An LLM-based multi-agent framework for agile effort estimation

Thanh-Long Bui
Decision System Lab
School of Computing and IT
University of Wollongong, Australia
tlb959@uowmail.edu.au

Hoa Khanh Dam
Decision System Lab
School of Computing and IT
University of Wollongong, Australia
hoa@uow.edu.au

Rashina Hoda
Faculty of Information Technology
Monash University, Australia
rashina.hoda@monash.edu

***Abstract*—Effort estimation is a crucial activity in agile software development, where teams collaboratively review, discuss, and estimate the effort required to complete user stories in a product backlog. Current practices in agile effort estimation heavily rely on subjective assessments, leading to inaccuracies and inconsistencies in the estimates. While recent machine learning-based methods show promising accuracy, they cannot explain or justify their estimates and lack the capability to interact with human team members. Our paper fills this significant gap by leveraging the powerful capabilities of Large Language Models (LLMs). We propose a novel LLM-based multi-agent framework for agile estimation that not only can produce estimates, but also can coordinate, communicate and discuss with human developers and other agents to reach a consensus. Evaluation results on a real-life dataset show that our approach outperforms state-of-the-art techniques across all evaluation metrics in the majority of cases. Our human study with software development practitioners also demonstrates an overwhelmingly positive experience in collaborating with our agents in agile effort estimation.**

***Index Terms*—Agile software development, effort estimation, large language models, multi-agent systems, planning poker, human-AI collaboration**

I. Introduction

Effort estimation is an important aspect of software project management, especially in the planning and monitoring stages. Its accuracy may significantly impact project outcomes – underestimating effort can lead to missed deadlines and budget overruns, while overestimating can reduce resource efficiency and diminish an organisation's competitive edge [1]. In modern agile software development, software is built through incremental releases and iterative cycles, each of which requires the completion of a number of *user stories*. As a result, the focus has shifted towards estimating the effort required to complete individual user stories rather than the entire project. It is now standard practice for agile teams to review and estimate each user story in the product backlog, enabling them to plan, prioritise and deliver value efficiently [2].

At present, agile teams mostly rely on experience, subjective assessments, and consensus to estimate user stories (e.g. planning poker [3], T-shirt sizing and dot voting). However, these techniques can result in significant inaccuracies, and more notably, inconsistencies across different estimates, even within the same project and the same team [4]. Recent approaches (e.g. [5]–[9]) have proposed a range of artificial intelligence (AI) models that can learn from a team's previous estimates and predict the effort for new user stories. Positive results from those approaches suggest that it is possible to build an AI-enabled prediction system to support agile teams in estimating user stories, enabling them to be consistent in their estimates.

However, those existing approaches suffer from several major limitations. Most of them are inherent "black box" models, and do not provide *explanation* for their estimates. Thus, it is difficult for agile teams to understand them and interpret their estimation. In addition, current agile estimation practices (such as the widely-used Planning Poker [3] method) are *consensus-based*: each team member provides an estimate and a consensus estimate is reached after several rounds of discussions and re-estimation. The purpose here is not only to produce an estimate, but also to promote team conversation and collaboration, leverage the cross-disciplinary nature of agile teams, and help uncover hidden complexities or overlooked aspects of the work. Existing AI-driven estimation approaches lack these important capabilities. They are *not* able to justify their estimates, and *cannot* participate in discussions with human developers to reach an agreed-upon estimate. These critical limitations have resulted in industry reluctance to adopt or deploy them in practice [10].

This paper aims to address the above limitations by proposing a multi-agent framework for agile estimation. We leverage the recent powerful LLM technologies to build a *new* class of LLM-based agents for agile effort estimation (hereafter called **S**oftware **E**ffort **E**stimation **Agent** or ***SEEAgent*** for short). Each ***SEEAgent*** is capable of not only estimating user stories but also providing justification for its estimates. LLMs often suffer from *hallucinations* where the generated information appears plausible but incorrect. To address this problem, we propose a combination of prompt engineering and fine-tuning methods to enrich our agent with knowledge specific to the context of a project through its past user stories. In addition, each agent in our framework can be further enhanced with knowledge specific to its role(s). Thus, our multi-agent approach can combine diverse expertise to lead to more realistic estimates through considering multiple angles of a user story.

Another key *novelty* of our approach is that our ***SEEAgent*** can coordinate and communicate with other agents and hu-

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00090

man developers in an agile estimation process – this is a significant departure from the existing approaches. In our multi-agent framework, each individual ***SEEAgent*** and human team member can share their perspectives and assumptions on the complexities of a user story before providing their own estimate. If the estimates differ widely, the agents and human team members can discuss the reasons behind the differences. Based on these discussions, our ***SEEAgent*** is then able to re-estimate, resulting in a more informed and consensus-based estimate. This may improve the accuracy and reliability of effort estimates by leveraging collective team knowledge and reducing biases. Furthermore, through enabling human-AI collaboration, our approach helps agile teams facilitate conversation, uncover hidden insights, and build team consensus – a key purpose of agile effort estimation.

We evaluate ***SEEAgent*** on two aspects: (i) its accuracy compared to prior state-of-the-art (SOTA) deep learning approaches for estimating user stories; and (ii) how well it works with human developers in the agile estimation process. In terms of accuracy, we used the same dataset and benchmarks as in the SOTA approaches, and the results show our ***SEEAgent*** outperforms SOTA in the majority of the cases. With respect to collaborating with humans, we conducted a human study where participants were asked to work with our ***SEEAgents*** to estimate user stories for a given software project. Feedback from the participants shows a strong positive experience working effectively with our agents. The participants found that the use of our agents benefits development teams to reach consensus, bridge the knowledge gap and promote communication between team members. Overall, the participants agreed that they trust our ***SEEAgent*** and would recommend using it in practice.

The structure of the paper is as follows. We present the details of ***SEEAgent*** in Section II. The experimental setup and results are reported in Section III. Related work is discussed in Section IV. Finally, we conclude this paper and outline future work in Section V.

II. Approach

We propose **S**oftware **E**ffort **E**stimation **Agent** (***SEEAgent***), a new agentic framework designed for aiding agile teams in the process of effort estimation and sprint planning [3]. ***SEEAgent*** aims not only to provide accurate and consistent estimates, but also to facilitate informational interactions with agile team members through offering insights and relevant project information beyond mere numerical predictions.

A. Architectural overview

We formulate the problem of agile estimation as predicting the effort E (often expressed in terms of *story points*) required to implement a user story described by a natural language specification Q.

Let $\mathcal{D} = \{(Q_i, E_i)\}_{i=1}^{n}$ denote a historical dataset of previously estimated user stories and their associated efforts. Given a new user story Q', the objective is to estimate the required effort $\hat{E}$ such that $\hat{E} \approx E'$, where E' is the unknown true story points.

Each ***SEEAgent*** $\mathcal{A}_i$ (see Figure 1) operates as an autonomous entity tasked with deriving an effort estimate $\hat{E}$ for a given user story Q'. Internally, ***SEEAgent*** features a modular architecture comprising four principal components: short-term memory, long-term memory, action module, and communication manager. The agent interacts with the software development environment through multiple means: (i) learns knowledge about projects through fine-tuning [11] or in-context learning on historical user story datasets D; (ii) generates estimations for new user stories Q'; and (iii) communicates with human practitioners and other agents using its action module. This architecture enables ***SEEAgent*** to maintain relevant information, learn from past estimates, and provide informed effort estimates while facilitating interaction with development team members and peer agents.

The internal reasoning process of an agent unfolds over multiple phases, influenced by both local knowledge and peer interactions. Given a user story Q', the agent retrieves relevant information from both memory modules. In short-term memory $\mathcal{M}_s$, the agent stores the user story Q', conversation history H, and peer estimates $\hat{E}_j$. From long-term memory $\mathcal{M}_l$, the agent draws relevant examples, patterns, or rationale stored in its latent space.

We follow the Planning Poker practice in agile, where estimating a user story is often conducted in multiple rounds of discussions. At each round t, the agent computes its updated effort estimate $\hat{E}^{(t)}$ based on three components:

- **Initial prompt reasoning:** The agent uses $\mathcal{M}_l$ to analyse Q' and determines whether clarification questions are needed before making an estimate.
- **Contextual feedback integration:** For $t > 0$, the agent observes peer predictions $\{\hat{E}_j^{(t-1)}\}_{j \neq i}$ and prior conversation trace $H^{(t-1)}$, then uses $\mathcal{M}_l$ to analyse them and determines if additional clarification questions are needed before updating its estimate.
- **Effort generation:** Based on available information and clarifications, the estimate is derived:

$$\hat{E}_i^{(t)} = \mathcal{M}_l(\cdot|Q', \mathcal{M}_s, \{\hat{E}_j^{(t-1)}\}_{j \neq i}, H^{(t-1)})$$

The agents continue this process until convergence is achieved across all participants or the maximum round limit r is reached.

This architecture allows agent behaviour to be explainable and auditable, with the communication trace and reasoning process offering insight into how $\hat{E}_i^{(t)}$ was derived. We will now discuss the major components of our ***SEEAgent*** architecture in detail.

B. Long-term memory

The long-term memory component, with reasoning capabilities powered by general-purpose LLMs, serves as the central information processing unit. It pertains to project-specific knowledge and general world knowledge. Given the stateless nature of language models [12], the information in

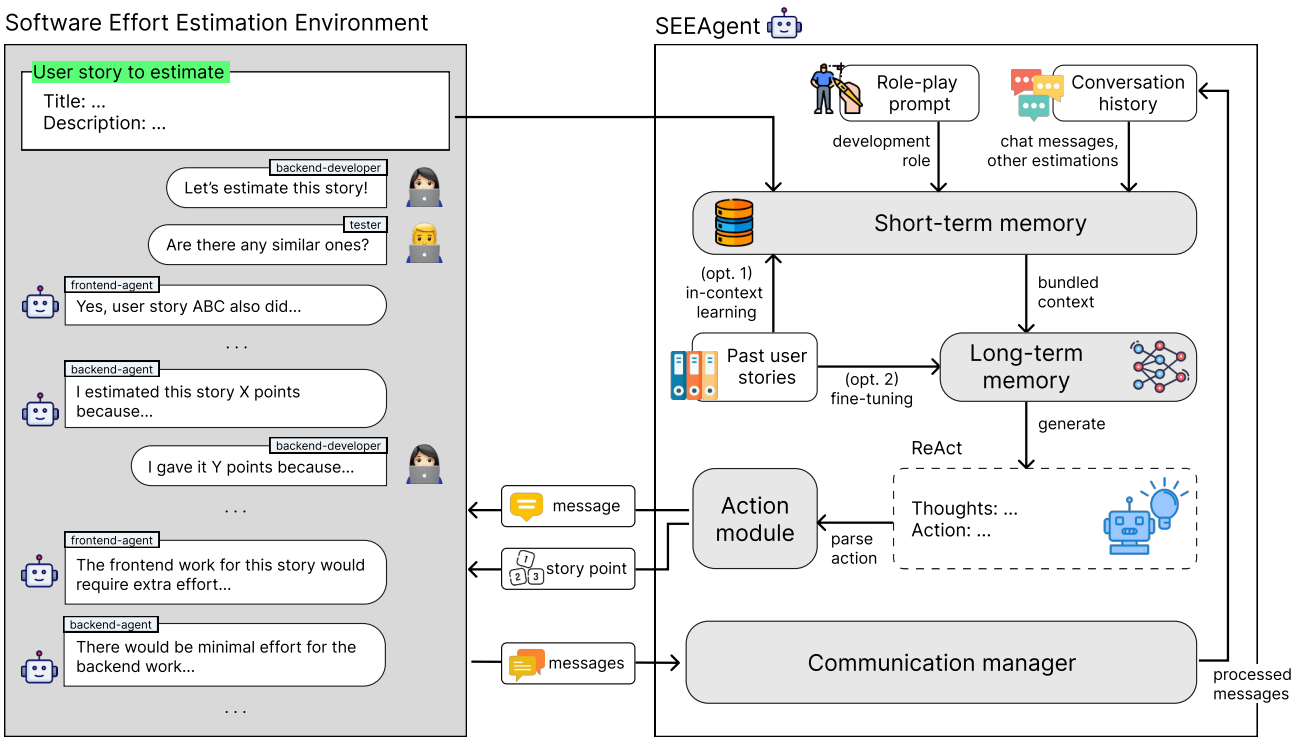

Fig. 1. Architecture and example conversations between ***SEEAgents*** and humans

long-term memory is prone to being outdated, necessitating continual updates. To address this, the process of enriching the module with the latest information about the project must be routinely updated through fine-tuning using newly estimated user stories. This continuous refinement ensures that it remains aligned with evolving project requirements. Furthermore, the integration of fresh information helps the agent to be aware of changes and provide more informed and context-aware responses, thereby making them more useful in supporting human practitioners in agile estimation sessions.

Story point estimation is inherently project-relative [5], with identical user stories potentially receiving different effort estimations across development teams. Therefore, to accommodate this variability, our approach adapts the base LLM model through two modes: fine-tuning for established projects with sufficient historical data, and in-context learning [13] for new projects. The fine-tuning process is detailed below, while the in-context learning process is discussed in Section II-C.

Large model fine-tuning is computationally costly and requires a substantial amount of data and computational resources. Due to the constraint of computation resources, we leveraged parameter-efficient fine-tuning (PEFT) techniques (e.g. [11], [14]–[17]). Specifically, we employed Low-rank Adaptation (LoRA) [11] to adjust the chosen base model's weights. LoRA introduces new trainable parameters in the form of low-rank decomposition matrices to the attention layers and freezes the pre-trained model weights. This allows the model to learn project-specific information while drastically reducing the number of trainable parameters. To further reduce the computation requirements, we applied QLoRA [18], 4-bit quantisation in combination with LoRA. QLoRA compresses the model weight precision from BF16 to NF4, helping reduce the memory usage four times, in theory, without significantly impacting the performance of the model.

In this study, we employed the *Llama-3.1-8B-Instruct* model as our base model. After LoRA is applied to the model, 20,971,520 additional parameters are introduced while the original 8 billion parameters are frozen. These newly introduced parameters are then updated through the Supervised Fine-tuning (SFT) technique [19] trained on historical user stories with ground-truth estimates.

SFT allows the model to learn downstream tasks by training it on labelled input-output pairs relevant to the tasks. The input will be a structured prompt containing a user story, and its story point will be the output.

The effectiveness of fine-tuning relies on the quality and design of the prompt used during training. Since our objective is to teach the base model to estimate story points for agile effort estimation, we crafted a prompt that closely resembles how a human practitioner would respond to such queries. The prompt is structured following a conversation format with clearly separated system instructions, user input (the user story), and assistant answer (the estimated story points). This structure enables the model to learn the semantic relationship between a user story and its corresponding numerical estimate while preserving the conversational context essential for agent

and human interactions.

The system message sets the context by assigning a specific role to the model: an expert software development effort estimator. The user message presents the user story, and the assistant message entails a succinct response, making it easier for the model to learn from examples. Through training with this prompt structure across multiple examples with varying story complexities and associated effort points, the model gradually learns to associate different types of user stories with appropriate effort levels. The complete prompt format used for fine-tuning is shown in Listing 1.

```
[system]
You are an expert software development effort
    estimator. Given a software development issue
    summary, predict the effort in story points.

[user]
### Summary:
{{USER STORY}}

[assistant]
My estimated story point is: {{ESTIMATED POINT}}
```

Listing 1. Prompt for fine-tuning model

C. Short-term memory

Short-term memory functions as immediate information storage for an estimation session. It maintains a context consisting of a development role, recently received messages, reasoning states, and relevant project artifacts. All these data are structured following the ReAct [20] prompting template. This template leverages interleaving explicit thoughts, proposed actions, and observations, making the agent's thought process auditable and traceable. Furthermore, using this prompting technique allows the agent to carry out autonomous decision-making and action-taking capabilities imposed by the implemented language model.

```
1  {{ROLE-PLAY PROMPT}}
2
3  You are an agent that participates in a planning
       poker session with other agents and human
       players.
4
5  ## Actions
6  You have access to actions that you can use.
       Think carefully about which actions to use.
7
8  Available actions:
9  {{ACTIONS DESCRIPTION}}
10
11 ## Input format
12 You will receive game state and chat history
       updates.
13
14 ## Output format
15 Always follow this format:
16
17 Thought: your reasoning about what to do next
18 Action: the_action_name
19 Action Input: {{"param1": "value1", "param2": "
       value2"}}
20
21 ## Planning poker rules
22 {{GAME RULE}}
23
24 ## Strategy
25 You objective is to communicate via chat with
       other players to come up with the best
       estimation for the given user story. For
       better estimation, you should base your
       estimation on the past user stories and
       similar user stories.
26
27 In the first round, you can either make an
       estimation or ask questions to clarify the
       user story. For the following rounds, you
       should justify your estimation based on
       similar user stories and consider others'
       estimations.
28
29 ## Past user stories
30 For reference, you can use the following past
       user stories to help you make your estimation
       :
31
32 {{PAST USER STORIES}}
33
34 ## User story to be estimated
35 {{USER STORY}}
```

Listing 2. SEEAgent system prompt

The system prompt of ***SEEAgent*** in Listing 2 comprises several sections:

- Line 1-3: The role-play prompt assigns specific profiles in software development to an agent, influencing the agent's responses to match the desired expertise. This role-play configuration consists of two principal elements: a dynamic software development expertise profile configured during agent initialisation, and a static profile defining ***SEEAgent***'s fundamental purpose as a planning poker session participant.
- Line 5-19: The prompt uses the ReAct prompting technique [20] through dedicated sections for actions, input specifications, and output formatting.
- Line 21-27: To ensure appropriate agent behaviour within the designed planning poker environment, dedicated sections detail the operational rules and strategic guidelines. These sections provide the necessary behavioural constraints and decision-making frameworks.
- Line 29-32: Our approach includes an optional in-context learning component through the incorporation of past user stories, providing an alternative approach when LLM fine-tuning is not feasible or when the project is in its early stage and the number of past user stories with ground-truths is limited.
- Line 34-35: The final section details the target user story designated for effort estimation.

D. Action and communication

The grounding of an ***SEEAgent*** is conducted in part by its Action module, which enables it to interact with the software development environment. The Action module acts as an intermediary between the internal of the agent, the reasoning engine, and the environment. This module is integral to ***SEEAgent***'s ability to elicit information and respond to

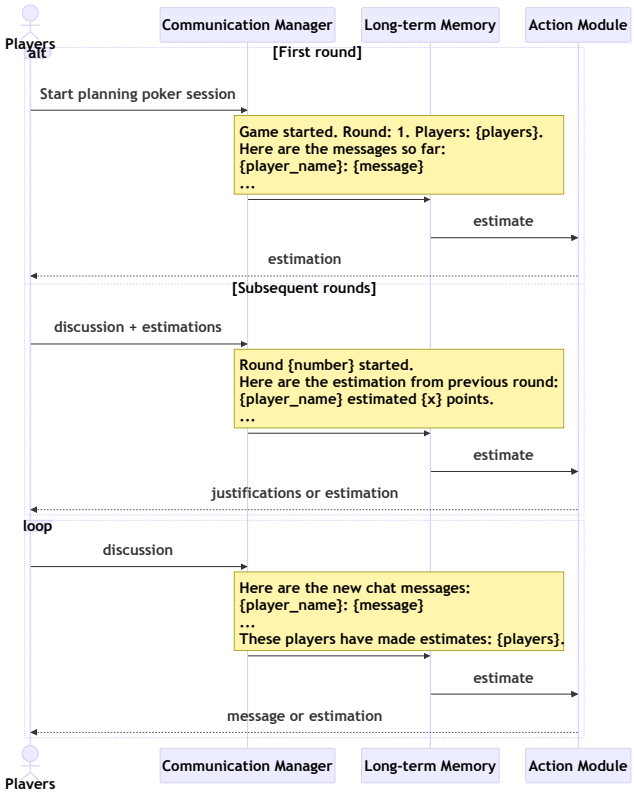

Fig. 2. Communication protocol of ***SEEAgent*** in an agile estimation session based on the Agile Planning Poker style

stakeholders, ensuring efficient communication with human practitioners and other agents. The action module defines two primitive interaction actions: (a) $chat$, allowing the agent to provide justifications, engage in collaborative dialogue, and articulate reasoning processes; (b) $make\_estimation$, allowing the agent to commit to an estimation. The input interfaces of these actions are concisely predefined to ensure the tool-use capability of LLMs can efficiently use them.

The Communication Manager serves as the main interface for managing external communication. The centralised message aggregation method implemented by this module handles communications from the environment, e.g. human practitioners and other ***SEEAgents***, and pre-processes them before passing them into the short-term memory. This module ensures operational coherence by systematically coordinating messages between external sources and the internal mechanisms of an agent. The role of this module is necessary as it provides a structured and meaningful stream of information that allows the agent to communicate effectively.

The communication protocol of ***SEEAgent***, illustrated in Figure 2, follows a structured communication flow between multiple internal components. The protocol differentiates between the first round and the subsequent rounds of interaction. On initiation of the first round, the Communication Manager aggregates players' information and message content into a structured prompt, whereas in subsequent rounds, the Communication Manager incorporates prior round estimations. This structured prompt is passed into long-term memory, enabling the agent to reason and determine its next action – whether to make an estimate or facilitate discussion. The agent then enters a loop of discussion with the players, and estimates are refined until the round concludes.

E. Prototype implementation

We have implemented our framework in a platform for human developers and ***SEEAgents*** to work together in estimating user stories for a given software project. The platform consists of a back-end server, which is responsible for orchestrating the agile Planning Poker estimation session mechanics and real-time updates. This component relies on the WebSocket protocol [21] to establish bidirectional communication with the Agent server and front-end. The protocol enables real-time broadcasting of participant actions across all connected clients. The Agent server functions as the centralised management system for ***SEEAgents***. This component is responsible for agent initiation and establishing WebSocket-based communication with the back-end server. Upon initialisation, each agent is configured with a specific role-play prompt. The system allows for flexible model selection to be used as the long-term memory, with the *gpt-4o-mini* as the default one.

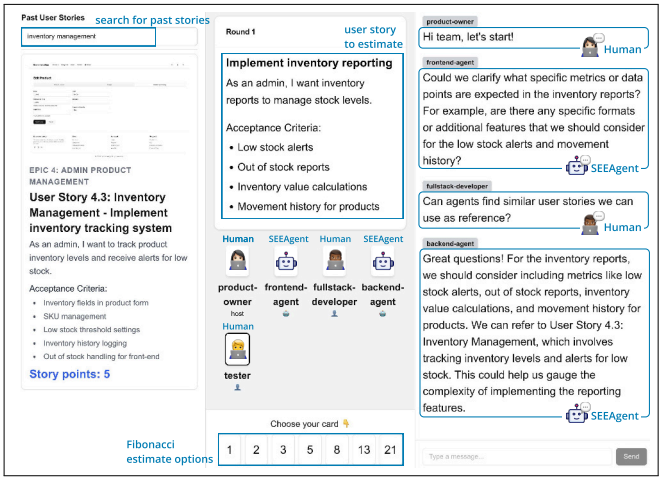

Fig. 3. Screenshot of the front-end for human players

The front-end is implemented as a web-based graphical user interface that facilitates human-AI interaction in a typical agile estimation based on the widely used Planning Poker method. Figure 3 illustrates the interface presented to the users. The middle column contains a card showing the active estimation round and the current user story being estimated. A roster where the active estimator is visually distinguished through a black border indication. Positioned at the bottom of the central panel, there is a Fibonacci sequence for point allocation. In the left panel, users are presented with past user stories of a software project developed for the human study. A text-based search box is included for quickly finding the stories

that the agents refer to. The right panel incorporates a real-time chat interface, facilitating message exchange between human participants and agents.

More details of our ***SEEAgent*** platform can be found at: *https://figshare.com/s/bb33c350859802b071ea*.

III. Evaluation

Our evaluation aims to demonstrate the predictive performance of a single ***SEEAgent*** against state-of-the-art techniques for agile effort estimation. We also examine how these agents synergise within human teams in a designed environment through a comprehensive human study. This dual approach allows us to assess both estimation accuracy and practical adoption factors of our framework. The research questions we aim to answer are:

- **RQ1:** *How accurate is **SEEAgent** in estimating user stories compared to state-of-the-art (SOTA) techniques?*
  Our agent architecture relies on long-term memory as the source to produce the estimations; hence, it is critical to validate whether it can churn out accurate effort estimates. Accurate estimates are vital for the agent's effectiveness in planning sessions as they build credibility in team discussions and enable the agent to meaningfully contribute to the conversation. To evaluate this integral component, we compare our approach against the most recent deep learning approaches, namely: Deep-SE [5], GPT2SP [8], and Fine-SE [9]. The same evaluation metrics (mean absolute error, mean magnitude of relative error, and performance indicator) and datasets in the SOTA approaches are used for a fair comparison. This question is essential to ensure our architectural choice of using LLMs for estimation is valid.
- **RQ2:** *How do human agile developers perceive **SEEAgent** in working with our **SEEAgents** in agile effort estimation?*
  For AI agents to be effective collaborators, they must be accepted and trusted by development teams. Therefore, understanding how human practitioners perceive and interact with our agents within an effort estimation environment is vital – it reveals the benefits, challenges, and improvement areas where the agents might possess. To answer this question, we conduct a human study with software development professionals, letting them interact with agents in a designed environment and then collecting their feedback through a mixed-format survey. This human-centred evaluation provides us with insights into the practical viability of our approach and guides improvements for real-world adoption.

A. *RQ1: Effectiveness of SEEAgent in estimating user stories*

1) Dataset: We used a publicly available dataset from the Fine-SE paper [9]. In the paper, the authors trained and evaluated their approach on a dataset with 17 industrial projects and four open-source software projects. To allow direct comparison with their approach and ensure reproducibility, we utilised their publicly available dataset of four open-source projects: Mesos (ME), Data Management (DM), Talend Data Quality (TD), and Usergrid (US). The projects come from a variety of software domains - ME focuses on distributed systems infrastructure, DM on data processing workflows, TD on data quality tools, and US on backend-as-a-service APIs. The data was mined from the Jira issue tracking system, where these projects are managed. The dataset consists of Jira issues that represent bug reports, development tasks, and user stories. Those issues were assigned with positive, non-zero story points that are appropriately sized (e.g., any estimations greater than 100 were filtered out). The statistical descriptions of the project are presented in Table I.

TABLE I
Statistical description of the open-source projects

Project	No. stories	Story points				
		Min	Max	Mean	Median	Std
DM	4,320	0	195	7.91	3	16
ME	414	1	13	3.93	3	2.6
TD	73	0.5	40	8.25	8	7.4
US	100	1	8	3.31	3	1.2
Total	4,907	-	-	-	-	-

2) Baselines: In part of the study, we experiment with two variants of ***SEEAgents***: one utilised the base *Llama-3.1-8B-instruct* model, and another implemented a fine-tuned version of the same model for the SEE task. LLMs have demonstrated significant abilities to understand the semantics in natural language, making them potential candidates for tasks involving the understanding of textual information in software tasks. While the base model ***SEEAgent*** leverages the general knowledge encoded in its parameters, the fine-tuned variant combines this foundational knowledge with project-specific insights gained through additional training on SEE-specific data.

In addition, we compare the performance of our approach against state-of-the-art techniques in agile effort estimation, including Fine-SE [9], GPT2SP [8], and Deep-SE [5]. Deep-SE established the initial adoption of deep learning for SEE when it introduced a hybrid architecture combining Long Short-term Memory (LSTM) networks with Recurrent Highway Networks (RHN). GPT2SP extended the GPT-2 architecture to generate point estimates. Most recently, Fine-SE employed BERT [22] to extract semantics features from user stories, integrating these with neural networks to produce effort estimates. We have replicated the evaluation of those approaches based on the code they provided.

3) Implementation configurations: In this experiment, we focus solely on evaluating the long-term memory component of ***SEEAgent***, comparing the accuracy of two variants: one using the base *Llama-3.1-8B-instruct* model, and another using its fine-tuned version for the SEE task. We evaluate ***SEEAgent*** in this standalone configuration to ensure a fair comparison, since ***SEEAgent***'s full architecture offers additional capabilities through multi-agent communication and interactive refinement. We deliberately isolate the core estimation com-

ponent to maintain equivalent testing conditions with previous approaches. This controlled setup enables direct performance comparison on the same metrics and datasets used.

For each project in our dataset, we performed separate fine-tuning of the base LLM model. We utilised the Unsloth library [23] for this process as it supports the QLoRA fine-tuning technique. To ensure fair comparison with results from the Fine-SE paper, we maintained the same train-test split ratios and data distribution as used in their published implementation. The fine-tuning process was consistent across all projects: we configured the loading of the model into memory with the following parameters: *max_sequence_length* is set to 2,048 tokens; the LoRA rank was set to 8 with an equal scaling factor to maintain a balance between the model capacity and new knowledge introduction. For training, we configured the epoch to 10 with a learning rate of *2e-4*. The training processes were conducted using Google Colab's A100 GPU infrastructure.

Training data for our approach includes the issue summary and the ground-truth story points. We do not include other information as used by the Fine-SE approach, such as the number of stories created by the creator, the number of stories assigned to the developer, and the number of stories tested by the tester. Data from each issue was processed and input into the fine-tuning prompt presented in Listing 1 and fed into the LLMs for training and evaluation.

4) Evaluation metrics: We use the same evaluation metrics, i.e. mean absolute error (MAE), mean magnitude of relative error (MMRE), and performance indicator (PRED), as in a prior work [9]. MAE measures the average of absolute differences between predicted and ground truth estimation. It intuitively captures the proposed model's average error margin across the whole test set.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

where y_i and $\hat{y}_i$ are the ground truth estimation and the predicted estimation, respectively. n is the number of instances in the test set.

MMRE computes the average magnitude of the relative error between the predicted and ground-truth estimations. It presents the relative normalised magnitude of error made by the model on the given projects, which is ideal for comparison across multiple projects of different estimation unit systems.

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{2}$$

PRED(50) examines the percentage of predictions that fall within 50% of ground truth estimation values. It offers a practical insight by showing how often the model achieves a certain level of accuracy, highlighting the model's ability to make estimations within a threshold of acceptable error margin (50%).

$$\text{PRED}(50) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1, & \text{if } \frac{|y_i-\hat{y}_i|}{y_i} \leq 0.50\\ 0, & \text{otherwise}\end{cases} \tag{3}$$

TABLE II
BENCHMARK RESULTS OF ***SEEAgent*** VS BASELINES

Proj	Approach	MAE	MMRE	PRED(50)
DM	Deep-SE	4.786	3.118	0.315
	GPT2SP	4.303	1.900	0.362
	Fine-SE	4.628	1.739	0.264
	SEEAgent $_{\text{base model}}$	17.053	11.698	0.157
	SEEAgent $_{\text{fine-tuned}}$	**4.251**	**1.491**	**0.383**
ME	Deep-SE	1.382	0.489	0.687
	GPT2SP	1.583	0.629	0.578
	Fine-SE	1.395	0.449	0.723
	SEEAgent $_{\text{base model}}$	7.398	2.463	0.108
	SEEAgent $_{\text{fine-tuned}}$	**1.217**	**0.414**	**0.747**
TD	Deep-SE	**4.215**	1.121	**0.533**
	GPT2SP	5.736	2.231	0.400
	Fine-SE	4.659	**0.755**	0.400
	SEEAgent $_{\text{base model}}$	7.133	1.382	0.467
	SEEAgent $_{\text{fine-tuned}}$	5.133	1.456	0.333
US	Deep-SE	0.618	0.163	0.850
	GPT2SP	0.447	0.120	0.950
	Fine-SE	1.826	0.541	0.150
	SEEAgent $_{\text{base model}}$	11.300	2.797	0.050
	SEEAgent $_{\text{fine-tuned}}$	**0.350**	**0.085**	**1.000**

[1] The lower of MSE and MMRE, the better.
[2] For PRED(50), a higher value is better.
[3] The best results on evaluation measures are highlighted in bold.

5) Results: The benchmark results (see Table II) between ***SEEAgent***$_{\text{base model}}$ and ***SEEAgent***$_{\text{fine-tuned}}$ reveal remarkable performance differences across all projects. For the DM project, the base model shows considerable deviation with an MAE of 17.053 and MMRE of 11.698, whereas the fine-tuned version substantially reduces these errors to 4.251 and 1.491, respectively. In the ME project, the fine-tuned variant improved MAE by 83.55% (from 7.398 to 1.217) and MMRE by 83.18% (from 2.463 to 0.414). The most improvement is observed in the US project, where fine-tuning reduces MAE from 11.3 to 0.35 and MMRE from 2.797 to 0.085. In terms of the PRED(50) metric, the fine-tuned model achieves notably higher scores across all projects, with 591% improvement in ME (from 0.108 to 0.747) and a 20-fold increase for US (from 0.05 to 1). These consistent improvements across the projects strongly suggest that the fine-tuning process successfully enabled the model to learn project-specific effort estimation patterns.

When comparing against the current state-of-the-art (SOTA) approaches, ***SEEAgent***$_{\text{fine-tuned}}$ outperforms in three out of four projects. In the DM project, our fine-tuned agent achieves up to 16.63% improvement over the second-best results across the evaluation metrics, notably reducing the MMRE from 1.739 down to 1.491. Regarding the ME project, our approach yields superior results across all metrics (MAE: 1.217, MMRE: 0.414, PRED(50): 0.747), outperforming Deep-SE (1.382, 0.489, 0.687), GPT2SP (1.583, 0.629, 0.578), and Fine-SE (1.395, 0.449, 0.723). For the US project, our fine-tuned agent achieves notable performance with the lowest MAE (0.350) and MMRE (0.085) among all approaches while having the perfect PRED(50) score. As for the TD project, our approach (MAE: 5.133, MMRE: 1.456, PRED(50): 0.333) outperforms GPT2SP across all metrics, but still leaves room for improvement compared to Deep-SE (MAE: 4.108, PRED(50): 0.438)

and Fine-SE (MMRE: 0.692). This is likely due to insufficient data for effective fine-tuning of LLMs (only 72 samples) and the high story point variance (std=7.4, indicating diverse issue types with varying complexity).

TABLE III
COMPARISON BETWEEN APPROACHES USING WILCOXON TEST AND $\hat{A}_{12}$ EFFECT SIZE (IN BRACKETS)

SEEAgent vs	Deep-SE	GPT2SP	Fine-SE
DM	<0.001 [0.59]	0.061 [0.53]	0.007 [0.55]
ME	0.028 [0.65]	0.042 [0.63]	0.032 [0.67]
TD	0.095 [0.27]	0.359 [0.73]	0.524 [0.40]
US	<0.001 [0.95]	0.002 [0.90]	<0.001 [0.95]

Table III presents the results of the Wilcoxon test (along with Vargha-Delaney $\hat{A}_{12}$ effect size) to measure the statistical significance of $\textbf{\textit{SEEAgent}}_{\text{fine-tuned}}$ against the SOTA approaches. Using the alpha level α of 0,05, our approach displays the strongest results in the US project, demonstrating statistical significance ($p < 0.001$) and a large effect size (0.9 to 0.95) compared to all other approaches. Performance in DM and ME projects shows moderate improvements with small to medium effect sizes (0.53 to 0.67), while results in the TD project indicate no significant improvements.

B. *RQ2: Human-AI collaboration*

1) Experimental design and procedures: We conducted a human user study using a mixed-format survey approach that combines interactive prototype evaluation with subsequent data collection through a survey incorporating both structured (quantitative) and open-ended (qualitative) response formats. The study was approved by our university's Human Research Ethics Committee (approval number 2025/015).

There are 12 participants recruited through professional networks within academic and industry settings, utilising a purposive sampling approach to ensure participants possessed relevant software development experience. To maintain ethical standards and minimise potential bias, all prospective participants were explicitly informed that their participation was voluntary and their decision to participate or decline would not affect their professional or personal relationships with the researchers.

The participants are asked to join a session in which they work collaboratively with some other human participants and ***SEEAgents*** to estimate user stories for a given software project. We chose a small, simplified e-commerce software project (namely *eShope*) which has functionalities (e.g. log in, log out, shopping cart, order, etc.) that are familiar to all the participants. The *eShope* project consists of 33 user stories written based on the Connextra format [24] and containing several acceptance criteria. Details of this project can be found at *https://figshare.com/s/592bb84543653d80a515*.

Each estimation session involves six participants: three human developers, two ***SEEAgents*** with different development roles, and a human product owner (the first author) who presents the user stories and guides the session. During each session, participants will estimate various user stories of the *eShope* application. The estimation session is designed to last approximately 30 minutes. Participants are then asked to complete a survey that consists of 12 Likert-scale questions and two open-ended questions.

The estimation session is designed based on the agile Planning Poker method (also known as Scrum Poker), a collaborative estimation technique widely used by agile teams in the industry [3]. The process begins with the product owner introducing a user story that requires estimation. The participants (*including both **SEEAgents** and human developers*) then discuss the details of the story, ensuring everyone understands the requirements and scope. Each participant will then provide their estimate in story points. Next, the estimates are compared. If all estimates are the same, indicating consensus, the team moves to the next story. However, if there is a difference in estimates, the participants provide justifications for their estimates, and a group discussion follows to address any misunderstandings or differing perspectives. After this discussion, the participants re-estimate the story. In our experiments, we set this cycle of discussion and re-estimation to be up to two rounds[1]. If consensus is not reached after this limit, the human product owner will make an ultimate decision based on the majority estimation. This process ensures that the team reaches a shared understanding of the story's requirements and agrees on the estimated effort.

2) Implementation configurations: Since the number of user stories of *eShope* is limited, instead of fine-tuning the long-term memory of ***SEEAgent***, we opted for the option of using in-context learning [13]. This method incorporates a small number of ground-truth samples directly into the short-term memory. This approach enables us to leverage the model's existing capabilities while contextually adapting to the project-specific knowledge without extensive training data. We used the *gpt-4o-mini* model from OpenAI as the long-term memory for ***SEEAgent*** as it provides a good balance between response time and reasoning ability.

We implemented two variants of ***SEEAgent*** with role-play prompting following research demonstrating that LLMs can effectively embody specialised technical roles. [25]. Each variant was initialised with either front-end development or back-end development expertise. This is achieved by establishing a role in agents' short-term memory when initiating them: *You are a {role} developer agent*. This methodology is backed by empirical evidence [25] suggesting that role-play prompting helps models maintain consistent domain expertise and generates more effective domain-specific reasoning compared to without the prompt. The role-specific agents exhibited different technical perspectives aligned with their assigned development domains, enabling the study participants to gather insights from complementary areas of software engineering expertise.

3) Post-session survey: We designed a mixed-format survey (see Table IV) to gain a holistic view of human practitioners' perception and acceptance toward ***SEEAgents***. The survey has

[1]This limit can be adjusted based on a specific context.

a structured questionnaire that assesses multiple aspects of ***SEEAgent***, including its ability to facilitate consensus, bridge knowledge gaps, identify risks, ease of use, and quality of collaboration with human developers. The survey also evaluated practitioners' sentiment towards the usage of ***SEEAgents*** in SEE, including their trust in the agents' estimations, willingness to rely on and recommend such systems. This survey helps evaluate both the practical utility and adoption potential of ***SEEAgents*** in SEE contexts. The quantitative analysis was complemented by qualitative data from open questions, which helps explore perceived benefits, challenges, and potential improvements to better agent design in software development environments.

TABLE IV
HUMAN STUDY QUESTIONNAIRE

ID	Question
Perceptions of *SEEAgents*	
Q1‡	The AI agent's estimations helped us reach consensus efficiently
Q2‡	I think AI agents can bridge knowledge gaps between team members
Q3‡	I think the AI agent brought valuable insights into potential risks
Q4‡	Learning to work with the AI agent was easy for me
Q5‡	I think collaborating with the AI agent felt natural
Q6‡	I find the AI agents' responses were clear and understandable
Q7‡	My opinion of AI in SEE has become more positive
Q8‡	I believe AI in SEE could benefit teams
Q9‡	I think the AI agent can be a substitute for an expert
Q10‡	I would be comfortable relying on AI agent's estimations
Q11‡	I am more likely to recommend using AI agents
Q12‡	I would trust the estimations provided by the AI agent
Open-ended Questions	
Q13*	Based on your experience, what are your thoughts on the potential benefits and challenges of using AI agents in software effort estimation?
Q14*	What suggestions do you have for improving the design or functionality of the AI agents to enhance their usefulness and effectiveness in estimation sessions?

‡ Likert scale: ○ Strongly disagree, ○ Somewhat disagree, ○ Neutral, ○ Somewhat agree, ○ Strongly agree
* Open-ended questions.
** The term "AI" and "AI agent" refer to ***SEEAgent***.

4) Results: Figure 4 presents the participants' responses to Questions 1–12 in our survey after they completed an agile estimation session with our ***SEEAgents***. We categorise the findings into three key dimensions: operational effectiveness, user experience, and user trust. These dimensions help understand the impact and potential of ***SEEAgents*** in collaborating with human developers in agile effort estimation.

In terms of operational effectiveness, more than half (66.7%) of the respondents agreed that the agents' estimations helped reach consensus more efficiently (Q1). Similarly, 58.3% believe that the agents bridged knowledge gaps between the team members (Q2). Comparably, 58.3% of the participants recognised the agents' capability to shed insights into potential risks (Q3). These results suggest that our ***SEEAgents*** address some inherent challenges in agile effort estimation, particularly in terms of collective decision-making and knowledge gap management.

In addition, the results demonstrate a positive user experience with our ***SEEAgents***, with half of the participants finding it easy learning to work with the agents (Q4). It is also important to note that a significant proportion (83.3%) of the participants felt natural when working with the agents' natural (Q5) and found that the agents' responses were clear and understandable (Q6). The positive experience with our ***SEEAgents*** has helped form a positive view towards using AI in agile effort estimation (83.3% of agreement for Q7). These results suggest that the ***SEEAgents*** have successfully achieved a high level of intuitiveness that could greatly facilitate their adoption in practice. The positive user experience also implies that a well-designed AI agent system can effectively mitigate professional scepticism and hesitancy toward AI adoption.

In terms of user trust, all of the participants believed that ***SEEAgent*** would benefit agile teams in effort estimation (Q8). While only one third of them thought that the agent can fully substitute an expert (Q9), 75% of the participants did not object to relying on the agent's estimations (Q10). The result also shows that more than half of the participants trusted ***SEEAgent***'s estimations (Q12) and would recommend using it in practice (Q11). These results suggested that, while their utility is universally acknowledged, there remains a preference for human oversight in essential decision-making processes. This indicates that successful adoption strategies should place emphasis on humans as decision makers and AI agents as supportive collaborators.

Our analysis of the responses for the open-ended questions (Q13 and Q14) also offers further insights and confirms the positive experience of the participants in collaborating with our ***SEEAgents***. They found that the agents "*give us rational feedback and opinions*" and "*benefit development teams to reach consensus*". Many participants found the agents are useful in the agile effort estimation session since they "*bridge knowledge gap between team members (frontend vs backend dev, new vs old team members)*" and promote communication between team members. The participants were also impressed with how our ***SEEAgents*** used past user stories to estimate new user stories since "*it can help the team not only estimate the user stories correctly but also see the related issues arising when implementing these sorts of features in the past*".

The participants also suggested several improvements to ***SEEAgents***, such as leveraging "*not only the user stories but also the source code and infrastructure*". They also suggested integrating ***SEEAgent*** with existing agile tools such as Jira, Trello and supporting voice communication (instead of only chat messaging) with human developers. We will develop these new features into ***SEEAgents*** as part of our future work.

C. Threats to validity

Our selection of the foundational LLM of our ***SEEAgent*** poses an internal threat. Different LLMs may exhibit different reasoning capabilities, which might affect the agent's behaviour in estimating user stories and collaborating with human agile team members to reach a consensus estimate. We have mitigated the threat by designing ***SEEAgent***'s ar-

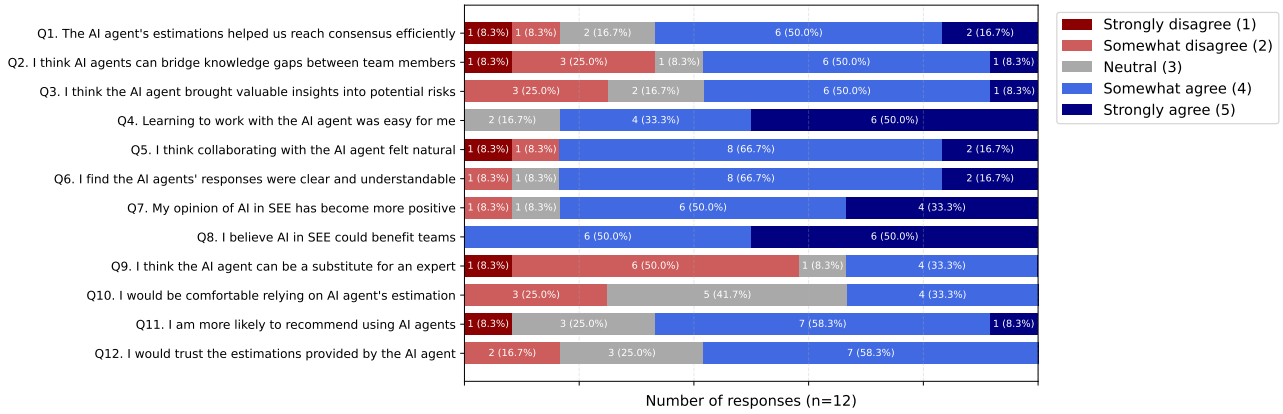

Fig. 4. Survey results

chitecture such that it does not depend on any specific LLM and experimented with two widely used foundational LLMs (Meta's Llama and OpenAI's GPT models). While the underlying LLM provides the foundation for natural language understanding and generation, LLM alone does not give the benefits of agentic behaviours (i.e., adaptability, interactivity, and consensus-building). The performance improvements we observe stem from our architecture that transforms stateless LLMs into collaborative agents capable of participating in team-based estimation processes. Future work would involve integrating other LLMs, investigating their impact on the performance of our agents in the agile estimation context, and conducting comprehensive ablation studies to validate our modular architecture. In addition, our replication of the SOTA baselines strictly followed the description provided in their work; however, we acknowledge that the environment we set up might be different from their original environment. To mitigate this threat, we used the same dataset and maintained consistent experimental settings across all comparisons to ensure fair evaluation.

A key external threat to our human study resides in the limited sample size of 12 participants using a simplified project. This relatively small number of participants may not be fully representative of the broader software development community, limiting the generalisation of our study to larger, high-pressure industrial settings. However, we note that recruiting participants to join live collaborative estimation sessions is significantly challenging since it takes significant time and effort. This sample size of human study is also common in the software engineering literature. The result has shown the effectiveness of our ***SEEAgents*** in collaborating with human developers in agile effort estimation. Insight from this study will help us attract the community's interest and extend it to a larger number of participants.

IV. Related Work

Agile effort estimation has been attracting significant interest in recent years. A range of machine learning-based techniques has been proposed to estimate user stories using the knowledge learnt from past user stories in the same project. One common approach is processing the description of a user story into a set of learning features (e.g. TF-IDF features as in [6]). Other approaches also explore the use of developer-related features (e.g. [7]) to improve the estimation accuracy, or employ clustering techniques to group similar issues and use this as the basis for estimation [26].

Deep learning approaches are also widely explored to tackle the problem since the seminal work of Choetkiertikul *et al.* [5], which leverages Long Short-Term Memory (LSTM) networks and Recurrent Highway Networks (RHN) to generate semantic features from user story descriptions and use them for estimation. In 2022, Ahmad and Ibrahim used only LSTM [27] for SEE and yielded better accuracy over traditional approaches. Fu and Tantithamthavorn [8] introduced GPT2SP, a transformer-based architecture that employs pre-trained language models and results in superior accuracy over previous methods. In 2023, Kassem *et al.* [28] developed a Deep Attention Neural Network that uses the attention mechanism to capture relative word importance in issue descriptions. A recent approach called Fine-SE [9] explored the idea of combining expert features with semantic features to produce better estimations. While these approaches have shown promising results in terms of accuracy, they operate as black-box models that do not explain their estimation nor participate in team discussion, an important aspect of agile effort estimation. Our ***SEEAgent*** framework leverages LLMs to create agents that not only provide estimates but also provide relevant justifications. Furthermore, they can communicate with human practitioners and provide different perspectives on a user story, similar to how cross-functional agile teams operate in practice.

Recent research in LLM-based agents has demonstrated their capability to autonomously solve complex and human-like tasks [29], [30]. Inspired by those breakthroughs, a range of approaches have been recently proposed to develop LLM-based agents for performing software engineering tasks such as requirement elicitation (e.g. [31]), code generation (e.g.

[32], [33]), quality assurance (e.g. [34], [35]) and software development (e.g. [36], [37]). Tao et al. [36] used LLM agents to simulate planning discussions for software tasks before the commencement of coding activities. Recent studies [38], [39] have demonstrated promising results of multi-agent systems in emulating agile software development workflows, where agents collectively simulate the roles and interactions inherent in agile methodologies. While prior research has shown the effectiveness of LLM-based multi-agent systems in various project management contexts, there remains a gap regarding their application in agile effort estimation. Our work distinguishes itself by studying not only the quantitative accuracy of estimations, but also understanding the human-AI interaction dynamics and practitioners' perceptions of AI agents in the context of a collaborative effort estimation scenario.

V. Conclusions and future work

In the paper, we have proposed ***SEEAgent***, an LLM-based multi-agent framework which is capable of not only estimating user stories accurately but also coordinating, communicating and discussing with human developers to reach an agreed-upon estimate. Our approach represents a significant departure from the prior state-of-the-art machine learning-based techniques in agile estimation, which *cannot* even explain or justify their estimates. Thus, our work has filled this very important gap. Beyond leveraging a powerful LLM, our key contributions lie in the design of a multi-agent framework that integrates specialised roles, structured memory mechanisms, and interactive justification protocols to collaboratively generate effort estimates. To the best of our knowledge, our work is the first dynamic, consensus-driven estimators that incorporate human-agent interaction and address challenges specific to agile estimation practice.

Results from a rigorous human study have shown that our ***SEEAgents*** can engage in meaningful discussions with human practitioners, make accurate estimations on user stories and provide useful justification. The participants found that our agents help them reach consensus more efficiently, bridge the knowledge gap, and bring valuable insights into potential risks. The participants also agreed that they would trust our ***SEEAgent*** and recommend using it in practice.

Feedback from the participants also provides several avenues for our future work, such as enriching ***SEEAgent***'s knowledge with source code and infrastructure data, or integrating it with existing agile project management tools such as JIRA. We will also incorporate voice interaction capability into ***SEEAgent*** to facilitate more natural communication with humans. Our future work also involves expanding the human study significantly into the commercial domain to bring our ***SEEAgent*** to the wider agile software development communities. This extension would entail increasing the number of participants and the scale of the estimated project backlog under evaluation. We plan to conduct future studies with a larger group of software developers with diverse expertise and domains, to improve the statistical robustness and cross-domain generalisation. We aim to experiment with ***SEEAgent*** on industrial-scale backlogs containing hundreds of heterogeneous stories with complex interdependencies and evolving requirements. Additional multi-agent interaction metrics (such as the number of rounds to consensus, convergence failure rate, and agent response time) could be incorporated to obtain deeper insights into human-agent interaction effectiveness.

References

[1] R. J. Leach, "Software cost estimation," in *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, Ltd. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W6939

[2] M. Pozenel, L. Fürst, D. Vavpoti, and T. Hovelja, "Agile effort estimation: Comparing the accuracy and efficiency of planning poker, bucket system, and affinity estimation methods," vol. abs/2401.16152. [Online]. Available: https://api.semanticscholar.org/CorpusID:264598156

[3] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.

[4] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, ser. PROMISE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 82–91. [Online]. Available: https://doi.org/10.1145/2639490.2639503

[5] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," vol. 45, no. 7, pp. 637–656, conference Name: IEEE Transactions on Software Engineering. [Online]. Available: https://ieeexplore.ieee.org/document/8255666/?arnumber=8255666

[6] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/2972958.2972959

[7] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proceedings of the 2018 International Conference on Software and System Process*. ACM, pp. 106–110. [Online]. Available: https://dl.acm.org/doi/10.1145/3202710.3203160

[8] M. Fu and C. Tantithamthavorn, "GPT2sp: A transformer-based agile story point estimation approach," vol. 49, no. 2, pp. 611–625, conference Name: IEEE Transactions on Software Engineering. [Online]. Available: https://ieeexplore.ieee.org/document/9732669/?arnumber=9732669

[9] Y. Li, Z. Ren, Z. Wang, L. Yang, L. Dong, C. Zhong, and H. Zhang, "Fine-SE: Integrating Semantic Features and Expert Features for Software Effort Estimation," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. ACM, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3597503.3623349

[10] A. Mohammad and B. Chirchir, "Challenges of Integrating Artificial Intelligence in Software Project Planning: A Systematic Literature Review," vol. 4, no. 3, pp. 555–571. [Online]. Available: https://www.mdpi.com/2673-6470/4/3/28

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models." [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[12] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths. Cognitive Architectures for Language Agents. [Online]. Available: http://arxiv.org/abs/2309.02427

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., pp. 1877–1901.

[14] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and

Y. Liu, Eds. Association for Computational Linguistics, pp. 7654–7673. [Online]. Available: https://aclanthology.org/2020.emnlp-main.617/

[15] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-Efficient Transfer Learning for NLP. [Online]. Available: http://arxiv.org/abs/1902.00751

[16] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a Unified View of Parameter-Efficient Transfer Learning. [Online]. Available: http://arxiv.org/abs/2110.04366

[17] X. L. Li and P. Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. [Online]. Available: http://arxiv.org/abs/2101.00190

[18] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized llms," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., pp. 10 088–10 115.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. [Online]. Available: http://arxiv.org/abs/2203.02155

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[21] A. Melnikov and I. Fette, "The WebSocket Protocol." [Online]. Available: https://datatracker.ietf.org/doc/rfc6455

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [Online]. Available: http://arxiv.org/abs/1810.04805

[23] M. H. Daniel Han and U. team, "Unsloth," 2023. [Online]. Available: http://github.com/unslothai/unsloth

[24] J. M. E. v. d. W. Garm Lucassen, Fabiano Dalpiaz and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *Requirements Engineering: Foundation for Software Quality*, M. Daneva and O. Pastor, Eds. Springer International Publishing, pp. 205–222.

[25] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang, and X. Dong, "Better Zero-Shot Reasoning with Role-Play Prompting," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. Association for Computational Linguistics, pp. 4099–4113. [Online]. Available: https://aclanthology.org/2024.naacl-long.228/

[26] V. Tawosi, A. Al-Subaihin, and F. Sarro, "Investigating the effectiveness of clustering for story point estimation," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 827–838, ISSN: 1534-5351. [Online]. Available: https://ieeexplore.ieee.org/document/9825830/?arnumber=9825830

[27] F. B. Ahmad and L. M. Ibrahim, "Software development effort estimation techniques using long short term memory," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, pp. 182–187. [Online]. Available: https://ieeexplore.ieee.org/document/9759751/?arnumber=9759751

[28] H. Kassem, K. Mahar, and A. A. Saad, "Story point estimation using issue reports with deep attention neural network," vol. 17, no. 1, p. 230104. [Online]. Available: https://www.e-informatyka.pl/index.php/einformatica/volumes/volume-2023/issue-1/article-4/

[29] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. E. Zhu, L. Jiang, X. Zhang, S. Zhang, A. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation." [Online]. Available: https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/

[30] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society," vol. 36, pp. 51 991–52 008.

[31] M. Ataei, H. Cheong, D. Grandi, Y. Wang, N. Morris, and A. Tessier. Elicitron: An LLM Agent-Based Simulation Framework for Design Requirements Elicitation. [Online]. Available: http://arxiv.org/abs/2404.16045

[32] J. Li, H. Le, Y. Zhou, C. Xiong, S. Savarese, and D. Sahoo. CodeTree: Agent-guided Tree Search for Code Generation with Large Language Models. [Online]. Available: http://arxiv.org/abs/2411.04329

[33] H. Nhat Phan, T. N. Nguyen, P. X. Nguyen, and N. D. Q. Bui, "HyperAgent: Generalist Software Engineering Agents to Solve Coding Tasks at Scale," p. arXiv:2409.16299. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2024arXiv240916299N/abstract

[34] G. Deng, Y. Liu, V. Mayoral-Vilches, P. Liu, Y. Li, Y. Xu, T. Zhang, Y. Liu, M. Pinzger, and S. Rass, "PENTESTGPT: Evaluating and harnessing large language models for automated penetration testing," in *Proceedings of the 33rd USENIX Conference on Security Symposium*, ser. SEC '24. USENIX Association, pp. 847–864.

[35] J. Yoon, R. Feldt, and S. Yoo, "Intent-Driven Mobile GUI Testing with Autonomous Large Language Model Agents," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pp. 129–139. [Online]. Available: https://ieeexplore.ieee.org/document/10638557

[36] W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng, "MAGIS: LLM-Based Multi-Agent Framework for GitHub Issue Resolution," vol. 37, pp. 51 963–51 993.

[37] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework." [Online]. Available: https://openreview.net/forum?id=VtmBAGCN7o

[38] S. Zhang, Z. Xing, R. Guo, F. Xu, L. Chen, Z. Zhang, X. Zhang, Z. Feng, and Z. Zhuang, "Empowering Agile-Based Generative Software Development through Human-AI Teamwork." [Online]. Available: https://dl.acm.org/doi/10.1145/3702987

[39] M. H. Nguyen, T. P. Chau, P. X. Nguyen, and N. D. Q. Bui. AgileCoder: Dynamic Collaborative Agents for Software Development based on Agile Methodology. [Online]. Available: http://arxiv.org/abs/2406.11912