Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2024.056979

ARTICLE

Automation of Software Development Stages with the OpenAI API

Verónica C. Tapia[1,2,*] **and Carlos M. Gaona**[2]

[1]Sistemas de Información, Universidad Técnica de Cotopaxi, Latacunga, Cotopaxi, 050108, Ecuador

[2]Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Santiago de Cali, 760042, Colombia

*Corresponding Author: Verónica C. Tapia. Email: veronica.tapia@utc.edu.ec

Received: 04 August 2024 Accepted: 28 October 2024 Published: 03 January 2025

ABSTRACT

In recent years, automation has become a key focus in software development as organizations seek to improve efficiency and reduce time-to-market. The integration of artificial intelligence (AI) tools, particularly those using natural language processing (NLP) like ChatGPT, has opened new possibilities for automating various stages of the development lifecycle. The primary objective of this study is to evaluate the effectiveness of ChatGPT in automating various phases of software development. An artificial intelligence (AI) tool was developed using the OpenAI—Application Programming Interface (API), incorporating two key functionalities: 1) generating user stories based on case or process inputs, and 2) estimating the effort required to execute each user story. Additionally, ChatGPT was employed to generate application code. The AI tool was tested in three case studies, each explored under two different development strategies: a semi-automated process utilizing the AI tools and a traditional manual approach. The results demonstrated a significant reduction in total development time, ranging from 40% to 51%. However, it was observed that the generated content could be inaccurate and incomplete, necessitating review and debugging before being applied to projects. In conclusion, given the increasing shift towards automation in software engineering, further research is critical to enhance the efficiency and reliability of AI tools, particularly those that leverage natural language processing (NLP) technologies.

KEYWORDS

Artificial intelligence tools; user stories; ChatGPT; AI estimation; automation of software development processes

1 Introduction

While artificial intelligence (AI) is not a new concept, the exponential growth of data and the continuous increase in processing power are profoundly transforming business operations. Deep learning has spurred significant advancements in areas such as computer vision and natural language processing. The rapid adoption of generative artificial intelligence technologies (specifically, those that utilize algorithms to generate new data, like linguistic models such as ChatGPT) has greatly simplified the accessibility and usability of AI-based products and services.

The importance of generative AI models, such as ChatGPT and Bard, in agile software development is becoming increasingly evident. In agile methodologies, where collaboration and adaptability are key, these models serve as powerful tools to enhance and accelerate the software development

process across several critical stages. During the project planning and definition phases, these models can assist development teams by efficiently generating User Stories.

By accurately interpreting and translating customer requirements into clear and detailed descriptions, these models contribute to a better understanding of project objectives and facilitate communication between team members and stakeholders. In the design phase, generative models can be employed to create technical documentation and specifications, streamlining the prototyping process and informing key decisions regarding software architecture. In the development phase, these models can aid in generating source code, potentially expediting the implementation process significantly. Although human oversight and review are still necessary, this capability can boost productivity and reduce the risk of errors.

In this paper, we present an approach that leverages OpenAI's "gpt-3.5-turbo" model to address a pertinent challenge in agile software development: the automatic generation of User Stories (US). User Stories are essential elements in the planning and execution of software development projects, and our approach explores how AI can effectively contribute to this fundamental process. Through this study, we aim to open new perspectives on using advanced models to enhance efficiency and quality in agile software development.

To support this, we have designed a tool utilizing OpenAI's application programming interface to generate User Stories and estimate the effort required for various tasks. These capabilities were tested across three distinct practical scenarios: developing a medical appointment and medication reminder system, implementing a hotel room reservation system, and designing a vehicle reservation and rental system.

Additionally, ChatGPT was employed for the automatic generation of application code. In parallel, the same use cases were developed manually using traditional methods. A detailed comparison of the development time between these two approaches was then conducted, allowing us to evaluate the relative efficiency and effectiveness of automation *vs*. conventional practices in implementing the required solutions. This comparative analysis provided valuable insights into the strengths and limitations of automation in the software development process.

The structure of this paper is as follows: Section 1 outlines the primary objective, provides a brief overview of the process, and outlines the paper's structure. Section 2 presents a comprehensive overview of essential topics related to ChatGPT, including its core principles, implementation specifics, and additional contextual information in various subsections. Section 3 details the research methodology employed. Section 4 presents the results obtained, and Section 5 discusses the conclusions drawn from the study.

2 Review of the Literature

2.1 Artificial Intelligence

John McCarthy, credited with coining the term "artificial intelligence" (AI) in 1955, defined it as the capability of machines to utilize language, conceptualize, address challenges, and engage in self-improvement [1]. Since its inception, AI has grown into a distinct discipline that aims to replicate human thought and behavior. AI's scope has expanded, attracting researchers from diverse fields. This interdisciplinary collaboration has applied AI to various domains, including education [2]. AI systems mimic human reasoning by leveraging algorithms inspired by the human brain's neural networks. The increasing adoption of machine learning and data processing tools continues to advance AI's role in

industries such as business and government, where AI contributes to solving complex problems and promoting sustainability [3].

2.2 Natural Language Processing

Natural Language Processing (NLP) is a vital branch of AI that focuses on the interaction between computers and human languages. NLP enables machines to comprehend, interpret, and generate human-like text or speech. While early NLP models emerged in the 1950s, recent advances in deep learning have significantly enhanced their performance [4]. Modern NLP applications include chatbots and virtual assistants, which provide personalized user support across domains like language learning and research. For instance, ChatGPT, an advanced generative language model, has gained widespread use since its release in early 2023, assisting users in diverse contexts [5].

2.3 ChatGPT

Chatbots have evolved from simple scripts to advanced systems deployed on servers and cloud platforms. These bots manage software repositories, answer questions, and support collaboration, utilizing conversational and voice-activated interfaces [5]. The primary objective of these bots is automating repetitive tasks, though their success depends on effective design and transparent communication capabilities.

ChatGPT, developed by OpenAI, is a notable example of a sophisticated AI chatbot designed to engage in human-like conversations using machine-learning techniques. It was built upon the GPT (Generative Pre-training Transformer) architecture, first introduced in 2018. GPT's goal was to predict subsequent words in a sequence by training on a vast dataset of human-generated text, finding success in fields like machine translation and text generation [6].

Although ChatGPT is advanced, it lacks consciousness, relying solely on programmed algorithms and reinforcement learning to mimic specific typing styles and conversation patterns. The model improves over time by learning from user queries and refining its responses [7]. ChatGPT excels in generating human-like text and retrieving information, making it a versatile tool for numerous applications, though it must be noted that the model's outputs can reflect inherent biases from the training data [8].

The training of ChatGPT, leveraging substantial data and computational resources, has propelled AI content generation forward. Its architecture, based on a deep learning framework, combines multiple components like pre-trained models and algorithms, though ethical considerations regarding data handling and bias remain prominent challenges [9].

OpenAI offers an API that enables developers to integrate advanced natural language processing capabilities into their applications. The API allows for tasks such as text generation, language translation, and question-answering, enhancing efficiency and user interaction across industries [10]. Several alternatives to ChatGPT, such as Google's Bard and Microsoft's Bing Chat, also provide conversational AI capabilities. Moreover, open-source platforms like Rasa and Botpress offer developers further customization for chat experiences [11].

2.4 Software Development

Software development is a multifaceted process that encompasses research, planning, design, development, testing, configuration, and maintenance. Over the last 50 years, this process has evolved from the informal "code and fix" approach to the adoption of disciplined methodologies aimed at

enhancing predictability and efficiency, particularly as systems have grown in complexity [12]. In software engineering, the development process is not a rigid directive but a flexible framework that allows teams to choose the most appropriate actions to ensure timely delivery of high-quality software. This framework typically consists of five core activities: communication, planning, modeling, building, and deployment, which are applied iteratively in cycles, each yielding incremental improvements in functionality [13].

Agile methodologies, a prominent approach in modern software development, are designed to accommodate changes in both product requirements and team composition. Agile teams emphasize flexibility, collaboration, and continuous customer involvement. They adapt quickly to new challenges by maintaining open communication and involving customers directly in the process, fostering a shared understanding of evolving requirements [13]. Flexibility is key in agile planning, as it allows teams to remain responsive to changes, even in uncertain or unpredictable environments.

Agile methodologies rest on three core principles: the difficulty of predicting future requirements, the tight interconnection between design and implementation, and the inherent unpredictability across the analysis, design, and testing stages. Agile processes address this unpredictability by relying on iterative development cycles that continuously adapt to project and technical conditions. Customer feedback is crucial in this process, as it helps guide adjustments, ensuring the delivered software aligns with user expectations [13].

By breaking complex projects into smaller, manageable increments, agile methodologies allow for faster delivery, flexibility in scope, and continuous improvement based on user feedback. This approach emphasizes collaboration, adaptability, and responsiveness, both within the development team and between the team and the end users. The result is a more efficient, adaptive process that meets evolving project demands while ensuring software quality [14].

2.5 User Stories

The concept of "User Story," introduced by Kent Beck in 1997, and formalized by Rachel Davies in 2002, captures customer problems from their perspective using the format: "As a {role}, I want {action} so that {justification/value}" [15]. User stories are concise narratives describing the interaction between a user and the system, emphasizing the value the user derives rather than focusing on technical details [16,17].

User stories break down customer goals into small, manageable tasks that can be rapidly implemented and reviewed for Quality Assurance (QA). When a story is complex or time-consuming, it is divided into smaller tasks to ensure early testing and feedback [18]. The time required to complete these tasks depends on factors such as the effort needed to optimize user experience, necessary preliminary research, and any external feedback delays that might impede progress.

2.6 Related Work

Experiments assessing ChatGPT's effectiveness across various domains are advancing, showcasing its potential in a range of fields.

Greengard [19] conducted a study evaluating ChatGPT's ability to assist clinical decision-making in salivary gland treatments. While the study demonstrates ChatGPT's promise in improving clinical processes, further development is necessary to ensure its reliability and safety in medical settings.

Altamimi [20] explored ChatGPT's use in automated essay grading, focusing on its accuracy and reliability. The study found that ChatGPT significantly improves the efficiency of grading, though it must be compared carefully against traditional grading methods.

In the context of software engineering education, a study [21] examined ChatGPT's performance in coding tasks, error correction, and mathematical logic problems. ChatGPT provided valuable insights, generating ideas and debugging code, but the authors emphasized the need for cross-verification with human expertise to ensure accuracy.

Another study [22] assessed the impact of ChatGPT on software engineering education, highlighting its potential to deliver personalized feedback to students. The study stressed the importance of adapting curricula to integrate generative AI tools while addressing concerns about their effect on the assessment process.

Petrović [23] demonstrated the application of ChatGPT in DevSecOps for runtime log analysis, finding it comparable to traditional classification methods in terms of precision. The study pointed to future research on integrating code generation for system recovery and attack mitigation.

Said et al. [24] introduced "Adam," an animatronic robotic head using ChatGPT, showcasing its effectiveness in customer service with an impressive 96% recognition rate in question-response tasks. This innovation underscores the practical applicability of ChatGPT in human-machine interaction.

In the realm of Infrastructure as Code (IaC), Petrović [25] applied ChatGPT for static analysis, identifying security and syntax issues in Terraform and Ansible scripts. Although promising, the study noted limitations such as processing time and associated costs, positioning ChatGPT as a supplemental tool in DevOps.

Uzair [26] proposed a six-tier AI architecture for automating software development, utilizing large language models like GPT to handle tasks across various abstraction levels. This approach offers adaptability and responsiveness, demonstrating resilience to changes in software development needs.

These recent researches have shown that AI tools, particularly large language models like ChatGPT, have the potential to accelerate various aspects of software development, including user story generation, automated coding, and debugging. However, despite these advancements, studies consistently emphasize that significant challenges remain. For instance, while ChatGPT has proven useful in clinical decision-making and medical data analysis, concerns regarding accuracy and safety limit its application in critical environments. Similarly, its performance in automated essay grading highlights efficiency, yet the risk of bias necessitates ongoing human oversight to ensure fairness and quality. In the realm of software engineering, ChatGPT's role in idea generation and debugging is promising, but developers are advised to use it as a supplementary tool due to its incomplete understanding of complex algorithms.

Overall, while AI technologies such as ChatGPT are being increasingly integrated into various fields, including software engineering, the need for human intervention remains essential. Current models exhibit limitations related to accuracy, bias, and processing speed, particularly in tasks like security automation and DevOps analysis. These issues underscore that AI tools are not yet fully autonomous nor suitable for replacing human developers entirely. Consequently, these models should be regarded as valuable complementary assets that augment human expertise rather than serve as standalone solutions.

3 Methodology

This section explains the steps of the procedure used for the study. There are eight key stages: identification of the problem, construction of the tool, testing, correction and debugging, proposal of the tool, application of the case studies with the tool, manual application of the case studies and, finally, comparison and documentation of the results (Fig. 1).

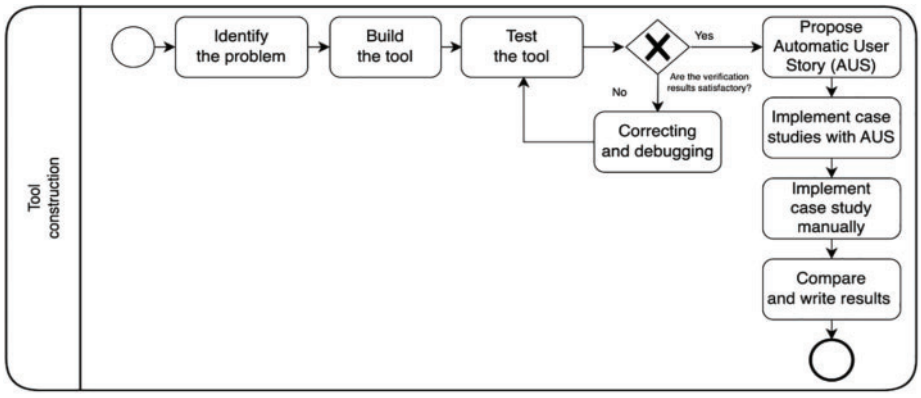

Figure 1: General stages of the investigation

3.1 Identify the Problem

This research aims to assess the effectiveness of ChatGPT in automating various phases of software development, with a specific focus on the generation of user stories and the estimation of effort required for each story. Additionally, we aim to quantify the time savings achieved through the use of the AI tool in comparison to traditional processes.

3.2 Build the Artificial Intelligence Tool

The tool, created utilizing the OpenAI API, boasts two primary functionalities: 1) Generating user stories for a given process or scenario, and 2) Estimating the effort required for the execution of these user stories. The development process for this tool is outlined as follows:

1. Requirements Gathering and Initial Planning:
   - Identification of the project objectives and needs, as well as the specific requirements of the AI tool.
   - Establishing Team Roles.
2. Defining User Stories:
   - Breaking down requirements into user stories.
3. Backlog Prioritization:
   - The product owner prioritizes user stories based on their value and complexity.
4. Development Iteration:
   - Executing work in brief development cycles, referred to as Sprints.
   - Within each sprint, the team selects a set of high-priority user stories for implementation.
5. Development:
   - Developers engage in the implementation of functionality, encompassing artificial intelligence aspects.

3.3 Test the Tool

The developed tool was subjected to unit and integration tests to ensure its correct operation. During each development cycle, corrections and adjustments were made based on the test results. Subsequently, the tool was applied in the selected case studies, where the execution time was measured and compared with manual methods, highlighting the improvements in efficiency provided by the use of AI.

3.4 Propose Automatic User Stories (AUS)

Following the successful completion of all tests and securing stakeholder approval, the tool is deployed into a production environment. Continuous monitoring of its performance takes place, and upon project completion, a retrospective review is conducted to assess the process and gather valuable lessons learned.

1. Internal modeling in the API

The internal process through the API may involve multiple stages of analysis and text generation. The model employs natural language processing techniques, such as the use of neural networks and attention mechanisms, to understand context, generate coherent text, and provide user stories. It decomposes the input into numerical representations, uses multiple layers of attention and processing to understand the context, and generates coherent text based on that context. In addition, the model is pre-trained with a large amount of textual data in order to generate contextually relevant and useful responses. Fig. 2 provides an overview of the process [10].

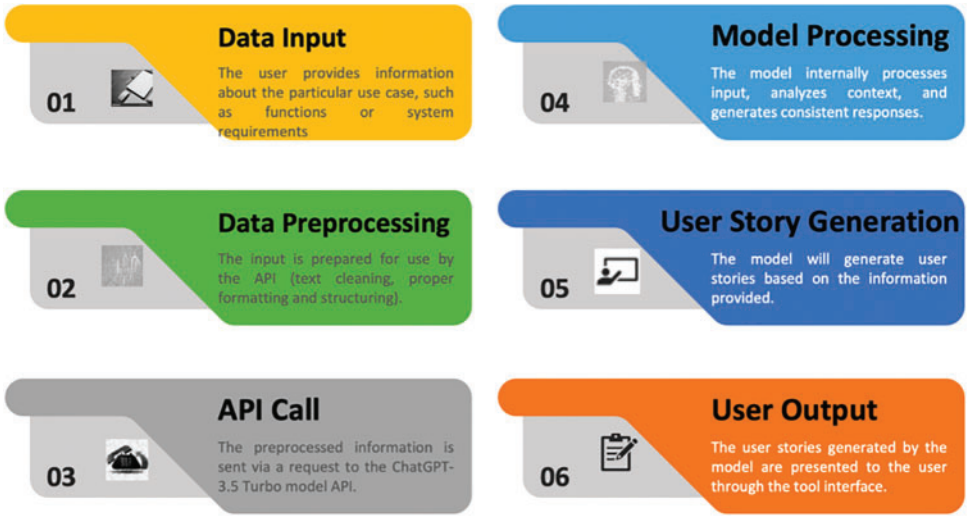

Figure 2: Internal process in the OpenAI API

3.5 Implement the Case Studies

This stage comprises two distinct processes:

- Executing the case studies with AUS
- Manually implementing the case studies

In each process, the execution times of various tasks are measured. A comparative analysis of the results enables the determination of which process is the most agile for software development and quantifies the reduction in development time. Subsequently, the findings are documented based on the acquired information.

Below are the specifics regarding the implementation of the case studies.

The experiment was conducted from April to August 2023 during the academic period within the subject "Fundamentals and Software Engineering (FIS)," part of the sixth cycle of the Information Systems program at the Technical University of Cotopaxi in Latacunga, Ecuador. The executed activities include the following:

1. Case study selection:

Three case studies (Table 1) were chosen from a pool of cases suggested by the students. The selection process considered general requirements, ensuring that the chosen problems were not overly complex and could facilitate the comprehensive execution of the project.

Table 1: Case studies

No.	Process
Case 1 (P1)	Reminders for medical appointments and medication.
Case 2 (P2)	System for managing inventory
Case 3 (P3)	Court reservation and rental

2. Formation of work teams:

During this academic period, 35 students were enrolled in the FIS course. All enrolled students have successfully completed the prerequisites, which include other programming subjects, and have actively participated in a minimum of two development projects. Teams were assembled based on the outcomes of previous projects, identifying the most accomplished students in programming.

Six teams, each consisting of four members, were established, with efforts made to ensure homogeneity within each team. Individuals who were not part of the development teams were involved in the process of measuring various aspects.

Each case is worked on by two teams:

- The initial team executes the process with assistance from the AI tool, which automatically generates two stages of the process. Additionally, ChatGPT is utilized to generate the application code, resulting in three automated stages (refer to Fig. 3).
- The second team manually executes the activities, adhering to agile practices in the conventional manner.

3. Implementation of the development stages:

In this phase, each team carries out various stages of software development, employing an agile methodological approach primarily guided by Scrum practices (Table 2).

The sole distinction is that three of them employ AUS and ChatGPT to automate particular processes (Fig. 3).

Figure 3: Stages automated with API OpenAI

Table 2: Stages of software development

Initial Planning Stage:
User Story Definition
Prioritizing the Product Backlog and determining the sprint duration.
Sprint Planning:
Selecting items from the Product Backlog for the upcoming Sprint and breaking down tasks.
Estimating the effort for each task.
Development:
Coding
Refining AI Code
Daily meetings to synchronize the team and discuss progress and obstacles.
Sprint Review:
Demonstration of completed functionalities
Evaluation, feedback and adjustment Product Backlog
Sprint Retrospective:
Adjusting and implementing improvement plans for the upcoming Sprint.
Preparation for the next Sprint:
The team updates the Product Backlog with feedback and new requirements.
Choosing tasks for the next Sprint based on prioritization and received feedback.
The iteration of Sprints involves repeating the steps according to the Sprint cycles, continuously enhancing the product and adjusting the process as necessary.
Product Delivery: Upon completion of the Sprints and ensuring the product is functional, a final review is conducted, and preparations for the launch are made.

4. Measurement of execution times:

Throughout the execution of each stage in the software development life cycle, the measurement team records the duration of each stage. The recording is documented in spreadsheets, and the time

measurement is in seconds. This choice is based on previous tests with ChatGPT, where the generation of such content rarely exceeds one minute. While executing the automated stages, measurements are repeated an average of three times to ensure data reliability. Therefore, records are documented with the averaged measurements from all repetitions.

5. Compare the results obtained from the implementation:

To compare the results obtained, execution time measurements were used for each phase of development, both in automated and manual processes. These measurements were recorded in seconds and multiple tests were performed to ensure data consistency. In addition, the correction and debugging time required to adapt the automatically generated code to the specific requirements of each case study was considered. These results are presented in tables and graphs to facilitate the understanding of the research.

4 Results and Discussion

The results indicate a significant time difference between executing tasks automatically (Fig. 3) and performing the same tasks manually.

The results are presented in seconds, reflecting the minimal time required to execute tasks with the AI tools (AUS and ChatGPT). Consequently, the measurements for manual tasks are also converted to seconds for consistency.

In Case 1 (Fig. 4) involving medical appointment reminders and the medication intake process, a system was devised to assist patients in efficiently monitoring their medical appointments and medication adherence. The outcomes reveal that the automated tasks require between 15.1 and 18015,1 s for completion, whereas the same tasks performed manually take between 14400 and 432000 s (Table 3).

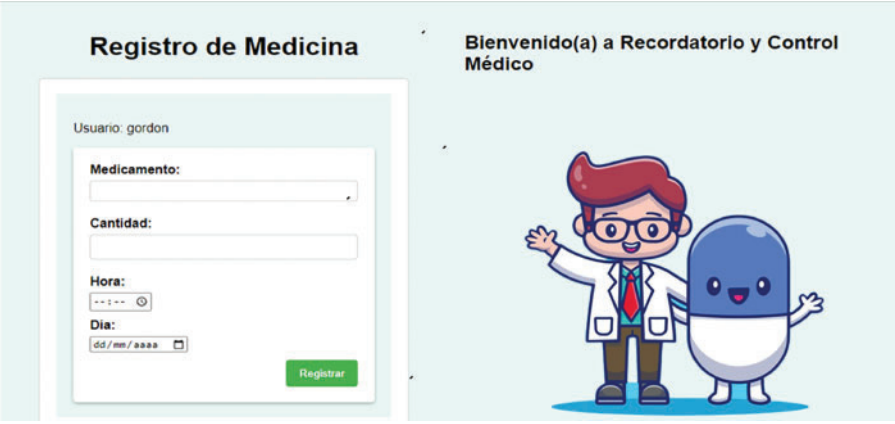

Figure 4: P1. Reminders for medical appointments and medication intake

Table 3: P1. Reminders for medical appointments and medication intake

Automatic tasks	TA	TM
Define user stories	45,18 s	14400 s
Estimate the effort required for each task	15,1 s	68400 s
Coding	18015,1 s	432000 s

Note: *TA = Automatic tasks; *TM = Manual tasks; *s = seconds.

In the context of Case 2 (Fig. 5), which involves the inventory process, a system designed to maintain a detailed and up-to-date record of an organization's products, materials, or assets in inventory, the results highlight a significant time difference between automatic and manual task durations. Automatic tasks range from 3.84 to 14403.84 s, whereas manual tasks take between 10800 and 288000 s (Table 4).

Figure 5: P2. Investment system

Table 4: P2. Investment system

Automatic tasks	TA	TM
Define user stories	29,39 s	10800 s
Estimate the effort required for each task	3,84 s	25200 s
Coding	14403,84 s	288000 s

Note: *TA = Automatic tasks; *TM = Manual tasks; *s = seconds.

Ultimately, in the context of Case 3 (Fig. 6), involving the reservation process and court rental, an application designed to oversee and streamline the reservation and utilization of sports facilities, the automatic tasks exhibit durations ranging from 8.23 to 14408.23 s. Conversely, manual tasks have durations falling within the range of 10800 to 288000 s (Table 5).

Figure 6: P3. Court reservation and rental

Table 5: P3. Court reservation and rental

Automatic tasks	TA	TM
Define user stories	27,1 s	10800 s
Estimate the effort required for each task	8,23 s	32400 s
Coding	14408,23 s	288000 s

Note: *TA = Automatic tasks; *TM = Manual tasks; *s = seconds.

The analysis of measurements enables us to infer that the task with the briefest execution time through the use of AI tools is the effort estimation task. Undoubtedly, this task holds significance as it aids in determining the time required for the implementation of each user story.

Conversely, the most time-intensive task is coding. This involves the generation of code for applications or processes. It's worth noting that ChatGPT can generate code for various programming languages; however, additional time is necessary for developers to analyze, debug, and adapt this code to ensure its functionality. Therefore, between 4 and 5 h were added, as calculated by each team.

Table 6 provides a comprehensive summary of the results, detailing the total time expended for each case based on the process type (semi-automatic or manual). It also highlights the time difference and the percentage of reduction achieved.

Table 6: General summary of results

Problem	Time TA	Time TM	Difference	Reduction
P1	446475,38 s	874800 s	428324,62 s	51%
P2	205237,07 s	507600 s	302362,93 s	40%
P3	244843,56 s	518400 s	273556,44 s	47%

Note: *TA = Automatic tasks; *TM = Manual tasks; *s = seconds.

The reduction is substantial, with the minimum reduction value being 40% of the time, reaching up to 51%. This implies that for every 10 h of manual work, the execution of a semi-automatic process with the assistance of AI tools results in a time reduction of at least 4 h (Figs. 7 and 8).

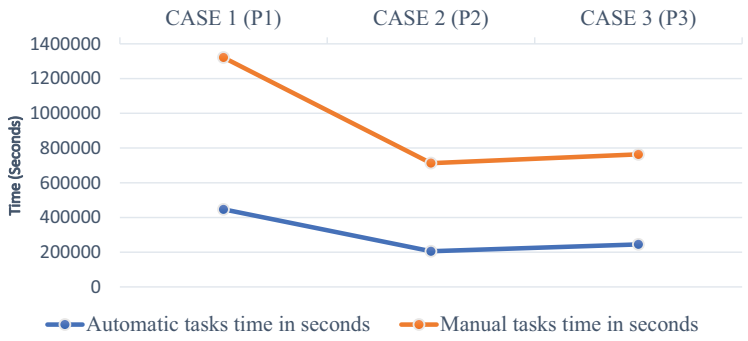

Figure 7: Time comparison: automatic tasks and manual tasks

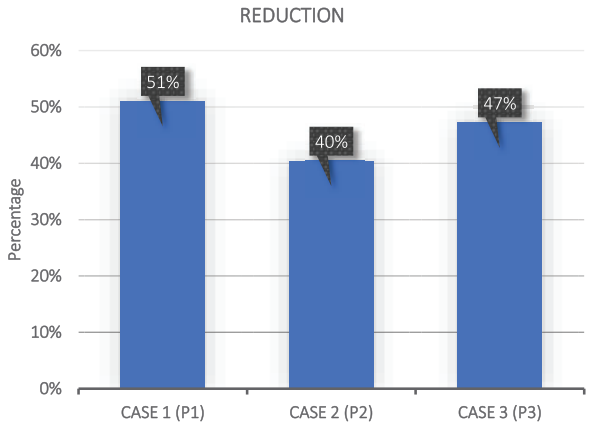

Figure 8: Time reduction with automatic tasks

As evident, the reduction in development time through the use of AI tools is highly significant. In essence, the support provided to development teams proves to be crucial, particularly in the three stages that were automated. However, it is crucial to highlight that the contents generated by AI tools should undergo analysis by developers to ensure their validity and applicability in the projects.

The study presents a detailed evaluation of the performance of automated tools, such as ChatGPT and AUS (Automatic User Stories), in comparison to manual development methods. Several critical dimensions can be analyzed from this comparison: time efficiency, code readability, testability and maintainability. This analysis can provide a comprehensive understanding of the strengths and limitations of both approaches:

– Time Efficiency. The results from the case studies demonstrate a significant reduction in development time when using automated tools. In all three cases, automation reduced the time by between 40% and 51% compared to manual development. Tasks such as user story generation, effort estimation, and code writing were completed much faster with AI-based automation.

For example, in Case 1 (medical reminder system), automated processes completed tasks in 15.1 to 18,015.1 s, while manual development took between 14,400 and 432,000 s. Similarly, in Case 2 (inventory management system), automated tasks were completed in 3.84 to 14,403.84 s, while manual efforts ranged from 10,800 to 288,000 s.

This dramatic improvement in time efficiency is primarily due to the speed at which AI tools like ChatGPT can generate functional code and content. However, the study also notes that, although the initial results are generated quickly, they often require post-processing in the form of debugging and refinement, which may slightly reduce the overall time savings.

– Code Readability. One of the main drawbacks of automation highlighted in the study is the lower readability of code generated by AI tools. Although automated systems can rapidly produce functional code, this code often lacks the structure and clarity that is typically achieved with manual coding practices.

  Readability is essential for long-term software maintenance, as future developers must be able to understand and modify the code with ease. The study indicates that AI-generated code tends to include unclear structures or convoluted logic, requiring further intervention by developers to improve its clarity and align it with best practices in software engineering.
– Testability. Another key dimension explored is the testability of the code. Automated tools like ChatGPT can quickly generate code, but this code often requires additional testing to ensure that it meets the system's functional requirements. Due to the inherent limitations of AI tools, which may not fully comprehend complex requirements, the generated code frequently contains errors or omissions that must be identified and rectified during testing.

  Developers often had to spend additional time conducting unit and integration testing on AI-generated code to identify and resolve these issues. This highlights the importance of human oversight during the testing phase, as automation alone cannot yet guarantee fully reliable code results.
– Maintainability. Maintainability refers to the ease with which software can be updated or modified throughout its lifecycle. The study emphasizes that while automated tools excel at quickly producing code, this code is not always optimized for long-term maintainability. Human developers typically adhere to modular and well-documented coding practices, which facilitate future modifications.

  In contrast, AI-generated code may be inconsistent and lack modularity, making it more difficult to adapt to new requirements or changes. As a result, while automation can streamline the initial phases of development, additional effort is often required to refactor and optimize the code to ensure its maintainability over time.

A holistic evaluation of these dimensions ensures a more reliable assessment of the overall effectiveness of automated *vs*. manual development. The study's conclusions suggest that, although automation tools significantly improve efficiency, they do not eliminate the need for human intervention to ensure code quality. The combination of human oversight and automated generation produces the most reliable results, balancing the speed of automation with the precision and maintainability of manual development.

Future research should focus on enhancing the accuracy of AI tools to minimize human intervention and implement real-time feedback mechanisms. The scope of automation should also extend to additional stages of the software lifecycle, such as testing and deployment. Exploring tools that ensure higher code quality and maintainability is crucial. In the long term, AI models must be trained on more specific datasets to improve contextual understanding and mitigate biases. Lastly, applying these

tools to more complex projects and examining how developer roles evolve in increasingly automated environments will be essential.

5 Conclusions

The findings indicate that utilizing AI tools to automate tasks results in a reduction in development time by 40% to 51%, depending on the specific scenario. However, it is essential to acknowledge that the outputs generated by these tools require debugging, refinement, and adaptation. Consequently, the overall development process, as well as the tasks assisted by AI tools, become semi-automated workflows. These workflows necessitate continuous monitoring and the integration of complementary actions to ensure the accuracy and applicability of the generated results in software development projects.

During the coding phase, several adjustments were required to utilize the code segments generated by ChatGPT, largely due to inaccuracies in the initial outputs. To enhance precision, queries had to be refined with greater specificity to obtain more accurate responses. Additionally, an extensive review and debugging process was necessary to make the generated code suitable for practical application. Naturally, these additional steps increased the time spent, which was accounted for in the overall task duration.

The AUS tool, which leverages Natural Language Processing algorithms for generating textual content, fulfills its intended functions, such as producing user stories and estimating development effort. However, the generated content frequently requires review and debugging, as it often lacks completeness and precision in specific contexts. Enhancing the reliability of such tools will require further research, particularly in exploring emerging paradigms such as AI-Augmented Software Engineering and Software 2.0.

While AI-driven tools like ChatGPT provide significant time-saving benefits, particularly in the early stages of software development, there are notable trade-offs in terms of code quality, readability, and maintainability. Manual development remains indispensable for ensuring long-term reliability, as it produces more structured, readable, and maintainable code. Therefore, an effective development strategy should integrate both the efficiency of automation and the precision of human expertise, ensuring that projects benefit from the speed of automated processes while maintaining the robustness and accuracy required for sustainable software systems.

Acknowledgement: We would like to thank the Technical University of Cotopaxi and the Universidad del Valle for their support of this research process. In general, we thank all those who contributed to the achievement of the objectives of this work.

Funding Statement: This project is funded by the Universidad Técnica de Cotopaxi, Latacunga, Ecuador.

Author Contributions: The authors confirm that they contributed to the article as follows: conception and design: Verónica C. Tapia and Carlos M. Gaona; data collection: Verónica C. Tapia; analysis and interpretation of results: Verónica C. Tapia and Carlos M. Gaona; preparation of the draft manuscript: Verónica C. Tapia; supervision: Carlos M. Gaona. All authors reviewed the results and approved the final approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] D. Kovačević, "Use of ChatGPT in ESP teaching process," presentadoen 2023 22nd Int. Symp. INFOTEH-JAHORINA (INFOTEH), 2023. doi: 10.1109/INFOTEH57020.2023.10094133.

[2] J. Harika, P. Baleeshwar, K. Navya, and H. Shanmugasundaram, "A review on artificial intelligence with deep human reasoning," presentadoen 2022 Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC), IEEE, May 2022, pp. 81–84. doi: 10.1109/ICAAIC53929.2022.9793310.

[3] A. Yadav and H. Sondhi, "Systematic literature review on sustainable marketing and artificial intelligence," in *Proc. 17th INDIACom; 2023 10th Int. Conf. Comput. Sustain. Global Develop., INDIACom 2023*, New Delhi, India, 2023, pp. 583–588.

[4] K. Fuchs, "Exploring the opportunities and challenges of NLP models in higher education: Is Chat GPT a blessing or a curse?," *Front. Educ.*, vol. 8, 2023, Art. no. 1. doi: 10.3389/feduc.2023.1166682.

[5] S. Santhanam, T. Hecking, A. Schreiber, and S. Wagner, "Bots in software engineering: A systematic mapping study," *PeerJ Comput. Sci.*, vol. 8, no. 3, Feb. 2022, Art. no. e866. doi: 10.7717/peerj-cs.866.

[6] C. M. Chiesa-Estomba *et al.*, "Exploring the potential of Chat-GPT as a supportive tool for sialendoscopy clinical decision making and patient information support," *Eur. Arch. Oto-Rhino-L.*, vol. 281, no. 4, 2023, Art. no. 2082. doi: 10.1007/s00405-023-08104-8.

[7] H. Allam, J. Dempere, V. Akre, D. Parakash, N. Mazher and J. Ahamed, "Artificial intelligence in education: An argument of chat-GPT use in education," in *2023 9th Int. Conf. Inform. Technol. Trends (ITT)*, IEEE, May 2023, pp. 151–156. doi: 10.1109/ITT59889.2023.10184267.

[8] M. N. -U. -R. Chowdhury and A. Haque, "ChatGPT: Its applications and limitations," in *2023 3rd Int. Conf. Intell. Technol. (CONIT)*, IEEE, Jun. 2023, pp. 1–7. doi: 10.1109/CONIT59222.2023.10205621.

[9] Z. Jin, "Analysis of the technical principles of ChatGPT and prospects for pre-trained large models," in *2023 IEEE 3rd Int. Conf. Inf. Technol. Big Data Artif. Intell. (ICIBA)*, IEEE, May 2023, pp. 1755–1758. doi: 10.1109/ICIBA56860.2023.10165540.

[10] OpenAI, "API reference," Introduction," Accessed: Oct. 06, 2023. [Online]. Available: https://platform.openai.com/docs/introduction.

[11] A. Tekin, "The 10 best ChatGPT alternatives source: The 10 best ChatGPT alternatives," 2023. Accessed: Oct. 06, 2023. [Online]. Available: https://www.dopinger.com/blog/best-chatgpt-alternatives.

[12] S. Soobia *et al.*, ", Analysis of software development methodologies," *Int. J. Comput. Digit. Syst.*, vol. 8, no. 5, pp. 445–460, Jan. 2019. doi: 10.12785/ijcds/080502.

[13] R. S. Pressman, "Ingeniería del software un enfoque práctico," *Angew. Chem. Int. Ed.*, vol. 6, no. 11, pp. 951–952, 2000.

[14] Scrum Alliance," *These Are the Differences between Agile and Scrum, and How They Differ from Waterfall*." Denver, USA: Scrum Alliance Inc., 2021. Accessed: Sep. 13, 2023. [Online]. Available: https://resources.scrumalliance.org/Article/epic-agile.

[15] J. SCHIEL, "*What Is an Epic in Agile? Create Structure in the Product Goals before You*". Denver, USA: Scrum Alliance Inc., 2023. Accessed: Sep. 13, 2023. [Online]. Available: https://resources.scrumalliance.org/Article/epic-agile.

[16] J. Pelclová, "Documentation in agile development," 2014. Accessed: Sep. 01, 2023. [Online]. Available: https://is.muni.cz/th/vi2bu/Pelclova_MastersThesis.pdf.

[17] P. Z. A. Heck, "A quality framework for agile requirements: A practitioner's perspective," 2014, *arXiv:1406.4692*.

[18] R. Felip, "Cómo Escribir Buenas Historias De Usuario" in *How to Write Good User Stories*. Barcelona, España: Apiumhub, 2020 (in Spanish). Accessed: Sep. 01, 2023. [Online]. Available: https://apiumhub.com/es/tech-blog-barcelona/como-escribir-buenas-historias-de-usuario/.

[19] Greengard, "AI rewrites coding," *Commun. ACM*, vol. 66, no. 4, pp. 12–14, Apr. 2023. doi: 10.1145/3583083.

[20] A. B. Altamimi, "Effectiveness of ChatGPT in essay autograding," in *2023 Int. Conf. Comput. Electron. Commun. Eng. (iCCECE)*, IEEE, Aug. 2023, pp. 102–106. doi: 10.1109/iCCECE59400.2023.10238541.

[21] A. Ashraf and A. Imam, "ChatGPT's use case for software engineers," in *8th Int. Conf. Comput. Eng. Technol. (ICCET 2023)*, Institution of Engineering and Technology, 2023, pp. 487–492. doi: 10.1049/icp.2023.1537.

[22] M. Daun and J. Brings, "How ChatGPT will change software engineering education," in *Proc. 2023 Conf. Innov. Technol. Comput. Sci. Educ. V. 1*, New York, NY, USA, ACM, Jun. 2023, pp. 110–116. doi: 10.1145/3587102.3588815.

[23] N. Petrović, "Machine learning-based run-time DevSecOps: ChatGPT against traditional approach," in *2023 10th Int. Conf. Electr. Electron. Comput. Eng. (IcETRAN)*, IEEE, Jun. 2023, pp. 1–5. doi: 10.1109/IcETRAN59631.2023.10192161.

[24] S. Said *et al.*, "Experimental investigation of an interactive animatronic robotic head connected to ChatGPT," in *2023 5th Int. Conf. Bio-Eng. Smart Technol. (BioSMART)*, IEEE, Jun. 2023, pp. 1–4. doi: 10.1109/BioSMART58455.2023.10162099.

[25] N. Petrović, "Chat GPT-based design-time DevSecOps," in *2023 58th Int. Sci. Conf. Inform. Commun. Energy Syst. Technol. (ICEST)*, IEEE, Jun. 2023, pp. 143–146. doi: 10.1109/ICEST58410.2023.10187247.

[26] W. Uzair and S. Naz, "Six-tier architecture for AI-generated software development: A large language models approach," 22 Jun. 2023. doi: 10.21203/rs.3.rs-3086026/v1.