Effort Estimation in Agile Software Development - Is AI a Resourceful Addition?

Vasilka Saklamaeva[1,*], Luka Pavlič[1]

[1]University of Maribor, Faculty of Electrical Engineering and Computer Science, Koroška cesta 46, 2000 Maribor, Slovenia;

Abstract

Effort estimation in agile software development serves as a helpful step to estimate the progress expected in the future development iteration (sprint). Using different estimation methods, developers can estimate how much they can get done, and their assigned project managers can forward this information to the customer. This, in turn, helps to achieve better customer collaboration, respond to changes and deliver a working software in the determined time. With the rising adoption of Artificial Intelligence (AI) in nearly all steps of the Software Development Life Cycle (SDLC), our research motivation was to examine the addition of AI into the planning process of the SDLC, especially regarding effort estimation. This paper encompasses the difference between widespread and AI-driven estimation methods in agile development, as well as the results of their use in an experiment. Our findings suggest that the chosen AI tool significantly overestimated the time required for task implementation. Additionally, users expressed a preference for utilizing AI as a complement to their own expertise, rather than relying on it exclusively.

Keywords

sprint planning, generative AI, GAI, conversational AI, convo-AI, CAI

1. Introduction

Agile software development consists of various frameworks and practices that are guided by the values and principles defined in the Agile Manifesto and its twelve accompanying principles [1]. Some of the most popular frameworks are Scrum, Kanban, Feature-Driven Development (FDD), Extreme Programming (XP), Pair Programming and Lean Software Development [2], among others. During the (agile) Software Development Life Cycle (SDLC) there are many individual steps that spike the research curiosity of researchers and academicians, such as planning, development and testing, among others.

Effort estimation during the planning phase is a significant challenge in software development, as it involves the accurate prediction of effort, cost, and duration necessary for effective scheduling. The initial software development stages, characterised by high uncertainty and limited information, make this particularly difficult [3]. Factors like customer pressure, different demands and outdated estimation methods often result in overly optimistic estimates. This, in turn, impacts software delivery, its quality and the budget allocated to its development. Accurate estimation, however, enables efficient planning and resource management, timely delivery, strong customer relationships and consistent software quality among others [3]. In the continuation of this paper, we will focus solely on effort estimation process during the planning phase.

The research goals we set for this paper are the following:

- **RG1:** Define the goal of effort estimation in software development and how it can improve the planning aspect of development sprints.
- **RG2:** Define what effort estimation methods exist, and what are the potential benefits and limitations of their use.
- **RG3:** Explore the potential and limitations of Generative AI (GAI) and Conversational AI (CAI) in effort estimation.

SQAMIA 2024: Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, September 9–11, 2024, Novi Sad, Serbia

*Corresponding author.

vasilka.saklamaeva@um.si (V. Saklamaeva); luka.pavlic@um.si (L. Pavlič)

0009-0005-0221-4840 (V. Saklamaeva); 0000-0001-5477-4747 (L. Pavlič)

The rest of the paper is structured as follows. We begin with an overview of related works, as presented in Section 2. In Section 3, we provide a look into the chosen research methodologies to help achieve the set research goals. Section 4 consists of the different types of effort estimation methods that our research has identified, as well as the potential benefits and limitations their use carries. In Section 5 we present the preliminary results of an experiment, where we explored the acceptance of an AI tool during the effort estimation process. In Section 6 we present the limitations and threats to validity, and, in Section 7, we summarize the findings to address the set research goals and conclude the paper.

2. Related work

Through the exploration of existing research, several papers have addressed the addition of AI in software development, specifically effort estimation. A significant number of contributions addressed the addition of CAI/GAI during the *development* phase, however, there is a significant lack of research specifically regarding the *planning* phase. During our research, we rarely came across any empirical studies addressing their use in the planning phase, since most of the results were solution proposals or literature reviews. The lack of literature addressing this particular topic was also the main motivation for our research.

Arman et al. [4] introduce a conceptual solution designed to improve estimation accuracy and efficiency while minimizing manual effort and cost. Their approach involves extracting entities and their relationships from the system, modelling them as semantic triples and developing conceptual micro-services. To enable a more precise functionality breakdown and more accurate effort estimation, they incorporate prompt engineering with ChatGPT, which in turn enhances the accuracy of effort estimation.

A web-based AI chatbot named Alfred is a solution that Ebrahim et al. [5] introduce in their paper. It aids agile software release planning by employing two machine learning models to estimate the duration of tasks and recommend optimal resource assignments for project managers. It is also able to categorize the task estimates based on their confidence level into three groups: low, medium or high. Barcaui and Monat [6] explored the bigger picture of project management in their research. They compared GPT-4 and a human project manager in developing project plans, analyzing aspects like scope and schedule, as well as cost (effort) and resource estimation. The study finds that AI and human plans have complementary strengths and weaknesses and it highlights the importance of human expertise in refining AI outputs.

What differs our paper from these existing related works is the exclusive focus on effort estimation in combination with CAI/GAI tools.

3. Research method

During the outline of this paper, we defined two different research methods to achieve the set research goals. The research methods chosen were the following:

- **Literature review:** We looked at existing literature that encapsulates the use of effort estimation methods in software development. This allowed us to gather relevant bodies of knowledge regarding their benefits and limitations, as well as specialities for their use.
- **Initial experiment:** We carried out an experiment that explored the use of an AI tool for effort estimation during the planning phase of a development sprint.

The combination of both of these research methods gave us valuable insights on the role of effort estimation during the planning phases of a sprint. It also laid a foundation for our understanding of the addition of CAI/GAI in sprint planning and the overall acceptance users experience when using AI tools. The process of the literature search covered the following points:

- For the purpose of finding relevant literature, we limited ourselves to the results found in five academic digital libraries: IEEE Xplore, ACM, Springer, Science Direct and Web of Science.

Table 1
Software estimation models (Source: [7]).

	Algorithmic	Non-algorithmic
Category	Linear / Non Linear Discrete Models Multiplicative Models Power Function Model	Expert Judgment Analogy Learn based Soft computing

- The keyword used for the literature search were:
  1. ("All Metadata":„effort estimation")
  2. (("All Metadata":AI) OR ("All Metadata":"conversational AI") OR ("All Metadata":"generative AI") OR ("All Metadata":artificial intelligence) OR ("All Metadata":"GAI") OR ("All Metadata":"CAI")) AND ("All Metadata":„effort estimation")
- The literature reviewed was limited to the last five years (2019 to 2024).
- We limited the results to only peer-reviewed literature.
- All relevant literature needed to be written in English.
- Literature that was not directly connected to achieving the set research goals was excluded.

During the literature review we looked at different types of research contributions. To make sure our research represents the most recent developments and innovations in this area, we included papers published within the previous five years. By avoiding redundancy with earlier research, we had insight into the newest perspectives on emerging trends and technologies, especially for technologies such as GAI and CAI. The results of the literature search consisted mostly of other literature reviews, mappings and solution proposals. The results of the gathered knowledge are presented in the following Sections.

4. Effort estimation methods

The authors of [7] categorized effort estimation techniques into two primary classes: Algorithmic and Non-Algorithmic models. Table 1 provides a comparative analysis of these estimation model types.

The use of effort estimation methods, according to [8], mainly contributes to two areas: (1) to estimate the effort needed and (2) to support the estimation process. A graphical representation of the categorization of estimation methods based on their purpose is presented in Figure 1.

During the research process, we came across a popular division of effort estimation methods, namely *expert-based* and *data-based*. For the purpose of this research, we wanted to focus our attention to effort estimation approaches that use CAI/GAI, and those that do not. In some way this division complies with our literature findings, however, the *data-based* approaches, besides AI, usually mention specific Machine Learning algorithms, Neural Networks and different kinds of Regressions among others, that rely more on *crafting the right model or learning from past data*, which is not the subject of our paper.

4.1. Widespread estimation methods in agile development

A systematic literature review [9] which serves as a continuation of two others, pinpointed different estimation methods which can be found in agile software development and their corresponding metrics. The most popular estimation methods are mostly based on a subjective assessment. However, based on the gathered literature, we will take a deeper look at some known representatives, which are presented below.

- **Planning Poker** - As the most known and widely used effort estimation representative, Planning Poker consists of a few different steps. The first step in the estimating process is choosing a task to estimate. After reading the task description aloud, team members are free to discuss and

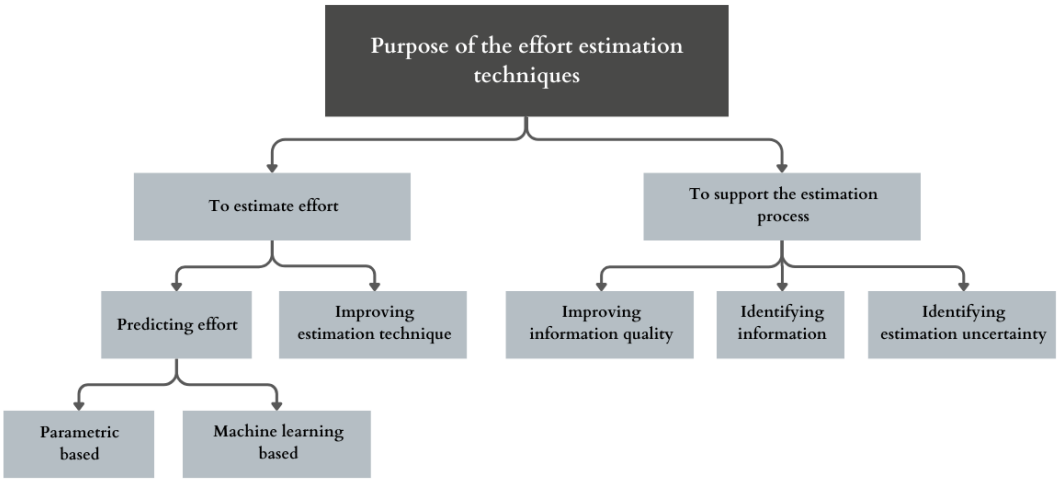

Figure 1: Categorization of effort estimation approaches based on their purpose (Source: [8]).

express questions. Once everyone is sufficiently informed, each person estimates the task on their own. The separate estimations are made public simultaneously and contrasted with one another. The task size that is suggested by the majority is selected as the final estimate. If not, the task is debated in public and the earlier stages are carried out again until a consensus is reached on the size [10].

- **Expert Judgement** - A popular representative for effort estimation is expert judgement, which draws on the knowledge and instincts of experienced experts. This method entails requesting estimates from people or organizations that have a specific subject expertise [11]. These individuals or groups evaluate the project's complexity and needs by drawing on their past experience and knowledge [12].
- **Use Case points method** - This model-based technique assesses the use cases' level of complexity within the system. A number of transactions is calculated for each use case and complexity weights (technical and environmental circumstances) are assigned, and, if needed, modified. The project's size estimate can be obtained from the total adjusted points, and different productivity factors can be used to convert that estimate into effort [11].

Although widespread effort estimation methods are extensively used and researched, there are some limitations to their application. Group dynamics may have an impact on the estimations, and most of the factors are related to human qualities, which do not necessarily provide accurate estimates. As Chitrak et al. [13] defined in their research, Planning Poker lacks analytical estimation and there is little literature available to grasp the efficacy of this approach. On the other hand, regardless of the expertise or influence within the team, Planning Poker helps ensure that everyone's voice is heard. This, in turn, assures that everyone has a better general comprehension of the duties, as well as gathering the results in a timely manner [10].

According to [13], the Expert Judgement approach, as the sole name says, suffers from bias. The previous experience of the expert and their subjective variability, can skew the estimates' accuracy, and therefore, it is not considered as a reliable estimation approach [12]. One benefit of using this approach is that it produces a fast result.

The Use Case Points Method offers a methodical technique to estimate effort based on functional requirements, which is especially helpful for projects that are just getting started [11]. It has proven to generate accurate and consistent predictions, particularly when paired with previous project data to adjust the productivity aspects unique to the company or project setting.

Table 2
Summary of limitations and benefits of widespread effort estimation methods. (Sources: [10, 11, 12, 13]).

	Benefits	Limitations
Planning poker	inclusion of every team member's opinion better comprehension of duties fast results	lacks analytical estimation little literature on it's efficacy
Expert judgement	contextual understanding fast results	experts are prone to bias not fully reliable estimates
Use case points	accurate and consistent predictions better results through historical data	subjectivity in weight assignments dependency on detailed requirements

A summary of the identified benefits and limitations of the use of widespread effort estimation methods is presented in Table 2. Even though widespread effort estimation methods have been more cumulatively used from the rise of agile software development, the rising potential of the use of AI tools in this area poses as a competition.

4.2. Methods using CAI/GAI

AI technologies used to produce natural, engaging, and human-like discussions between humans and machines are known as conversational AI (CAI). These AI systems, usually chatbots, virtual assistants, or voice assistants, can understand and react to human language in a manner that mimics natural speech [14].

The potential of AI and Large Language Models (LLMs) across all phases of the SDLC is rapidly advancing, offering opportunities to enhance it's various aspects. Generative and conversational tools, such as those powered by models like GPT, are being integrated more frequently into software development processes. As for their integration into the planning process, and in the phases of effort estimation specifically, these tools can be a helpful addition to numerous areas, including:

- Analysis of historical project data,
- Gathering requirements through Natural Language Processing (NLP),
- Help optimize resource allocation and aid the management of risks,
- Produce accurate effort estimates by identifying patterns and predicting problems,
- Help project managers make informed decisions.

Other forms of AI, such as supervised learning (with correct labeling), unsupervised learning (clustering/without labeling), and reinforcement learning (interacting to receive feedback), also have an impact on software development in addition to GAI (generating new data from existing data). As of right now, there are tools available to help with different execution models' simulation, testing, documentation, debugging, code generation, and discovery. These many forms of AI will be employed in a variety of roles in the future [15].

As stated in [15], developer productivity for GAI may increase rapidly, but this goes beyond simple processes or tooling. Significant growth appears to be occurring in the areas of repetitive jobs, the initial creation of prototypes and development using templates, to name a few. However, there hasn't been a good outcome from projects requiring a high level of creativity or orchestration. This, in turn, may mean that the addition of AI in effort estimation may not be the most adequate step, since it does, to some degree, operate on creativity, orchestration and team skills. Other challenges that arise with the specific use of open source CAI/GAI tools are ethical concerns. Issues like data privacy, bias and misinformation have to be properly addressed [14].

Recent research by Gartner [16] has pinpointed different AI-augmented DevOps tools across the SDLC. The area of our interest - Planning and Collaboration, housed six different AI-driven tools,

which, according to this report, have the ability to revolutionize the planning process during software development. A short description of three chosen representatives and their functionalities are presented in the following.

1. **Amazon Q** - Amazon Q is a generative AI-powered assistant that helps developers by taking on time-consuming jobs including planning, coding, testing, debugging, and application maintenance. Because Amazon Q can interact with a variety of enterprise systems and repositories, it can expedite decision-making and problem-solving processes by automating common operations and providing timely, relevant information [17].
2. **Atlassian Intelligence** - Atlassian Intelligence uses generative AI to automate processes like creating test plans, compiling project documentation, and summarizing meeting minutes. It facilitates real-time assistance via virtual agents in Jira Service Management, offers AI-powered insights and suggestions, and enhances teamwork and communication [18].
3. **GitLab Duo** - GitLab Duo is a suite of AI-powered capabilities that provides features including real-time code explanations, automatic test generation, and code suggestions to increase development efficiency. It contributes to security by explaining vulnerabilities and proposing fixes, with the ultimate goal of streamlining procedures and lessening developers' cognitive load [19].

After considering Gartner's suggestion and conducting our own tool exploration and evaluation, we have chosen to utilize GitLab Duo, which is integrated into GitLab Enterprise. The number of AI-driven features this specific tool enables was the primary factor in our decision. GitLab Duo Chat, which functions as an integrated CAI, and the ability to generate issue descriptions were important considerations in our selection.

5. Initial experiment results

To enrich the takeaways and lessons learnt from this research, we conducted an experiment that focused on the acceptance of the adoption of a CAI/GAI tool during the effort estimation process during sprint planning.

The experiment was conducted in a controlled laboratory setting with **eighteen** developers divided into six groups of three. The study spanned throughout two weeks on three separate sessions. The participant profile was the following:

- Enrolled in the second year of undergraduate studies in the Informatics and Data Technologies program.
- An average of one to three years experience with software development.
- Good knowledge of software development planning methods.
- Average understanding of AI and ML concepts.

The participants were given access to the same foundational project and the same eight user story backlogs. Examples of two user stories used in this context are presented in Figure 2.

The developers' assignments spanned through multiple steps. The first step was to break each user story into tasks and provide task time estimates using the tool **GitLab Duo**. For the purpose of this study, all participants had access to the Ultimate SaaS plan. The developers were then instructed to transfer all generated output, as well as the prompt used, in a separate spreadsheet.

The next step was the implementation phase, where they had to implement the chosen user stories to the foundational project. The developers were additionally instructed to keep track of the actual implementation time of each task, without having access to their prior estimations. They were also provided access to additional AI-driven functionalities in GitLab Duo, which they were permitted to use.

This experiment gave us some insights as to how CAI/GAI tools can help developers during the software planning and development processes. We noticed a Significant difference between the predicted and actual times of implementation, which are presented in Figure 3.

User Story #2

Monthly reporting **of measurements outside the allowed tolerances.**

- Add the option "monthly report" to the application menu.
- The monthly report should include all measurements from the last 30 days that were **outside the tolerances.**
- They should be sorted by deviation from the allowed temperature in **descending order.**
- The report should also include the number of non-compliant measurements for each product.

User Story #8

Switch between different languages: DE / EN / SI.

- Introduce settings in the application menu where it is possible to switch between German, **English,** and Slovenian languages.
- Depending on the selection, the **appropriate language** for the user interface should be displayed throughout the entire application.

Figure 2: User Story examples.

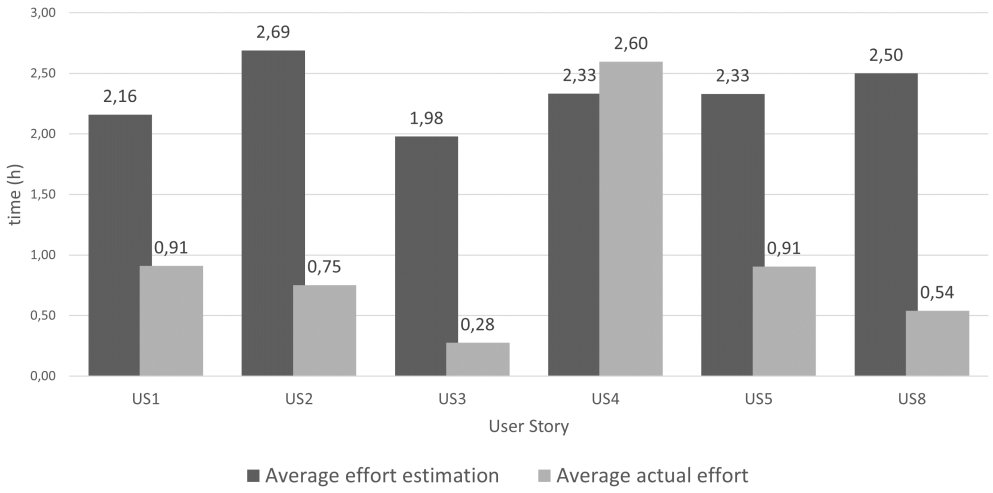

Figure 3: Predicted vs. actual effort for each user story.

As can be seen on the Figure, in most cases, the AI tool estimated a greater amount of effort in comparison to the actual implementation effort. The only exception is User Story 4 ("Switch between different measurement systems °C/°F"), which was, according to the experiment participants, more demanding in comparison to the others. As mentioned in the beginning of this Section, the developers had a backlog of 8 User Stories, but at the end of the experiment we had results for only 6 of them. The reason is quite clear - most of the groups didn't implement User Stories 6 and 7, so we didn't have a realistic result for the predicted values. Regarding the performance (number of implemented user stories), the groups performed average, implementing 3,33 user stories per group. The results are presented in Figure 4.

The biggest takeaway from our experiment was that nearly all developers expressed that they would prefer to use a combination of widespread, (such as Planning Poker and Expert Judgement) and AI-driven approaches (ex. different CAI/GAI tools) during the software planning and development phases, which we also observed in the reviewed literature.

6. Limitations and threats of validity

The addition of AI in the planning phase of the SDLC is attracting and requesting more and more research to be done. AI's potential during the effort estimation phase is a different story. There is an extensive amount of literature that explores the fusion of different ML algorithms, regressions and so on, scoping from predicting outcomes to helping all included to make better decisions. That being said, our research was primarily focused on the addition of CAI/GAI tools, which we concluded that is not yet thoroughly researched.

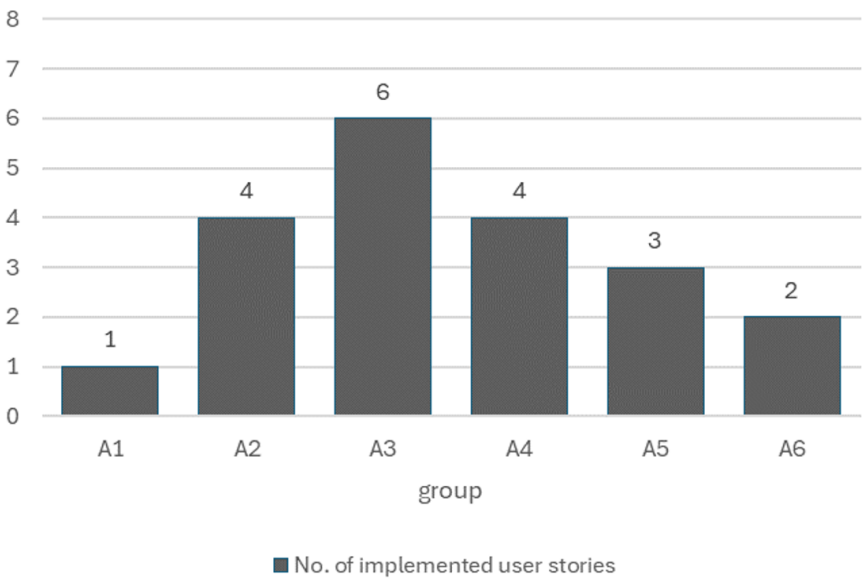

Figure 4: No. of implemented user stories per group.

Another limitation addresses the realization of the experiment. Namely, we were constrained to include a number of participants that we had access to in our academic settings. This, in turn, may have potential threats on the generalization of our findings. We also limited ourselves to the tool GitLab Duo for the purpose of the experiment execution, since it offers the widest amount of functionalities that matter to our research area.

Because our research and literature selection were done in mid-2024, it is crucial to point out that even with our best efforts, there is always a chance that we could have overlooked a crucial paper. Recent innovations that were not taken into consideration during our research could potentially have an impact on our findings.

7. Conclusion and future work

This paper delved into the exploration of effort estimation methods and the addition of CAI/GAI into the planning phase of the SDLC. The findings derived from a literature review in combination with an experiment, point out the potential CAI/GAI has in this area. Even though this specific subject is not researched well enough, and AI-driven tools have not yet reached a level of maturity to fully replace human estimators, we believe that the fusion of these technologies into the SDLC may provide improved results. With this being the main focus of our paper, we explored a few different research goals, whose findings we sum up in the following.

RG1: ***Define the goal of effort estimation in software development and how it can improve the planning aspect of development sprints.***

In order to ensure proper project planning, budgeting, and resource allocation, effort estimating is a crucial step in the SDLC. Its goal is to predict the time, resources, and labor required to accomplish a software project. Using estimation approaches, as discussed in Section 1, can lead to a variety of benefits, including accurate work estimations, effective planning and resource management, timely delivery, strong customer relationships, and consistent software quality. To achieve this, numerous estimation methods exist, spanning from taking into account only one persons' (expert's) opinion, to considering every team member's opinion, to using historical data to predict future outcomes.

RG2: ***Define what effort estimation methods exist, and what are the potential benefits and limitations of their use.***

With the help of our literature review results, we delved into the exploration of different effort estimation

methods. Our findings, presented in Section 4, were limited to two groups: algorithmic and non-algorithmic estimation models, as well as widely used methods and approaches with CAI/GAI. We additionally explored the numerous benefits and limitations of using the most widely used methods. Widespread estimating techniques might be time-consuming and may not adequately account for the complexities of modern projects, but they have the advantage of being based on past data and expert judgement, offering an organized and predictable approach. In contrast to this, CAI/GAI can provide estimates rapidly by evaluating large volumes of data and adjusting to real-time inputs, but they might not have the same contextual understanding and domain-specific knowledge as human estimators.

RG3: ***Explore the potential and limitations of Generative AI and Conversational AI in effort estimation.***

Despite being in their early development phases, CAI/GAI tools can influence various stages of the SDLC. This, in turn, raises some limitations like the ability to produce an original solution (since these kinds of tools primarily work on past data) and data privacy and availability. Our findings for this specific research goal are presented in Sections 4.2 and 5. Since this was the main goal of our paper, with the help of a literature review and an experiment, we discovered potential into the fusion of CAI/GAI into effort estimation processes, especially in combination with the human factor. The results of our experiment specifically raise the fact that users would like to use CAI/GAI tools to assist them in the estimation process, with them having the upper hand.

This paper lays important groundwork for several propositions for future work. Firstly, a potential research idea would hover on the identification of different tools and their functionality comparisons, as well as their potential to support different phases of the SDLC. Another potential future work, based on our experiment findings, is conducting an empirical study that combines both widespread and AI-supported effort estimation approaches.

In conclusion, there is a great deal of room for improvement in the accuracy of effort estimation when CAI/GAI tools are integrated during the planning phase of the SDLC. Compared to widespread methods, CAI/GAI can assess historical data, project requirements, and team dynamics more efficiently since they use advanced ML algorithms and NLP. Our experiment's results, however, show a poor estimation precision, indicating that participants would rather use AI in addition to their experience than depending solely on it. This conclusion is especially significant for agile development frameworks, where quick feedback and flexibility are critical components.

Acknowledgments

The authors acknowledge financial support from the Slovenian Research and Innovation Agency (Research Core Funding No. P2-0057).

References

[1] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, Manifesto for agile software development, 2001. URL: https://agilemanifesto.org/.

[2] E. Mircea, Project management using agile frameworks, Economy Informatics Journal 19 (2019) 34–43. URL: https://agilemanifesto.org/. doi:10.12948/ei2019.01.04.

[3] P. Pospieszny, B. Czarnacka-Chrobot, A. Kobylinski, An effective approach for software project effort and duration estimation with machine learning algorithms, Journal of Systems and Software 137 (2018) 184–196. URL: https://www.sciencedirect.com/science/article/pii/S0164121217302947. doi:https://doi.org/10.1016/j.jss.2017.11.066.

[4] A. Arman, E. Di Reto, M. Mecella, G. Santucci, An Approach for Software Development Effort Estimation Using ChatGPT, in: 2023 IEEE International Conference on Enabling Technologies:

Infrastructure for Collaborative Enterprises (WETICE), 2023, pp. 1–7. URL: https://ieeexplore.ieee.org/document/10477777/?arnumber=10477777. doi:10.1109/WETICE57085.2023.10477777.

[5] E. Ebrahim, M. Sayed, M. Youssef, H. Essam, S. A. El-Fattah, D. Ashraf, O. Magdy, R. ElAdawi, Ai decision assistant chatbot for software release planning and optimized resource allocation, in: 2023 IEEE International Conference On Artificial Intelligence Testing (AITest), 2023, pp. 55–60. doi:10.1109/AITest58265.2023.00018.

[6] A. Barcaui, A. Monat, Who is better in project planning?generative artificial intelligence or project managers?, Project Leadership and Society 4 (2023) 100101. URL: https://www.sciencedirect.com/science/article/pii/S2666721523000224. doi:https://doi.org/10.1016/j.plas.2023.100101.

[7] A. Khatibi Bardsiri, S. M. Hashemi, Software effort estimation: A survey of well-known approaches 3 (2014).

[8] J. Pasuksmit, P. Thongtanunam, S. Karunasekera, A systematic literature review on reasons and approaches for accurate effort estimations in agile, ACM Comput. Surv. 56 (2024). URL: https://doi.org/10.1145/3663365. doi:10.1145/3663365.

[9] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, E. Insfran, An update on effort estimation in agile software development: A systematic literature review, IEEE Access 8 (2020) 166768–166800. doi:10.1109/ACCESS.2020.3021664.

[10] P. Sudarmaningtyas, R. B. Mohamed, Extended planning poker: A proposed model, in: 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2020, pp. 179–184. doi:10.1109/ICITACEE50144.2020.9239165.

[11] C. Carbonera, K. Farias, V. Bischoff, Software development effort estimation: A systematic mapping study, IET Software 14 (2020). doi:10.1049/iet-sen.2018.5334.

[12] P. G. F. Matsubara, I. Steinmacher, B. Gadelha, T. Conte, Much more than a prediction: Expert-based software effort estimation as a behavioral act, Empirical Software Engineering 28 (2023) 98. URL: https://doi.org/10.1007/s10664-023-10332-9. doi:10.1007/s10664-023-10332-9.

[13] V. C. Dave, P. Abhishek, K. Utkarsh, An efficient framework for cost and effort estimation of scrum projects, International Journal for Research in Applied Science and Engineering Technology 9 (2021) 1478–1487. doi:10.22214/ijraset.2021.39030.

[14] G. Bansal, V. Chamola, A. Hussain, M. Guizani, D. Niyato, Transforming conversations with ai—a comprehensive study of chatgpt, Cognitive Computation (2024). URL: https://doi.org/10.1007/s12559-023-10236-2. doi:10.1007/s12559-023-10236-2.

[15] J. Sauvola, S. Tarkoma, M. Klemettinen, J. Riekki, D. Doermann, Future of software development with generative ai, Automated Software Engineering 31 (2024) 26. URL: https://doi.org/10.1007/s10515-024-00426-z. doi:10.1007/s10515-024-00426-z.

[16] Manjunath Bhat, Cameron Haight, Bill Blosen, How Platform Engineering Teams Can Augument DevOps With AI, 2024. URL: https://www.gartner.com/document/5083131?ref=solrAllImg&refval=420021645&.

[17] Antje Barth, Introducing Amazon Q, a new generative AI-powered assistant (preview) | AWS News Blog, 2023. URL: https://aws.amazon.com/blogs/aws/introducing-amazon-q-a-new-generative-ai-powered-assistant-preview/, section: Announcements.

[18] Ryan Narragon, Unleash every team: Using AI to transform your organization - Work Life by Atlassian, 2024. URL: https://www.atlassian.com/blog/announcements/team-24-using-ai-to-transform-your-organization.

[19] GitLab Duo | GitLab, 2024. URL: https://docs.gitlab.com/ee/user/gitlab_duo/.