The Influence of Using LLMs on the Activities of a Software Testing Team: An Industry Case

Flávia Oliveira*
Sidia Institute of Science and Technology
Manaus, Brazil
flavia.oliveira@sidia.com

Leonardo Tiago
Sidia Institute of Science and Technology
Manaus, Brazil
leonardo.albuquerque@sidia.com

Lennon Chaves*
Sidia Institute of Science and Technology
Manaus, Brazil
lennon.chaves@sidia.com

ABSTRACT

Context: In this study, we investigated the integration of Large Language Models (LLMs) in test teams, aiming to understand how these tools can improve the test process and improve productivity. The main goal was to determine if the implementation of LLMs can enhance the productivity of test professionals. **Methodology**: We performed an empirical study with 14 participants, and the results were analysed quantitatively and qualitatively. For the quantitative analysis, we analysed the frequency of activities, and for the qualitative analysis, we analysed the participants' perception of productivity through a scoring system, in which participants gave a score between 0 to 5 for their perception of learning from before and after the implementation of LLM in the team. **Results**: The results show a significant improvement in productivity (p-value = 0,0193) after the implementation of LLM. Furthermore, the qualitative analysis provided data on the impact of LLM in learning, use experience, trust perception and productivity. **Conclusion**: It was noted that the integration of LLMs in test teams can be a valuable tool to improve productivity and efficiency of test professionals.

KEYWORDS

Large Language Models, Software Industry, Software Testing

1 Introduction

Software tests aim to verify if the software behaves as expected, following requirements and specifications, and reaches the end-user free of defects [8]. In particular, the Software Test Life Cycle (STLC) occurs during the software development life cycle, and is usually divided in the following steps: requirement analysis, test planning, test development, test execution, test result report, retest of any issues found, regression test and conclusion of the test [3]. Requirement analysis consists in understanding the main requirements with which the test will be conducted. During test planning, the test plan is developed, so as to define the test's goals. During test development, test cases are created, establishing the conditions for the test. Then the test execution occurs, tests are performed and the results are compared with the expected results. After that the test results are reported, with "PASS" status if no issues are found, and "FAIL" status if any issues are detected during testing [3, 6]. However, each of these stages demands time, and the implementation of LLMs makes it possible to optimize the test process, improving the general quality of the software [2].

As affirmed by Wand et al. [9], through a revision of 102 works that used LLMs for software tests, LLMs have been applied to activities such as: generation of unit test cases, generation of test oracles, fixing programs, analysing and fixing issues. A similar work was developed by [5], in which they performed a systematic review of 395 papers regarding the use of LLM in Software Engineering, identifying which LLMs are being employed to solve issues in Software Engineering, how Software Engineering data is collected, pre-processed and used by LLMs, which techniques are used to optimize and evaluate LLMs for Software Engineering, and which Software Engineering activities were effectively approached with LLMs. Furthermore, LLMs have achieved satisfactory results for valid test case generation [7] and test code generation similarly to code generated by traditional tools [4].

Our work describes the implementation of Llama 3.1, which was introduced with the goal of analysing how members of the team use LLM for their daily activities since its implementation in July 2024. The test team in which this study was conducted acts within a software institute and is responsible for performing functional, sanity and exploratory tests. Besides test execution, team members participate of other activities such as writing test cases, automatically generating test codes, among others. Thus, this work presents the results of an empirical study of the use of LLMs in software test activities, in a real scenario from the industry.

2 Methodology

2.1 LLM Setup

To conduct this study, we used the 3.1 version of Llama, which was implemented in the test team during the month of July 2024, and since them the LLM has been available to be used by the test team. Llama was implemented using the Open Web UI platform, which serves as an interface between itself and the user. The test team received training on Prompt Engineering in the same month that Llama was implemented, and this training had 2 purposes: (1) encourage the test team to use the tool for their activities; (2) teach Prompt Engineering techniques to the team, so as to ease the team's interaction with the LLM.

2.2 Scope

We adopted the GQM (Goal-Question-Metric) framework [1] to define the main goal of the empirical study, as follows: Analyze **the usage of an LLM by practitioners** for the purpose of **evaluating** with respect to their **productivity perception** from the point of view of the **researchers** in the context of **software testing team which employed LLM to perform their daily activities**.

2.3 Planning

2.3.1 Context Selection. This study was conducted within the test team of a global company, whose members act in different activities,

*Both authors contributed equally in this research.

such as: requirement analysis, update or creation of test cases, issue analysis, test execution, test automation and development of test reports. The test team is located in a software institute that validates requirements in mobile devices.

2.3.2 Hypotheses. According to the experiment goals, we designed the null and alternative hypotheses, which were formulated based on productivity perception, comparing the results before and after using the LLM for software testing activities. The experiment was designed to test the following hypotheses:

- **Null Hypothesis (H_{01})**: There is no difference between **productivity** before and after using the LLM.
- **Alternative Hypothesis (H_{A1})**: There is a difference between **productivity perception** before and after using the LLM.

2.3.3 Variables. This study considered the following variables:

- **Independent variable**: The way to execute a software test task (with LLM or without LLM).
- **Dependent variables**: The dependent variable is called *Productivity Perception*, and is defined as a score, with a range of 0 to 5, in which each software tester performed an auto-evaluation in terms of productivity before and after the implementation of the LLM.

2.3.4 Participants. The participants of this study are professionals from the software industry that act as software testers in a software test team developing test cases, code for test automation, executing test cases, and analysing requirements and issues. This study was conducted with 14 participants selected based on a non-probabilistic sampling and by convenience [10].

2.3.5 Survey Design. We created a survey with 16 questions, 3 questions for characterization and 13 questions to comprehend the use of LLM by the participants.

2.3.6 Instruments. This work included the following instruments:

- **Informed Consent Form (ICF)**: The participants received a form to fill in, in which they agreed to participate in the study;
- **Opinion Survey Form**: Alongside the ICF, the participants answered a survey to help us understand their characterization and comprehension regarding the use of LLM.

2.4 Conducting the Study

To perform the study, we acted in two stages:

(1) **Pilot Study**: We conducted a pilot study, in which a participant who used LLM to perform their daily activities answered a form. After the pilot study, the form was adjusted to be later used during the study execution.
(2) **Study Execution**: The form was open to answers during 4 days, during the month of May. In all, 14 participants were able to contribute with their perceptions about the use of LLM.
(3) **Quantitative Analysis**: In this stage, we performed a quantitative analysis of the results by developing boxplots, Shapiro Wil normality tests and statistical tests with 0.05 significance level to validate the hypothesis.

Table 1: How LLM is used by the Software Testing Team?

How LLM is used by the software testing team?	Frequency
Use of LLM for studies: learning or clarifying new concepts	8
Use of LLM to clarify doubts about automation code	8
Use of LLM to translate texts	7
Use of LLM to improve technical reports	5
Use of LLM to refactor automation code	3
Use of LLM to support in management activities	3
Use of LLM to improve the description of a bug report	3
Use of LLM to conduct scientific research	2
Use of LLM to create or generate automation code	2
Use of LLM to document automation code	2
Use of LLM to refine a requirement	1
Use of LLM to generate a bug report	1
Use of LLM to write scientific articles	1

(4) **Qualitative Analysis**: We performed a qualitative analysis with feedback from the participants, so as to complement the quantitative analysis.

3 Findings

3.1 Participants Characterization

The study was conducted with 14 participants , all members of the test team, including 3 interns (22%), 6 junior testers (43%), 3 senior testers (21%) and 2 test leaders (14%).

Furthermore, we mapped the frequency of use of the LLM to perform test activities. It was observed that 5 participants use the LLM frequently (several times a week), representing 36% of participants. 4 participants use the LLM daily, representing 28% of participants. Lastly, 5 participants only use the LLM occasionally (a few times a month), representing another 36% of participants. We did not have any participants using the LLM rarely. Thus, we understand that our sampling is representative, given that participants do use the LLM to perform test activities.

3.2 Quantitative Analysis

One of the survey's questions regarded for which activities the members of the test team used the LLM, with predefined answers, but the participants could choose more than one answer and also propose new answers. The activities most frequently performed by the team are: (1) "Use of LLM for studies: learning or clarifying new concepts", (2) "Use of LLM to clarify doubts about automation code" e (3) "Use of LLM to translate texts", as illustrated in Table 1. Furthermore, the test team uses the LLM to improve technical reports, refactor automation codes, improve the description of bug reports, among others.

Another analysis we performed was in regards to the productivity after the implementation of the LLM in the test team. Participants had to give a score from 0 to 5 about their perception of productivity before and after the use of the LLM. Figure 1 illustrates the boxplot, and shows that the productivity was improved after the implementation of the LLM in the test team. To confirm, we performed a statistical test, in which firstly we applied the Shapiro Wilk test to measure the normality of the samples' distribution, and observed that the p-value was > 0.17, indicating that the samples follow a normal distribution. After that, we performed a parametric test that achieved p-value 0.0193 with 0.05 significance level, indicating that we can reject the null hypothesis H_{01}, that is, there is a

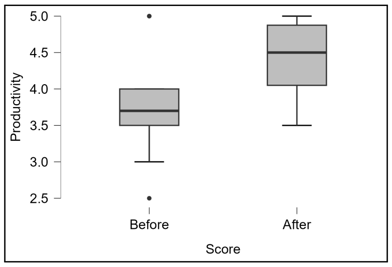

Figure 1: Boxplot of Scores from Productivy Perception

difference in the productivity of the test team after the implementation of the LLM, indicating that the productivity actually improved. The tests were performed using JASP Stat in version 0.19.1.

3.3 Qualitative Analysis

We performed the analysis of the feedback of the participants about LLM usage, and we obtained the following categories:

- **Use of LLM to Support Learning**: The implementation of LLM has been demonstrated as a tool to enhance learning through interactive engagement. Participants indicated that utilizing LLM facilitates content presentation and promotes learning for testing professionals. They also noted that LLM reduces cognitive load for task comprehension, establishing it as a supportive learning tool.
- **Experience of Use**: Participants found LLM use straightforward and intuitive. However, some indicated needing better understanding of LLM's capabilities and limitations to optimize its use, particularly in structuring prompts for more assertive outputs.
- **Perception of Confidence**: Participants indicated that LLM enhances test professionals' confidence in executing activities, due to its accurate and rapid responses. However, some participants express the need to validate responses before considering them definitive.
- **Practical Use of LLM**: The testing team employed LLM in a variety of practical applications. These include facilitating communication with non-native speakers through translation, and utilizing LLM as a tool for brainstorming and information retrieval.
- **LLM improves productivity**: Participants indicated that LLM utilization is correlated with enhanced productivity. This enhancement is attributed to the model's accuracy and rapid response capabilities, which afford participants the perception of increased efficiency in task execution. Furthermore, participants noted that employing LLM appeared to diminish the time required for task completion, thereby streamlining activities.
- **Limitations in using LLM**: Some participants reported challenges in formulating appropriate prompts for the model, indicating a need for training to enhance the effective use of LLMs. Additionally, they noted that the responses were not consistently precise, suggesting that software-testing professionals occasionally encounter hallucinations.

4 Conclusions

This study explores the integration of an LLM within a team to facilitate the execution of software testing activities over a period of 10 months from July 2024. To comprehend this phenomenon, we conducted a survey to gather participants' perceptions of the test professionals, thereby enabling quantitative and qualitative assessments. In the quantitative evaluation, we identified the activities most frequently supported by LLM, its use in aiding learning, clarifying doubts regarding test automation codes, and translating texts. Furthermore, we assessed the perceived productivity of the LLM using statistical tests, which yielded a p-value of 0.0193, indicating a significant change in productivity following the introduction of LLM in the testing team. Additionally, the qualitative evaluation revealed that LLM supports the learning of test professionals, is user-friendly and intuitive, enhances participants' confidence, and can be employed in a variety of software testing activities. The analysis also indicated that participants recognized the necessity of verifying answers and encountering challenges in designing prompts that generate more appropriate responses for their specific activities. Future research could explore how the use of prompts can enhance participants' activities and how the incorporation of LLM contributes to collaboration within the testing team.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020. The authors extend their gratitude to the Software Engineering Laboratory for its contribution to the IT infrastructure.

REFERENCES

[1] Victor R Basili. 1994. Goal, question, metric paradigm. *Encyclopedia of software engineering* 1 (1994), 528–532.

[2] Vahit Bayrı and Ece Demirel. 2023. Ai-powered software testing: The impact of large language models on testing methodologies. In *2023 4th International Informatics and Software Engineering Conference (IISEC)*. IEEE, 1–4.

[3] Gerald D Everett and Raymond McLeod Jr. 2007. *Software testing: testing across the entire software development life cycle.* John Wiley & Sons.

[4] Vitor Guilherme and Auri Vincenzi. 2023. An initial investigation of ChatGPT unit test generation capability. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*. 15–24.

[5] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.

[6] Muhammad Abid Jamil, Muhammad Arif, Normi Sham Awang Abubakar, and Akhlaq Ahmad. 2016. Software testing techniques: A literature review. In *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*. IEEE, 177–182.

[7] Laura Plein, Wendkûuni C. Ouédraogo, Jacques Klein, and Tegawendé F. Bissyandé. 2024. Automatic Generation of Test Cases Based on Bug Reports: a Feasibility Study with Large Language Models. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 360–361. doi:10.1145/3639478.3643119

[8] Sanjay Kumar Singh and Amarjeet Singh. 2012. *Software testing*. Vandana Publications.

[9] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering* (2024).

[10] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.