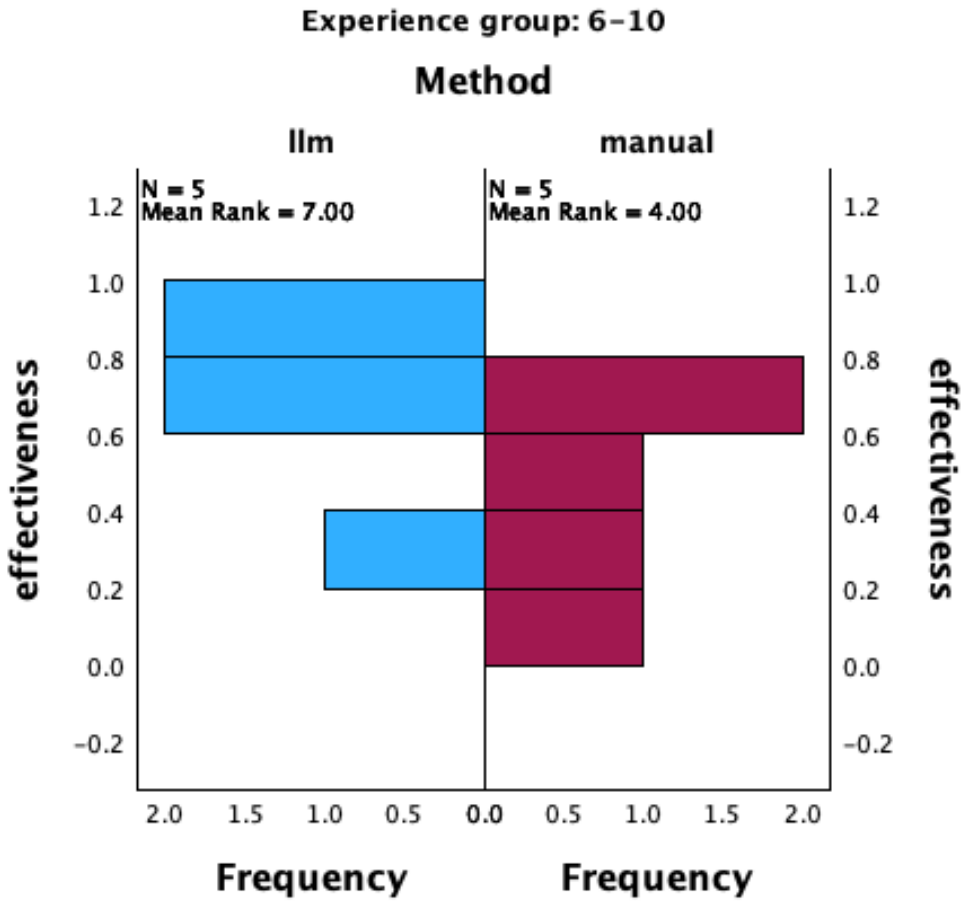
Mann-Whitney U test results for developers' effectiveness in programming tasks for experience group ≤ 5



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers with less than 5 years of experience (group ≤ 5) in performing programming tasks.

Figure C.8.

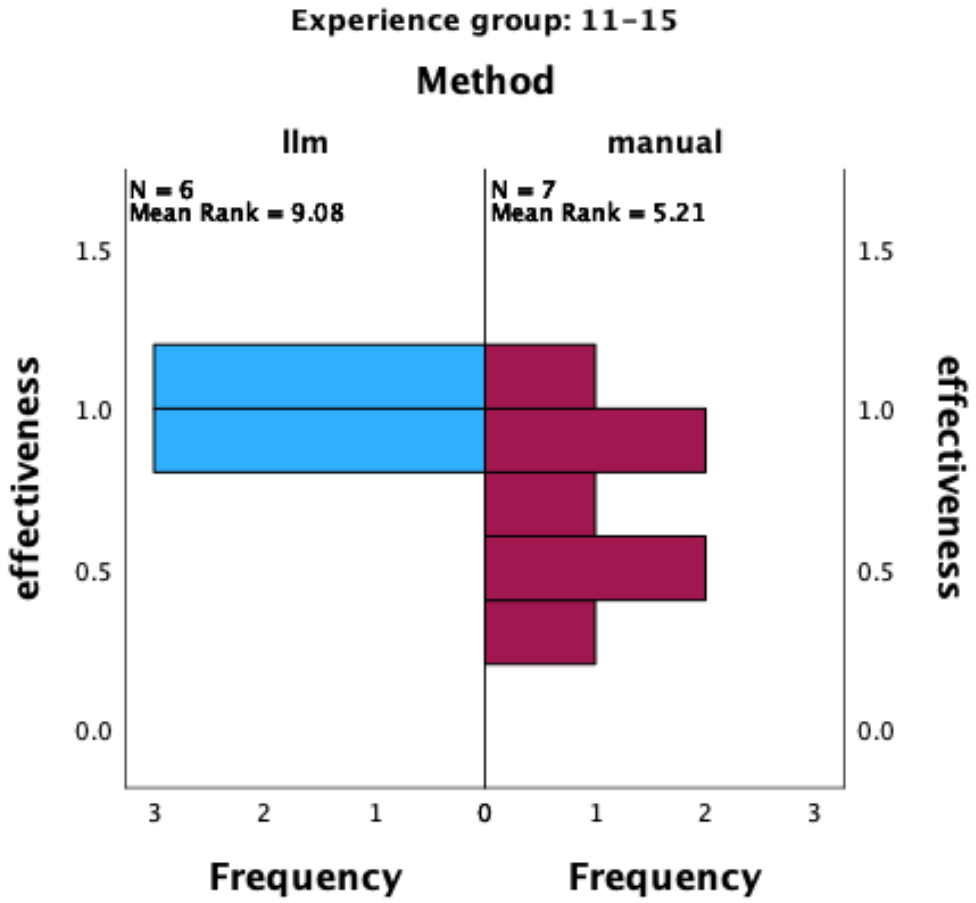
Mann-Whitney U test results for developers' effectiveness in programming tasks for experience group $6-10$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $6-10$ in performing programming tasks.

Figure C.9.

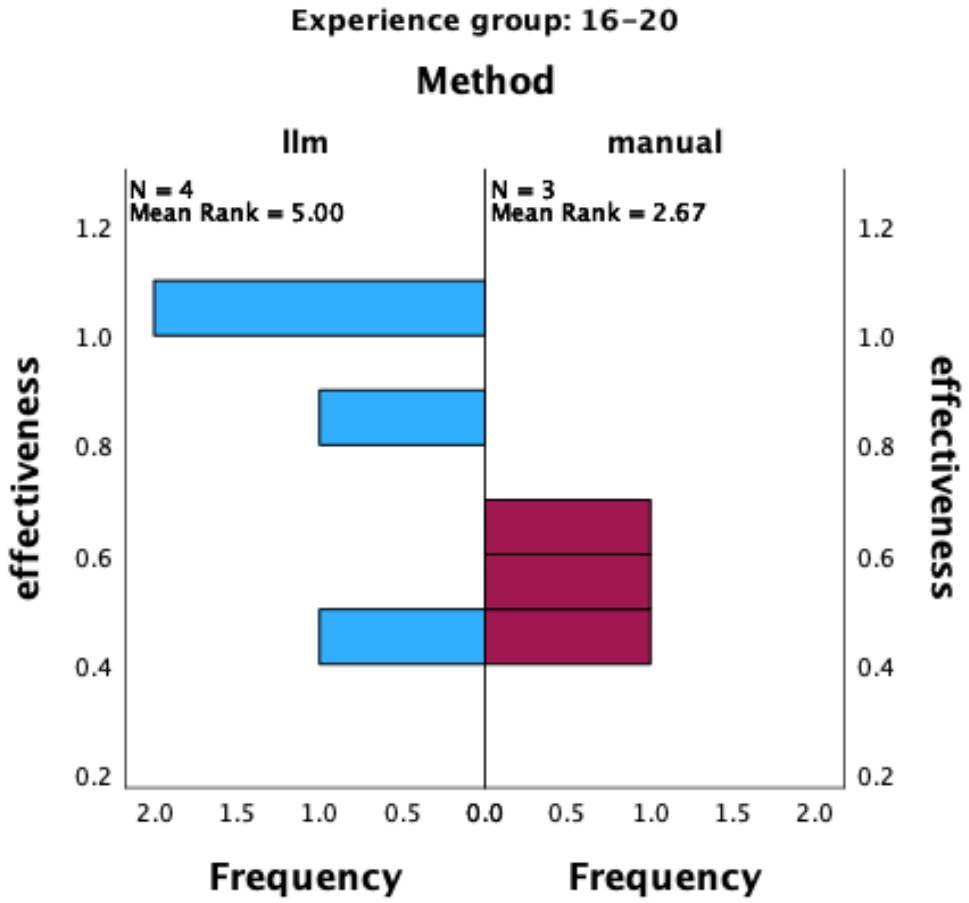
Mann-Whitney U test results for developers' effectiveness in programming tasks for experience group $11 - 15$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $11 - 15$ in performing programming tasks.

Figure C.10.

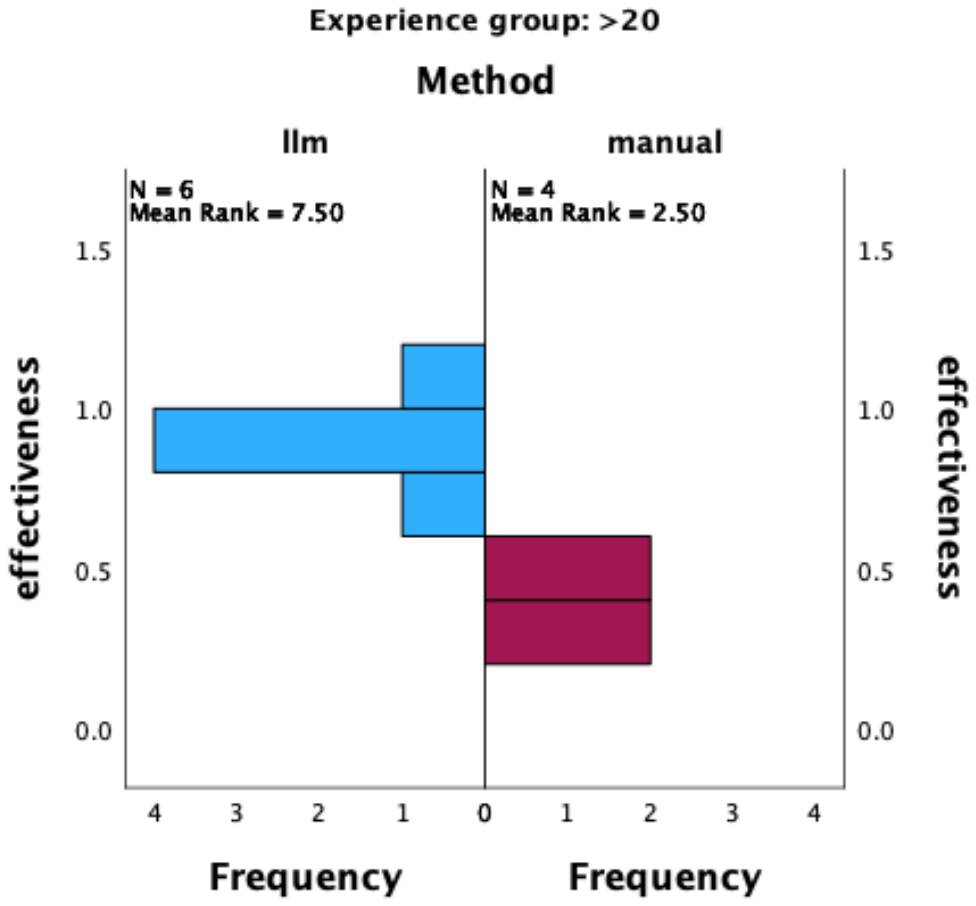
Mann-Whitney U test results for developers' effectiveness in programming tasks for experience group $16 - 20$



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group $16 - 20$ in performing programming tasks.

Figure C.11.

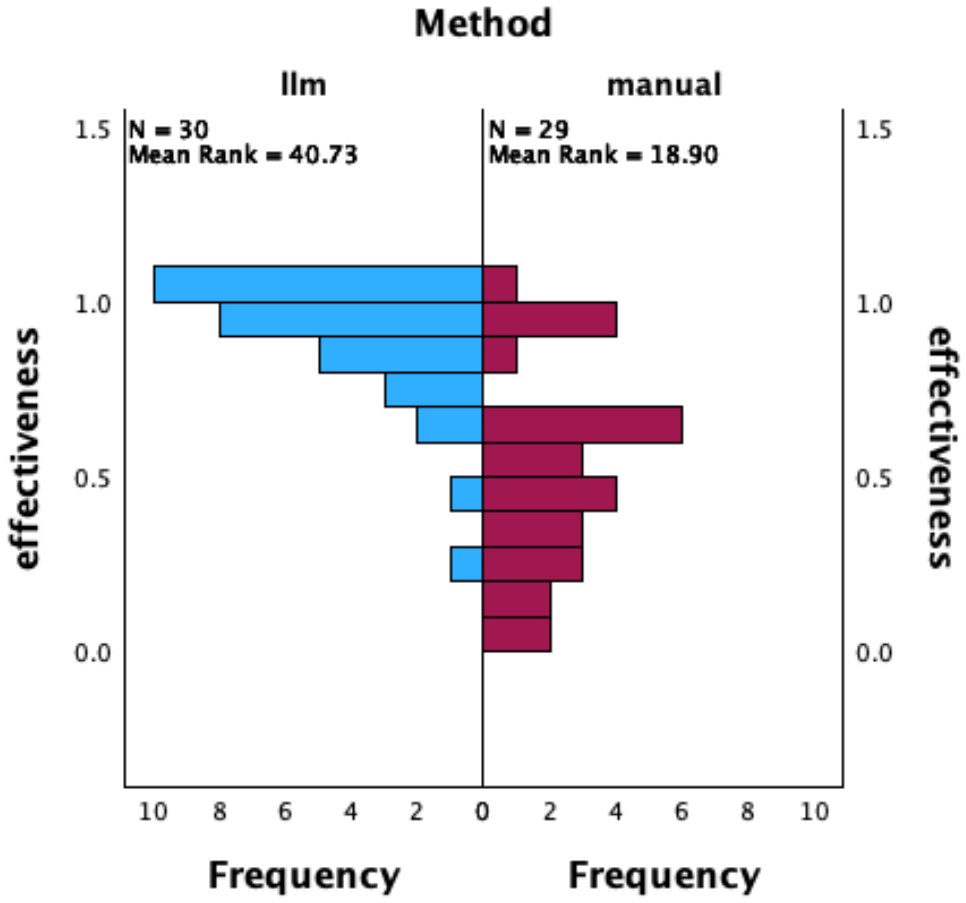
Mann-Whitney U test results for developers' effectiveness in programming tasks for experience group > 20



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers within the experience group > 20 in performing programming tasks.

Figure C.12.

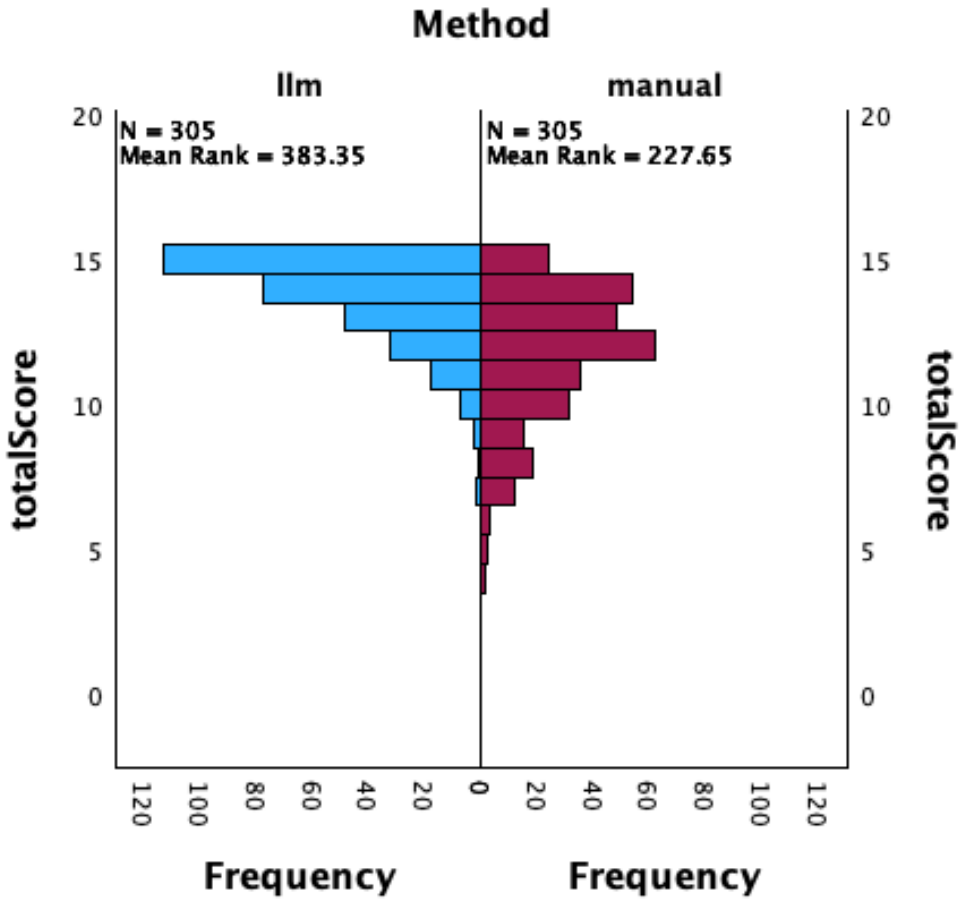
Mann-Whitney U test results for developers' effectiveness in programming tasks for all experience groups collectively



Note. This figure presents the results of the Mann-Whitney U Test, analyzing the effectiveness of developers in all experience groups collectively in performing programming tasks.

Figure C.13.

Mann-Whitney U Test results of the total quality scores



Note. This figure shows the results of the Mann-Whitney U Test, which evaluates the total quality scores given by developers to documentation created by other developers vs. documentation produced by the LLM-powered system. The scores are based on developers' assessments during the documentation quality survey.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden