



SimAC: simulating agile collaboration to generate acceptance criteria in user story elaboration

Yishu Li¹ · Jacky Keung¹ · Zhen Yang² · Xiaoxue Ma¹ · Jingyu Zhang¹ · Shuo Liu¹

Received: 3 November 2023 / Accepted: 16 May 2024 / Published online: 21 June 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract

In agile requirements engineering, Generating Acceptance Criteria (GAC) to elaborate user stories plays a pivotal role in the sprint planning phase, which provides a reference for delivering functional solutions. GAC requires extensive collaboration and human involvement. However, the lack of labeled datasets tailored for User Story attached with Acceptance Criteria (US-AC) poses significant challenges for supervised learning techniques attempting to automate this process. Recent advancements in Large Language Models (LLMs) have showcased their remarkable text-generation capabilities, bypassing the need for supervised fine-tuning. Consequently, LLMs offer the potential to overcome the above challenge. Motivated by this, we propose SimAC, a framework leveraging LLMs to simulate agile collaboration, with three distinct role groups: requirement analyst, quality analyst, and others. Initiated by role-based prompts, LLMs act in these roles sequentially, following a create-update-update paradigm in GAC. Owing to the unavailability of ground truths, we invited practitioners to build a gold standard serving as a benchmark to evaluate the completeness and validity of auto-generated US-AC against human-crafted ones. Additionally, we invited eight experienced agile practitioners to evaluate the quality of US-AC using the INVEST framework. The results demonstrate consistent improvements across all tested LLMs, including the LLaMA and GPT-3.5 series. Notably, SimAC significantly enhances the ability of gpt-3.5-turbo in GAC, achieving improvements of 29.48% in completeness and 15.56% in validity, along with the highest INVEST satisfaction score of 3.21/4. Furthermore, this study also provides case studies to illustrate SimAC's effectiveness and limitations, shedding light on the potential of LLMs in automated agile requirements engineering.

Keywords Large language models · Prompts engineering · User story · Acceptance criteria · Agile requirements engineering

1 Introduction

Over the past decades, the agile software development method has experienced widespread adoption and usage by both small and large established companies (Pantiuchina et al. 2017; Hoda et al. 2018). Wang et al. (2014) highlighted the significance of Requirements Engineering (RE) within the context of agile practice. Their findings indicated that over 90% of the interviewed agile practitioners regarded RE as an important aspect of their work. Notably, elaborating user stories emerged as the prevailing method for requirements elicitation within agile projects (Wang et al. 2014).

A user story is a short-loose text describing the needs of customers, while acceptance criteria usually appear as an extension of the user story. Acceptance criteria define what is “done” for the given user story (Sedano et al. 2019), enhancing the comprehension of practitioners on the requirements and facilitating the development process. The BABOK Guide,¹ one of the most popular industry standards of presenting RE techniques (Meredith et al. 2019), emphasizes the critical nature of user story elaboration, as it constitutes a fundamental component of the agile requirement analysis process. During elaboration, the detailed design and acceptance criteria for the user story are defined in order to deliver a working solution. However, this process involves iterative collaboration among agile team members to refine and finalize the details in each sprint. Consequently, Generating Acceptance Criteria (GAC) during the process of user story elaboration is labor-intensive but essential in agile RE.

Agile methodologies aim to enhance product development efficiency (Sverrisdottir et al. 2014). Automating labor-intensive tasks in agile development provides a practical means of streamlining processes, thereby enhancing efficiency. GAC demands extensive human involvement to ensure a comprehensive understanding of project requirements. Although automating this task may inadvertently result in team members lacking a full grasp of requirements, the generation of high-quality acceptance criteria at the initial stage can establish a solid foundation for further refinement, thereby facilitating more efficient collaboration. Building upon this understanding, our study aims to explore an automation paradigm for GAC to reduce the initial human effort involved in the process.

Few studies have dedicated themselves to automating GAC due to the absence of labeled datasets in User Stories associated with Acceptance Criteria (US-AC). That is to say, task-specific learning-based techniques have difficulty in automating this process. Recently, Large Language Models (LLMs), pre-trained by a huge amount of human-written text/code, have drawn immense interest in the Software Engineering (SE) community owing to their powerful generality even without supervised fine-tuning (Xia et al. 2023; Li et al. 2023; Kasneci et al. 2023). For example, Xia et al. (2023) revealed that directly applying state-of-the-art LLMs can substantially outperform all existing automated program-repairing techniques since traditional learning-based techniques are limited to the reliance on bug-fixing datasets to craft fix

¹ <https://www.iiba.org/career-resources/a-business-analysis-professionals-foundation-for-success/agile-extension/>

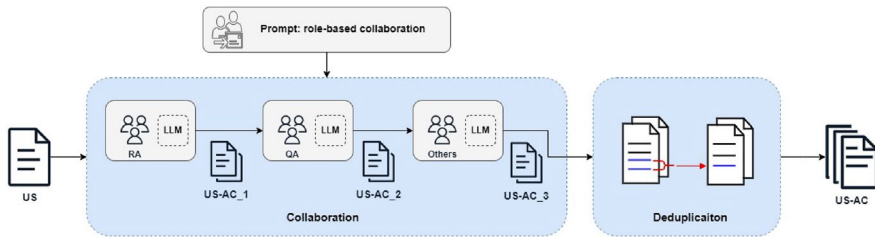


Fig. 1 Brief workflow of SimAC

templates. Li et al. (2023) proposed to approach role-playing and inception prompting to guide the communicative LLMs toward the programming task autonomously. In summary, LLM-based approaches circumventing supervised fine-tuning bring a chance to transition the labor-intensive SE tasks toward an automated paradigm.

Motivated by the powerful generality of LLMs, we propose SimAC, a framework that **Simulates** agile collaboration to generate **Acceptance Criteria** in user story elaboration by employing LLMs. Figure 1 presents the brief of SimAC, which encompasses two phases: collaboration and deduplication. In the collaboration phase, an LLM is initiated with prompts that provide instructions and illustrative examples to demonstrate how collaborative tasks interact. Following a prescribed procedure of create-update-update, the LLM acts collaboratively in three groups of roles, namely the Requirement Analyst (RA), Quality Analyst (QA), and Other team members (Others) in sequence. The requirement analyst first creates an initial suite of US-AC based on the responsibilities of comprehending and transferring requirements. Subsequently, the quality analyst updates the US-AC to supplement some testable details. After that, the US-AC is further updated by other team members based on their expertise. In the deduplication phase, we utilize the Spacy (Honnibal et al. 2020), an open-source Python library, to compute the similarity score of the generated acceptance criteria, which aims to eliminate the redundancy and keep the most informative items in the US-AC instance. The finalized US-AC will serve as a working solution reference for the practitioners.

The primary objective of this work is to propose an automated acceptance criteria generation framework, namely SimAC, which aims to reduce the initial human effort involved in GAC during user story elaboration in agile software development. To gain an in-depth understanding of the effectiveness and practicality of SimAC, we proposed the following Research Questions (RQs) to guide our study.

RQ1 *How is the generality of SimAC employed in different LLMs?* This RQ investigates whether SimAC carries universal effectiveness on diverse LLMs. To answer it, LLMs with different families and parameter sizes are included in experiments.

RQ2 *What is the contribution of each role group in SimAC?* SimAC incorporates the cooperative endeavors of three distinct role groups in GAC, adhering to

agile practice. In this RQ, we aim to discern the individual contributions of these role groups.

RQ3 How do K-shot promptings impact the performance of SimAC? Within SimAC, a diverse set of examples are integrated into prompts to guide the LLM in GAC. Hence, this RQ aims to examine the impact of the number (i.e., K) of interactive examples.

RQ4 How do different settings of the similarity threshold impact the performance of SimAC? In the deduplication phase in SimAC, the similarity score is utilized to eliminate the redundancy. Different settings of the similarity score threshold produce distinct outcomes. Thus, this RQ aims to explore the impact of these settings on the SimAC's performance.

Considering the incorporation of industry perspective and the lack of ground truths in the US-AC dataset, we adopt human evaluation in this study to evaluate both the quality and the relevance of the auto-generated US-AC. The quality assessment aims to determine whether the US-AC follows INVEST (Independent-Negotiable-Valuable-Estimable-Small-Testable) standards as proposed in Cohn (2004). Meanwhile, the relevance assessment serves to measure the extent to which the auto-generated US-AC provides relevant information compared to human-crafted acceptance criteria, with indicators of completeness and validity.

Owing to the huge overhead on time during the human evaluation, we randomly selected 30 samples of US-AC for analysis. During the quality evaluation, we engaged experienced agile practitioners to score US-AC from 0 to 4 in each individual metric of INVEST. Furthermore, to evaluate the relevance, practitioners were invited to familiarize themselves with the project context and then crafted gold standards for the aforementioned 30 examples. The gold standard serves as a benchmark for evaluating completeness and validity.

The quality assessment reveals a significant improvement in satisfaction across all INVEST metrics for the generated US-AC through SimAC. Besides, SimAC also boosts all its underlying LLMs' performance (including LLaMA and GPT-3.5 series), demonstrating its generality among different backbones. Particularly, *gpt-3.5-turbo* model obtains the highest improvement from 2.89 to 3.21 in terms of the INVEST score on average. Additionally, the relevance assessment indicates that US-ACs generated by SimAC with *gpt-3.5-turbo* model achieve 60.60% completeness and 51.29% validity, representing improvements of 29.48% and 15.56% respectively. The introduction of role-based prompts in SimAC effectively enhanced the completeness of the output across all baselines. The findings indicate that the US-AC generated by our proposed framework can receive satisfactory quality and relevance. Such high-quality output can serve as a reliable reference for further refinement in agile sprint planning, thereby substantially improving RE automation and efficiency.

The primary contributions of our study can be summarized as follows:

- *Propose an automated acceptance criteria generation framework (SimAC) motivated by agile practice.* Simulating the collaboration in the agile practice, we

proposed SimAC, which incorporates the role-based prompt to initiate LLMs to collaborate and generate high-quality acceptance criteria for a given user story. The output provides meaningful references for practitioners, reducing their effort in drafting and refining acceptance criteria. To the best of our knowledge, it is the first automated framework for generating acceptance criteria.

- *Extensive experiments conducted based on industrial evaluation.* We conducted extensive studies to evaluate SimAC implemented on four LLMs covering diverse series and sizes. To reflect the real-world feedback and overcome the ground truth unavailability problem, we engaged industrial practitioners to evaluate the quality of US-AC, demonstrating the effectiveness of SimAC and proving the potential of LLMs in the GAC task.
- *Provision of human-craft US-AC gold standard.* We released the gold standard for the dataset crafted by industry practitioners. This dataset establishes a valuable foundation for further research on agile RE.
- *Open-source the replication package.* We published the source code² of this work to facilitate future research and application in the domain of agile RE.

The rest of the paper is organized as follows. Section 2 discusses the background and related work. Section 3 presents the preliminary analysis. Section 4 introduces the framework proposed in our study, and Sect. 5 demonstrates the experiment design settings. Section 6 showcases the detailed experiment setups and results for answering research questions. Furthermore, we provide comprehensive case studies on Sect. 7. In Sect. 8, the threats of our study are reviewed. Finally, Sect. 9 concludes our study.

2 Background and related work

This section introduces the background of the user story and acceptance criteria, as well as related work concerning RE automation.

2.1 US-AC in agile RE

2.1.1 User story and acceptance criteria

The widespread adoption of agile software development methods increases the usage of user stories as requirements elicitation methods in industry (Dalpiaz and Brinkkemper 2018). User stories and acceptance criteria are two key concepts in agile requirements specification (Ferreira et al. 2022), where the former is a brief description of some aspects of a software system (Sedano et al. 2019). The latter encompass functional behavior, business rules, and quality aspects to be tested and are attached to a user story (Ferreira et al. 2022).

² <https://github.com/liyishu0308/SimAC.git>.

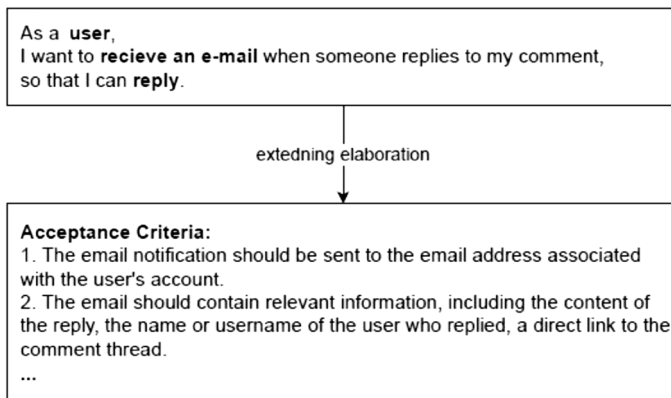


Fig. 2 An example of elaborating user story associated with acceptance criteria

The popular and widely adopted Connextra template (Dalpiaz and Brinkkemper 2018; Ferreira et al. 2022; Ahmed et al. 2023) of user story is: “As a ⟨type of user⟩, I want to ⟨goal⟩, [so that ⟨reason⟩].” In this template, the user story captures the essential elements of a requirement: ⟨type of user⟩ defines *who* it is for, ⟨goal⟩ denotes *what* is expected from the system, and optionally, ⟨reason⟩ expresses *why* it is important (Lucassen et al. 2016).

Within the project environment, user stories exhibit a spectrum of granularity levels, which contains epic, story, feature, and theme (Dimitrijević et al. 2015; Wautelet et al. 2016; Ferreira et al. 2022). Wautelet et al. (2016) introduced a macro-level structure of user story. Initially, epics serve as user stories with a high level of abstraction in the early stages of agile project development, requiring subsequent refinement into smaller stories for more precise requirement articulation. Subsequently, when introducing features into the system, smaller stories can be derived from these epics (Dimitrijević et al. 2015). At the story level, the user story should contain a feature or part of the features that express the delimited characteristics of a system. Stories sharing common features may be consolidated into the same theme, representing a collection of lower-level user stories.

At the story level, a user story is usually attached to acceptance criteria constructed for the development process (Wautelet et al. 2016; Ferreira et al. 2022). Acceptance criteria define the functionalities required by a user story (Cohn 2004), containing preconditions, triggering events, and post conditions (Gupta et al. 2019). According to the interviews conducted in Bjarnason et al. (2015), the two most popular formats of acceptance criteria are scenario-oriented (Given/When/Then template) and rule-oriented (checklist template).³ This paper adopts the checklist template because some cases are not applicable to scenario-oriented structures. For instance, some user stories describe system-level functionality that acceptance criteria are hard to represent as the Given/When/Then template.

³ <https://www.altexsoft.com/blog/acceptance-criteria-purposes-formats-and-best-practices/>

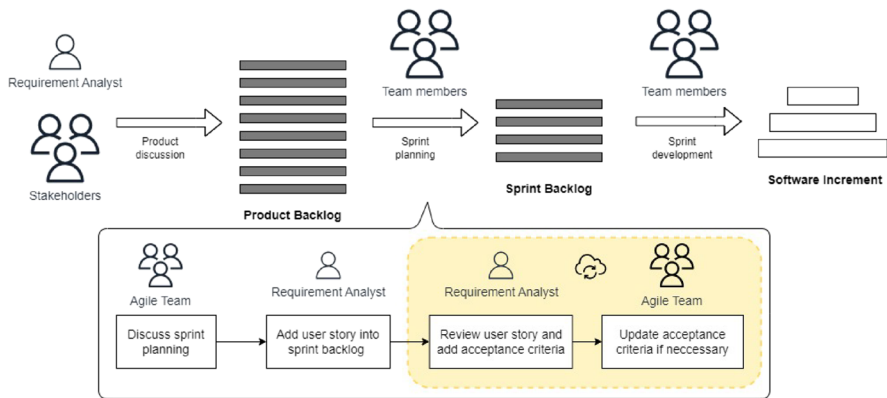


Fig. 3 Agile practice in sprint planning: collaborative acceptance criteria generation from agile team members

In this paper, we discuss the user story at the story level, which can be elaborated with the acceptance criteria in agile project development. Figure 2 illustrates an example of US-AC. Following the Connextra template, the user story defines the user type as a standard user. Then, it specifies the goal, which is to receive an email notification so that the user can reply to the comments through email. The user story provides a brief introduction to the functionality that needs to be developed. In the sprint planning phase, the user story is further elaborated with detailed acceptance criteria. For example, it extends the elaboration of the user story to specify how the email should be notified and what contents should be included in the email. These details, such as email content, define what should be included to achieve the goal of receiving an email notification. Generated acceptance criteria serve as a valuable reference for delivering the working solution. In accordance with the guidelines delineated in the US-AC, developers can determine an appropriate data format for email content, while testers can formulate a comprehensive series of test cases designed to validate the functionalities.

2.1.2 Collaboration in GAC

The most adopted practice in agile development is the hybrid method of Scrum and eXtreme Programming (XP) (Sverrisdottir et al. 2014). Sverrisdottir et al. (2014) presented the key activities in Scrum practice. Building upon these established practices, we elaborate on the specific procedures involved in GAC from the sprint planning phase onwards, as depicted in Fig. 3.

Within the lifecycle of agile project development, the delineation of the product backlog, which encompasses user stories, defects, tasks, and other work elements, is an endeavor undertaken collaboratively by the requirement analyst and stakeholders (Sedano et al. 2019). In the recurrent sprint planning sessions, the requirement analyst selects the user story from the product backlog and integrates it into the sprint backlog after agile team discussion.

Subsequently, GAC requires collaboration from the agile team members. The workflow of GAC is illustrated in the dashed box of Fig. 3, where the requirement analyst is responsible for creating an initial version of acceptance criteria (Sverrisdottir et al. 2014; Diebold et al. 2015), and then other agile team members update or revise the previously generated acceptance criteria if necessary (Sedano et al. 2019). Finally, the refined US-AC is delivered to the team as the reference for subsequent sprint development activities.

2.2 LLMs and RE

2.2.1 NLP techniques in RE

Many RE activities (e.g., requirements elicitation and specifications) require text analysis and human analyst engagement and have benefited from the Natural Language Processing (NLP) techniques in the past decades. According to the systematic mapping study conducted in Zhao et al. (2021), the top-ranked NLP tools developed for RE are those designed for requirement modeling (Lucassen et al. 2017) and detection (Ferrari and Gnesi 2012), followed by those aimed at extraction (Hotomski and Glinz 2018) and classifications (Carreño and Winbladh 2013). Additionally, Zhao et al. (2021) reports the commonly used NLP techniques in RE research are based on lexical or syntactic analysis, such as Parts Of Speech (POS) tagging (Ali et al. 2019) and tokenization (Almanaseer et al. 2022). These NLP-based tools contribute a lot to RE automation.

For example, in requirements elicitation activities, NLP-based tools are developed to reduce the human effort needed to understand the project requirements. Xiao et al. (2012) proposed a general framework incorporating syntactic and semantic analysis to extract model instances of natural language software documents and produce formal specifications. By leveraging universal sentence encoder (Cer et al. 2018) and BERT (Devlin et al. 2018), Nema et al. (2022) proposed classifying the privacy information and employing summarized topic clusters by extracting representative reviews from the cluster. In the requirements modeling activities, the development team will derive the domain model for the requirements, where the NLP-based tools can assist the human analyst. For instance, Thakur and Gupta (2016) developed AnModeler, a tool for generating analysis models from software requirements specified as use cases. It extracts the Type Dependencies (TD) and POS tags of sentences and then identifies sentence structures using a set of framed rules. Finally, the TDs, POS tags, and sentence structures are utilized to generate the class diagrams. Several NLP-based tools have been developed to assist with requirements management and classify functional requirements (FRs) and non-functional requirements (NFRs). Hey et al. (2020) proposed NoRBERT, which fine-tunes BERT on FRs and NFRs classification tasks.

In light of the impressive capabilities demonstrated by recently released LLMs in NLP-related tasks, RE researchers have started exploring the breakthrough of integrating with LLMs. Zhang et al. (2023) conducted a preliminary evaluation of ChatGPT utilized in two types of requirements information retrieval tasks: requirements

classification and information extraction. The results highlight the impressive ability of ChatGPT to retrieve requirements information from different types of artifacts involving multiple languages under a zero-shot setting. They argued it is worthwhile for the research and industry communities to study generative LLMs for RE tasks and develop corresponding tools. Ferrari et al. (2024) examined the efficacy of ChatGPT in generating Unified Modeling Language (UML) diagrams and sequence diagrams based on natural language requirements. Their findings suggest that augmenting ChatGPT with additional contextual cues and domain-specific knowledge holds promise for enhancing its capacity in UML diagram generation. Li et al. (2024) incorporates step-by-step reasoning into the prompting to derive the class diagram from user stories. The results demonstrate that LLM-based approaches, particularly incorporating well-crafted few-shot prompts, outperform students who were provided proper guidance in identifying classes from user stories. Integrating stepwise reasoning in prompts further improves the comprehensiveness of relationship identification among classes. Spoletini and Ferrari (2024) proposed a vision to employ fine-tuned LLMs to support the formalization and explanation of requirements. The study outlines a roadmap towards realizing this vision, which includes leveraging LLMs to automatically produce formalized properties, using LLMs to explain and generalize counterexamples to rectify the resulting code, and employing LLMs to generate explanations and recommendations.

2.2.2 Overview of LLMs

LLMs have showcased their remarkable capabilities in a variety of tasks, including machine translation (Yang et al. 2024), summarization (Xue et al. 2024), and question-answering (Liu et al. 2024). LLMs undergo unsupervised training using billions of open-source text tokens to achieve comprehensive language modeling. The utilization of diverse data sources during LLM training enriches their cross-domain knowledge, thereby facilitating effective generalization for corresponding downstream tasks (Zhang et al. 2023).

There are three categories of LLMs: encoder-only model, encoder-decoder model, and decoder-only model (Fan et al. 2023). Encoder-only LLMs accept an input sequence and transform it into a vectorized representation, which are text understanding models without text generation capability (Lewis et al. 2019). These encoder models, exemplified by the widely used BERT (Devlin et al. 2018), can be fine-tuned for various downstream tasks like sentiment analysis (Hoang et al. 2019) and named entity recognition (Hakala and Pyysalo 2019). In contrast, encoder-decoder LLMs incorporate a decoder network, which generates an output sequence through iterative produced tokens or symbols based on the context vector and previously generated tokens. Encoder-decoder models, such as BART (Chipman et al. 2010) and T5 (Raffel et al. 2020), are particularly good at tasks necessitating intricate mappings between the input and output sequence, particularly where capturing the relationships between the elements in both sequences is crucial. Some common use cases for encoder-decoder models include machine translation (Cho et al. 2014) and summarization (Ma et al. 2023). Decoder-only models typically adopt architectures such as autoregressive models, wherein the output is generated token-by-token,

with each token conditioned on preceding tokens and context. Unlike encoder-only and encoder-decoder LLMs, decoder-only LLMs do not have an encoder component to process the input data but only a decoder component that directly generates an output sequence based on a given context or input. The autoregressive nature inherent in decoder-only models ensures token-by-token generation of output, utilizing previously generated tokens as context for subsequent token generation (Fan et al. 2023). Decoder-only models have proven to be highly effective in various NLP tasks that can be framed as autoregressive problems, such as text generation (Subramanian et al. 2018).

Recent studies (Nijkamp et al. 2022; Fried et al. 2022; Nijkamp et al. 2023) mostly focus on training LLMs with decoder-only architecture because of its effectiveness and flexibility (Zhao et al. 2023). The decoder-only LLMs demonstrate superior zero-shot and few-shot generalization capacities compared to other architectures (Zhao et al. 2023; Wang et al. 2022). Besides, the parameter sizes of encoder-decoder models are relatively small, such as CodeT5 (Wang et al. 2021) and UniXcoder (Guo et al. 2022) less than 1B, resulting in a deficiency in emergent abilities (Zhao et al. 2023), rendering them unsuitable for tasks without fine-tuning. Given the inherent nature of GAC and the prevalent utilization of decoder-only LLMs in practice, this study exclusively focuses on decoder-only LLMs.

Among the decoder-only LLMs, the GPT (Generative Pre-trained Transformer) families and LLaMA series stand out for their popularity and prowess due to the large corpora on which they are trained. For example, GPT-3 is trained on 45TB of text data and has 175 billion parameters. LLaMA is trained on 1.4 trillion tokens with a variety of model sizes ranging from 7 billion to 65 billion parameters (Fan et al. 2023).

While the content generated by LLMs is often grammatically correct, they may not always be semantically correct (Ozkaya 2023). Despite their robust performance, LLMs sometimes struggle to produce accurate answers when faced with complex tasks. In response, researchers have proposed various prompting techniques to enhance the reasoning capability of LLMs in NLP tasks (Zhang et al. 2023).

2.2.3 Prompt engineering in LLMs

Prompts are instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities of generated outputs (White et al. 2023). Recent studies have shown that well-designed prompting methods, such as chain-of-thought (CoT) promoting (Wei et al. 2022; Kojima et al. 2022), can significantly improve the performance of LLMs.

Recent research indicates an increasing interest in utilizing multi-agent collaboration techniques to unleash the power of LLM further. For instance, VirtualGPT (Wu et al. 2023) and HuggingGPT (Shen et al. 2023) explored the collaboration of LLMs with other models, specifically the LLMs are used as decision centers to control and invoke other models to handle vision, speech, and signals. Dong et al. (2023) developed a self-collaboration prompt framework to guide ChatGPT to understand collaboration in the task of code generation. These studies primarily employ qualitative analysis of case studies to demonstrate the efficiency of the proposed framework. In

(Dong et al. 2023), the authors also conducted a quantitative analysis to evaluate the self-collaboration code generation. Our work aligns with the concept of collaboration of LLMs, but it is uniquely positioned with the field of agile RE, which provides practical guidance to automation on GAC in agile RE.

3 Preliminary analysis

In this section, we perform a preliminary analysis of the GAC process to investigate what roles should be integrated into our proposed framework and how the procedures should be implemented.

3.1 Role division

In an agile team, there are several key roles that collaborate to deliver software increment, as presented in Fig. 3. These roles may vary slightly depending on the specific agile framework being used but commonly include Product Owner (PO), Business Analyst (BA), Quality Analyst (QA), Developers (Dev), Project Manager (PM), and other relevant team members.

PO and BA are two roles that mainly participated in agile requirement activities (Wang et al. 2014). The PO acts as a liaison between the development team and stakeholders, ensuring that the product aligns with stakeholder expectations and requirements. On the other hand, the BA focuses on analyzing the business needs of the organization and translating them into functional requirements for the development team to implement. It is noted that in some agile teams, the PO is frequently staffed by a BA (Kelly 2019) owing to their shared expertise and knowledge. Due to their joint responsibility for documenting acceptance criteria as the initial step in the GAC process, we have combined these two roles for introduction to the LLM. Following the convention observed in Wang et al. (2014), we use the umbrella name Requirement Analyst (RA) to represent these two roles. Both of the roles share the essential responsibilities of comprehending and transferring requirements, making them integral contributors to initiating the US-AC.

Within an agile team, the QA is crucial for ensuring that the software product meets the desired quality standards. Their primary responsibilities involve test design and execution. Given that acceptance criteria serve as the reference for acceptance testing (Pandit and Tahiliani 2015; Fischbach et al. 2020) in agile development, the QA assumes a pivotal role in refining acceptance criteria by incorporating testable specifics. Therefore, the QA constitutes the second group introduced to LLM for GAC.

During sprint planning sessions, the agile team actively participates in discussions, as depicted in Fig. 3. Other participants, such as developers and project managers, are responsible for updating acceptance criteria based on their respective expertise, if necessary. However, empirical observations suggest that these roles exert less effort on GAC and provide support only when essential. Given this

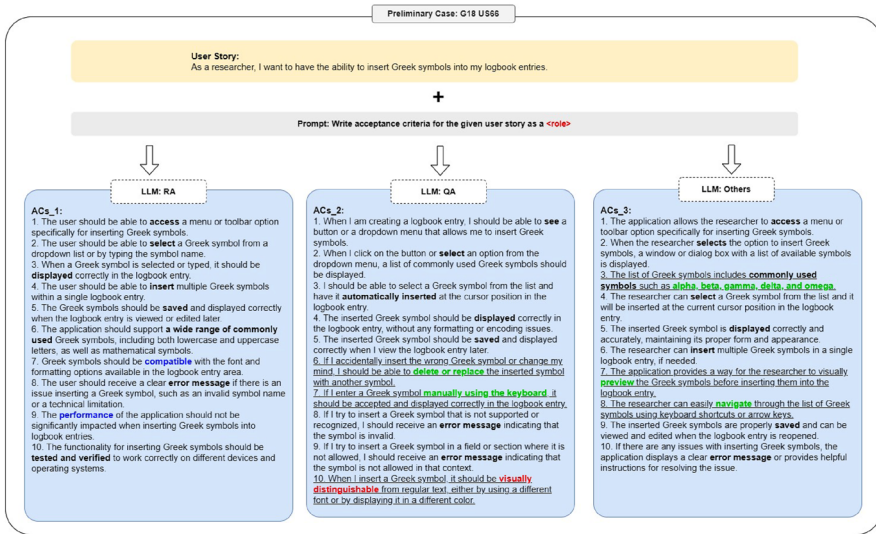


Fig. 4 Acceptance criteria generated by LLM acting in different roles

observation, the remaining roles are grouped as Others and introduced to the LLM to provide supplementary details.

Based on the above observations and analysis, we categorize the roles into three primary groups in our proposed framework for simulating the collaboration process in GAC: RA, QA, and Others.

3.2 Procedures

Following the agile practice presented in Fig. 3, the acceptance criteria are firstly crafted by the RA, then it is collectively revised by the agile team members.

To establish an appropriate procedural design introduced to the LLM, we constructed a preliminary analysis dataset D_p . Excluding projects selected for evaluation, D_p was constructed by randomly selecting two samples from each of the remaining group projects. During the preliminary analysis, we designed a basic command (referred to as cmd_t) to direct the LLM to act in a specific group of roles separately. Three groups of roles in cmd_t are specified in angle brackets (<role>).

cmd_t: Write acceptance criteria for the given user story as a <role>.

Our findings demonstrate a high level of semantic similarity among the acceptance criteria generated by different roles for the overall US-AC. Specifically, the similarity values are 0.9748 for RA vs. QA, 0.9751 for RA vs. others, and 0.9762 for QA vs. Others. Despite the pronounced similarity observed, it is noteworthy that each role group contributes distinct nuances based on its respective expertise. To evaluate the significance of such nuances, our researchers meticulously examined the newly introduced contents. The investigation indicates that 72.22% of the additions from

QA and 61.11% of those from Others were deemed worthy of further discussion in agile practice.

Fig. 4 presents an example. The user story delineates the researcher's desire to insert Greek symbols into the logbook entries. The acceptance criteria, denoted as ACs_1, ACs_2, and ACs_3, are generated by the LLM acting in roles as RA, QA, and Others, respectively. The bolded terms accentuate key phrases within each item. The three sets of outputs share substantial information, indicating a high similarity. The acceptance criteria produced by RA predominantly emphasize functional descriptions, including the basic operational flow on the insertion and specific consideration on performance and compatibility, as highlighted in blue. Furthermore, the outputs generated by LLMs acting as QA and Others introduce new criteria, distinguished by underlining and color-coding, with green indicating topics worthy of discussion in agile practice and red denoting less pertinent aspects. In this case, Points 6 and 7 within ACs_2 offer valuable QA insights into test scenarios, encompassing the edit options and the manual insertion methods. Additionally, Point 3 in ACs_3 offers analogous information to Point 6 in ACs_1 while supplementing nuanced details, such as examples of more commonly used symbols. These details are more pertinent from the developers' standpoint, relating to implementation specifics. Such role-specific information contributes to meaningful collaboration within the GAC.

The observed high similarity suggests that there is no necessity to create separate sets of US-AC for each role group. Instead, it is advisable to modify the acceptance criteria when specific updates are required, aligning with actual agile collaboration. The significance of the newly introduced nuances underscores the necessity for collaboration among diverse role groups because such additions yield meaningful discussion points. Consequently, we introduce a collaborative procedure as *create-update-update* to our proposed framework.

4 Framework

Inspired by the insights derived from agile practice, we propose SimAC, which is a collaboration-simulation framework to empower LLMs in the task of GAC. This section presents the problem formulation and detailed design of our proposed framework.

4.1 Problem formulation

This paper aims to leverage LLMs to generate acceptance criteria, thereby extending and defining the “done” of the story. More formally, given the user story x and the designated procedures f leveraging the LLMs, our objective is to formulate a series of prompts p introduced to the LLM that yields an estimation of US-AC, namely $\hat{y}$, which is expected to approximate the human-satisfied US-AC y . Subsequently, the generated $\hat{y}$ will be delivered for agile sprint planning discussion. The problem can be formulated as follows.

$$\arg \min_p ||y - \hat{y}||, \text{ where } \hat{y} = f(x, p) \quad (1)$$

4.2 Overview of SimAC

As shown in Fig. 5, to simulate the inherent collaborative process of GAC in agile practice, we design SimAC, a two-phase framework, including the collaboration phase and the deduplication phase. SimAC instructs the LLM to act as different role groups, each tasked with distinct responsibilities aligned with the practices in the dashed frame of Fig. 3. During the collaboration phase, the LLM sequentially performs different role groups to create or update the acceptance criteria for the given user story. These acceptance criteria are then processed in the deduplication phase, which aims at eliminating redundant information.

4.3 Phase I: collaboration

The collaboration phase comprises two stages. Firstly, the LLM is initiated by role-based prompts with demonstrated examples. Following this, the LLM acts as the designated role group for GAC.

4.3.1 Role-based prompt initialization

Figure 6 presents the detailed role-based prompt (P_{ro}) tailored for SimAC. This prompt encompasses three essential components: context, role instruction, and K-shot prompting.

Context In order to facilitate and enhance collaborative efficiency, the definition of GAC is specified in the context of the prompt. As dictated in the prompt, the LLM is designated as a collaborative tool to generate acceptance criteria for the given user story. Following the good practices of prompt guided by OpenAI,⁴ the context encompasses four facets aimed at instructing the LLM on its behavior in GAC: roles engaged in the activity, procedures of GAC, guidelines of writing clear acceptance criteria, and output formats.

As previously discussed, to simulate the agile practice as highlighted with the dashed frame in Fig. 3, the LLM is initially instructed to act as three distinct role groups: RA, QA, and others. Subsequently, a procedural workflow is introduced, which aligns with a sequence of activities: creation by RA, update by QA, and second-round update by Others. In accordance with the recommendation proposed in (Ferreira et al. 2022), the guidelines for writing clear acceptance criteria are introduced in the context. Finally, the designed output format is specified, which is a mechanism aimed at facilitating subsequent data extraction and processing endeavors.

Role instruction The prompt employed to initiate the LLMs primarily consists of comprehensive role instructions, which share the knowledge of role-based

⁴ <https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api>.

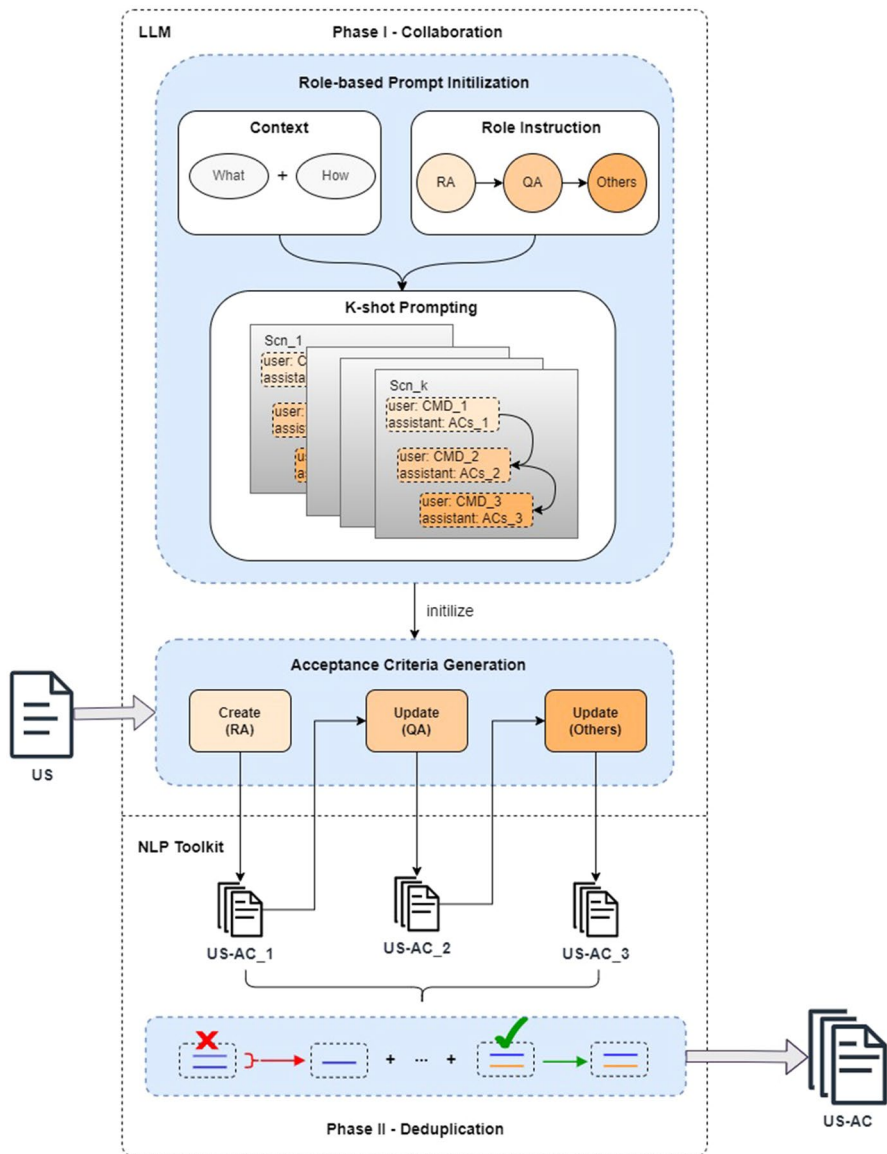


Fig. 5 Two phases of GAC activities: I. collaboration (Stage I: role-based prompt initialization; Stage II: acceptance criteria generation); II deduplication

collaborations in practice. For each participating role, the responsibilities and aspects that need to be considered in the activity are introduced to the LLM.

The expertise of each role engaged in the activity is emphasized so that the LLM can receive a better understanding and act as different experts. As discussed in role division, three groups of roles are introduced. Following the responsibility

Prompt = Context + Role Instruction + K-shot Prompting	
Context	What You are a collaborative tool to write acceptance criteria for a given user story from perspectives of different roles. The activity of writing acceptance criteria will be completed by four roles: Product Owner, Business Analyst, Quality Analyst, and Others. The procedure of writing acceptance criteria for a given user story should follow the steps as follows: Step 1, the product owner and business analyst write the majority of the acceptance criteria together. The acceptance criteria may encompass functional behaviour and business rules. Step 2, the quality analyst reviews the acceptance criteria and update the acceptance criteria from quality aspects. Step 3, the other team members (e.g., project manager and developers) in the agile team review the acceptance criteria and update the acceptance criteria if the user story requires an evaluation from other aspects, for example, version management and coding management. Step 4, consolidate the final acceptance criteria, remove the items with similar syntax, and pick the most important items. How Here are some guides for writing clear acceptance criteria: 1. Use popular linguistic patterns when writing the acceptance criteria. 2. Identify the artifact by a unique Id, starting from 1. 3. Use a consistent and straightforward syntax to write each fragment of the acceptance criteria. 4. Avoid writing general and vague acceptance criteria. 5. Identify if acceptance criteria concern a success or a failure case. 6. Enrich the acceptance criteria with failure cases. The generated items of acceptance criteria should be placed between tags "<start>" and "<end>". Desired output format: Acceptance Criteria: <start> 1. <insert first acceptance criteria here> 2. <insert second acceptance criteria here> ... <end> Length: The acceptance criteria for a given user story should be limited to a maximum of 10 items.
	RA Role: Product Owner Responsibility: As a product manager or product owner, you may be responsible for writing acceptance criteria for the stories in your product backlog. When writing acceptance criteria, the product owner should consider from following aspects. 1. Define the completeness of the user story. 2. Write the acceptance criteria that encompass functional behaviour. QA Role: Business Analyst Responsibility: As a business analyst, you may be responsible for working closely with the product owner to understand the customer needs and translate them into acceptance criteria. When writing acceptance criteria, the business analyst should consider from following aspects. 1. Define the specific, measurable and unambiguous acceptance criteria. 2. Write the acceptance criteria that encompass business value. Others Role: Quality Analyst Responsibility: As a quality analyst, you may be responsible for collaborating with the product owner and business analyst to validate that the criteria are testable and that the testing process is achievable. When writing acceptance criteria, the quality analyst should consider from following aspects. 1. Verify the acceptance criteria is testable or not. If not, update the acceptance criteria. 2. Check the acceptance criteria cover enough failure cases or not. If not, supplement the acceptance criteria. Role: Others Responsibility: As other team members, you may be responsible for helping to evaluate the acceptance criteria requires to be updated from perspectives of project management or code management. When writing acceptance criteria, the other team members should consider from following aspects. 1. Verify there is release plan or release items in user story. If yes, update acceptance criteria from perspectives of release version management. 2. Verify there is budget items in user story. If yes, update acceptance criteria from perspectives of budget management. 3. Verify there is code management issue in user story. If yes, update acceptance criteria from perspectives of code management.
K-shot Prompting	user: "Write acceptance criteria for the given user story as the Product Owner and Business Analyst." + {us_1} assistant: {ac_ra_1} user: "Update or supplement the above acceptance criteria for the given user story as a Quality Analyst if necessary." + {us_1} assistant: {ac_qa_1} user: "Update or supplement the above acceptance criteria for the given user story as other team members if necessary." + {us_1} assistant: {ac_others_1}

Fig. 6 Three components in the role-based prompt (P_{ro}): context, role instructions, and K-shot prompting

of different role groups in the agile team discussed in Kelly (2019), we deliver the detailed instructions to different role groups as follows.

- *Requirement Analyst (RA)*

- *Product Owner* (PO) should ensure the completeness of the user story with product functionalities.
- *Business Analyst* (BA) is responsible for presenting the business value in the user stories.
- *Quality Analyst* (QA) should verify whether the acceptance criteria are testable and ensure the acceptance criteria should cover as many different case scenarios as possible.
- *Other Team Members* (Others)
 - *Developer* (Dev) should assist with considering the implementation details to be measurable or not.
 - *Project Manager* (PM) takes care of the release schedule and budget management-related stuff.

K-shot prompting As illustrated in Brown et al. (2020), GPT-3 exhibits promising results in few-shot settings in some NLP tasks. Illustrative examples are integrated into prompts to enhance the comprehensive understanding of collaborative role activities. Following the collaborative work process, we incorporate K-shot promptings. In each shot, the LLM acts in accordance with the prescribed procedure. The prompt examples replicate different scenarios involving different roles in the activity. Further details regarding the experimental settings of the examples will be discussed in Sect. 5.

4.3.2 Acceptance criteria generation

Following initialization with the role-based prompt, the LLM adheres to predefined procedures, namely *create-update-update*, to execute the task of GAC using the designated command for each step. Given that the RA serves as the umbrella name within SimAC, the LLM may lack a comprehensive understanding of its meaning. To mitigate this potential ambiguity, we incorporate the roles of PO and BA in the command introduced to the LLM. The commands for each step are presented as follows.

cmd_1 (command for creation): *Write acceptance criteria for the given user story as the Product Owner and Business Analyst.*

cmd_2 (command for 1st update): *Update or supplement the above acceptance criteria for the given user story as a Quality Analyst if necessary.*

cmd_3 (command for 2nd update): *Update or supplement the above acceptance criteria for the given user story as other team members if necessary.*

Figure 7 demonstrates the step-by-step prompt with commands (exemplified in *gpt-3.5-turbo*). Initially, the LLM acts as the role of RA to generate a set of acceptance criteria for the given user story. Subsequently, the LLM transitions to the role of QA

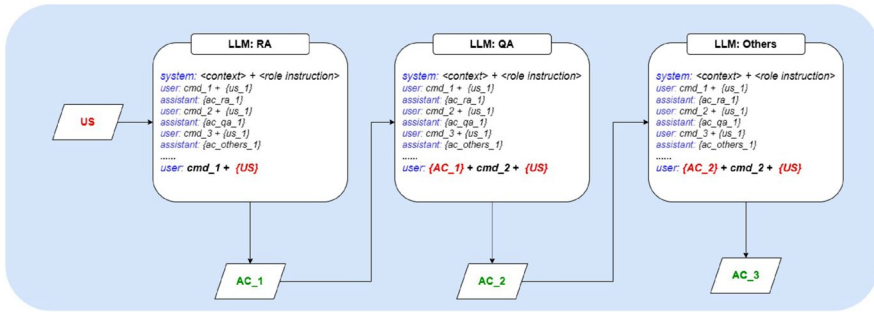


Fig. 7 Collaborative roles-playing LLM in three-steps of GAC (exemplified in *gpt-3.5-turbo*)

to update the acceptance criteria. Finally, the LLM assumes the role of Others, contributing to supplement the US-AC with additional details from their own expertise. The abbreviations used in Fig. 7 are elucidated as follows.

- **<context>** - the context component in the prompt (as presented in Fig. 6).
- **<role>** - the role instruction component demonstrated in prompt (as presented in Fig. 6).
- **cmd** - the commands for each step (as listed in the previous discussion).
- **{us}** and **{ac}** - the user stories and the exemplified acceptance criteria generated by different roles as demonstrated in examples (as presented in Fig. 6)
- **{US}** and **{AC}** refer to the given user story and the acceptance criteria generated from each step.

4.4 Phase II: deduplication

Throughout the collaboration process, each procedural step yields its own set of acceptance criteria. Nevertheless, different roles may introduce modifications to the US-AC that convey similar meanings. To alleviate redundancy, SimAC employs a semantic similarity processing mechanism to scrutinize and refine the US-AC prior to its finalization.

In the deduplication phase, the acceptance criteria extracted from the three sets of US-AC are arranged in pairs to calculate similarity scores. Once the scores surpass a specified threshold, one of the acceptance criteria in this pair will be excluded to eliminate redundancy.

Cosine similarity (c_{sim}) is a widely implemented metric for measuring similarity between sentences (Achananuparp et al. 2008). The sentences are converted into vectors based on TF-IDF. The c_{sim} between two sentence vectors (S_A and S_B) is computed by taking the dot product of the vectors and dividing it by the product of their magnitudes. Mathematically, it is defined as:

$$c_{sim} = \frac{S_A \cdot S_B}{\|S_A\| \|S_B\|} \quad (2)$$

5 Experimental setup

In this section, we outline the experimental setup of this study, including research questions with corresponding experimental design, dataset, baselines, implementation, and evaluation employed.

5.1 Research questions

Here, we introduce four research questions in detail, outlining the respective experimental designs to address them.

5.1.1 RQ1: How is the generality of SimAC employed in different LLMs?

This research question aims to assess the effectiveness of SimAC and its generality on various LLMs. In this study, different LLMs of diverse sizes are selected for investigation, including *gpt-3.5-turbo*, *text-davinci-003*, *OpenLLaMA-7B*, and *OpenLLaMA-13B* as further introduced in Sect. 5.3. Through this examination, we are able to quantitatively assess the generalization capability of SimAC within the domain of GAC and explore the feasibility of automating this task.

In SimAC, the LLM is prompted by P_{ro} (as presented in Fig. 6) to simulate collaborative interactions. To facilitate the comparative analysis, we meticulously designed a regular prompt (P_{re}), which requests the LLM to generate acceptance criteria directly. The P_{re} excludes the role instruction, which only introduces the context and good practices. The structure and details of the regular prompt template are outlined in Fig. 8. The command introduced to the LLM initiated by the regular prompt is shown below:

cmd: *Write acceptance criteria for the given user story.*

5.1.2 RQ2: what is the contribution of each role group in SimAC?

SimAC incorporates the collaborative efforts of three distinct role groups in GAC, following agile practices. Each role group brings specific expertise to the GAC process. In RQ2, we aim to discern the individual contributions of these role groups.

To achieve this, we deconstruct the role groups as three combinations (denoted as R): (1) RA (i.e., $R = 1$), (2) RA and QA (i.e., $R = 2$), and (3) RA, QA, and Others (i.e., $R = 3$). The assessment of each combination enables a comprehensive analysis to evaluate the impact of each role group contribution within the SimAC.

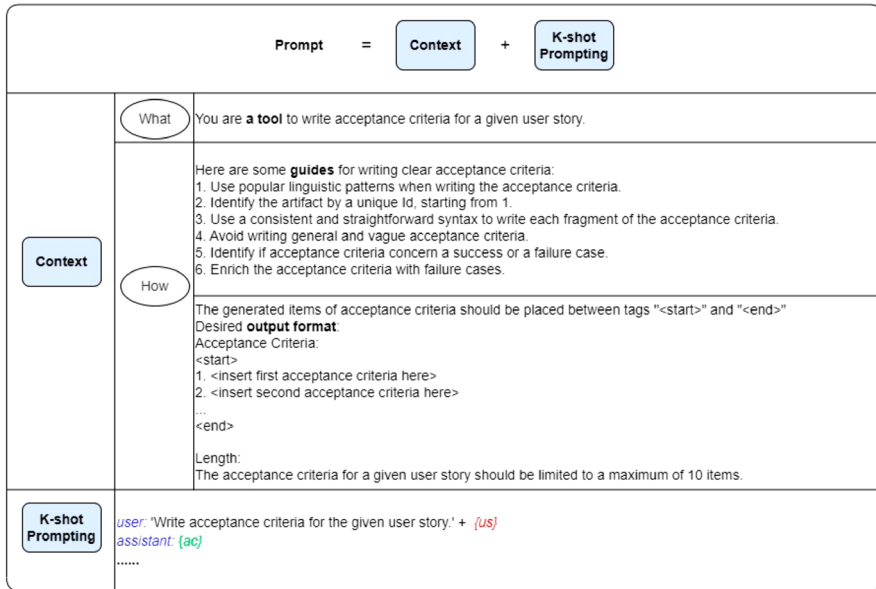


Fig. 8 Two components in the regular prompt (P_{re}): context and K-shot prompting

5.1.3 RQ3: how do K-shot promptings impact the performance of SimAC?

Within SimAC, a variety of examples are integrated into prompts to guide the LLM in GAC. While adding more examples offers increased reference, it also consumes valid prompt capacity and may lead to redundancy. Thus, RQ3 targets to explore the impact of K-shot prompting in SimAC.

Due to the maximum token constraint of LLMs' input, we designate the number of examples in K-shot prompting as **0, 1, 3, and 4**. Here, K denotes the number of sets of interactive examples covered. Each set of examples adheres to the *create-update-update* process, comprising three distinct examples. The examples were crafted by the invited practitioners based on their professional expertise. In order to bolster the resilience of LLMs, we deliberately selected some common examples demonstrated in online resources, avoiding reliance on evaluation dataset (D).

Table 1 provides an overview of the examples utilized, demonstrating varying engagement levels from different roles. Drawing from observations in daily practice, some roles tend to make minor modifications to US-AC due to a diminished necessity required from them. Following the *create-update-update* process in GAC, the predominant scenario (sc_1) entails all role groups participating equally in creating or updating the US-AC, as evident from the modification of ac_1 detailed in Table 1. The second scenario (sc_2) involves updates contributed by both QA and Others, albeit with minor alterations to the US-AC. As sc_2 delineated in Table 1, the RA creates ac_2 , while QA and Others slightly update on ac_2 . Furthermore, the third scenario (sc_3) exclusively involves participation from RA and QA, without

Table 1 Description of examples in K-shot prompting generated under different scenarios

K	Scenarios (#)	Examples (#)	Type	Collaboration in GAC
0	0	0	—	—
1	1	3	sc_1	E1.1: RA creates ac_1 for us_1 . E1.2: QA updates on ac_1 for us_1 . E1.3: Others update on ac_1 for us_1 .
3	3	9	sc_1	E1.1: RA creates ac_1 for us_1 . E1.2: QA updates on ac_1 for us_1 . E1.3: Others update on ac_1 for us_1 .
			sc_2	E2.1: RA creates ac_2 for us_2 . E2.2: QA slightly updates on ac_2 for us_2 . E2.3: Others slightly update on ac_2 for us_2 .
			sc_3	E3.1: RA creates ac_3 for us_3 . E3.2: QA slightly updates on ac_3 for us_3 . E3.3: Others do not update on ac_3 for us_3 .
4	2	12	sc_1	E1.1: RA creates ac_1 for us_1 . E1.2: QA updates on ac_1 for us_1 . E1.3: Others update on ac_1 for us_1 .
			sc_2	E2.1: RA creates ac_2 for us_2 . E2.2: QA slightly updates on ac_2 for us_2 . E2.3: Others slightly update on ac_2 for us_2 .
			sc_1	E4.1: RA creates ac_4 for us_4 . E4.2: QA updates on ac_4 for us_4 . E4.3: Others update on ac_4 for us_4 .
			sc_2	E5.1: RA creates ac_5 for us_5 . E5.2: QA slightly updates on ac_5 for us_5 . E5.3: Others slightly update on ac_5 for us_5 .

sc_1 refers to 1st type of scenario

us_1 refers to User Story 1

ac_1 refers to acceptance criteria attached for us_1

E1.1 refers to the 1st version of ac_1 as illustrated in the example

E1.2 refers to the 2nd version of ac_1 as illustrated in the example

contributions from Others. The sc_3 outlined Table 1 illustrates ac_3 is created by RA and slightly updated by QA, with no involvement from Others.

In **zero-shot prompting** ($K = 0$), no collaborative examples are provided to the LLM. Only the context and role instructions are presented. For **one-shot prompting** ($K = 1$), a single set of interactive examples is introduced to the LLM, reflecting the most common scenario (sc_1). In **three-shot prompting** ($K = 3$), three sets of examples covering sc_1 , sc_2 , and sc_3 are provided. However, due to token constraints, **four-shot prompting** ($K = 4$) cannot include six sets of examples covering all three

scenarios. Therefore, our focus is exclusively on sc_1 and sc_2 , as they are the most commonly occurring scenarios observed in daily practice.

5.1.4 RQ4: How do different settings of the similarity threshold impact the performance of SimAC?

In this research question, we aim to identify the optimal threshold (t) for reducing the redundancy of generated US-AC. We perform a pairwise comparison on US-AC generated in D_p . The results reveal the sentence similarity scores fall within a range spanning from 0.85 to 0.95. Within this range, we designate the lower, middle, and upper bounds as potential threshold values, where t is set as **0.85**, **0.9**, and **0.95** in this experiment.

5.2 Dataset

To assess the effectiveness of SimAC, a publicly available dataset (D) as outlined in Dalpiaz et al. (2019) is utilized in our study. Three projects are selected to construct our test dataset (D_t) containing different sizes of actors ($\langle \text{type of user} \rangle$), thereby encompassing anticipated requirements from diverse stakeholders. Project NSF (G11) incorporates multiple actors within its user stories, delineating user needs from both business flow and development progression perspectives. Project UniBath (G24) entails user stories with medium-sized actors predominantly focusing on feature-based descriptions. Conversely, Project Zooniverse (G28) comprises user stories with only a few actors, primarily emphasizing the user flow of both administrative and regular users. These three projects offer representative examples encompassing diverse levels of user story complexity for the purposes of our experimental evaluations. Table 2 presents the details of the selected projects. The random samples refer to the user stories being randomly picked for final evaluation.

5.3 Baselines

Considering GAC is a text generation task, baselines are selected from casual language models. The selected models encompass various LLM families, including GPT families and LLaMA families, spanning a range of parameter sizes (7B, 13B, and over 175B) and originating from diverse sources (both closed-source and open-source).

GPT-3.5 family The GPT-3.5 series models are trained with over 175B parameters. Among the GPT-3.5 families, ***gpt-3.5-turbo*** is a chat text completion model, which allows the roles playing in chat interaction. It is the most capable and cost-effective model in the GPT-3.5 family. Furthermore, ***gpt-3.5-turbo*** was trained in both natural languages as well as codes, which is suitable for comprehending software development processes. Therefore, it has been selected to assess the efficacy of SimAC.

Within the GPT-3.5 family, ***text-davinci-003*** is recognized for its versatility in handling a wide array of NLP-related tasks with higher quality. As part of our

Table 2 Details of selected projects in testing dataset D_t

ID	Acronym	Project description	US (#)	Actors (#)	Random samples
G11	NSF	New version of the NSF website: redesign and content discover	73	22	[30, 63, 53, 37, 26, 57, 65, 2, 40, 58]
G24	UniBath	Institutional data repository for the University of Bath	53	11	[50, 27, 7, 25, 11, 26, 16, 21, 8, 1]
G28	Zooniverse	Citizen science platform that allows anyone to help in research tasks	60	3	[6, 49, 5, 20, 25, 50, 17, 3, 15, 21]

comparative analysis, it is selected as another baseline. This allows us to assess the performance of SimAC with an LLM trained in pure natural language.

Open-LLaMA LLaMA (Touvron et al. 2023) is competitive with the best models developed by Meta AI.⁵ OpenLLaMA (Geng and Liu 2023) is a permissively licensed open-source reproduction of LLaMA. It is published under the business-friendly Apache 2.0 license. It achieves a similar performance as LLaMA, and it is free to use for commercial purposes without any restrictions. OpenLLaMA offers a range of models, including 3B, 7B, and 13B, trained on 1T tokens. In our experiment, **OpenLLaMA-7B** and **OpenLLaMA-13B** models are utilized to evaluate the generality of SimAC across LLMs with different parameter sizes.

5.4 Implementation

Regarding LLMs, we implement GPT-3.5 by invoking OpenAI's API.⁶ According to the guidelines, the temperature values vary between 0 and 2. Higher values increase the randomness of the output, while lower values will make the results more focused and deterministic. Therefore, the default *temperature* = 1 is adopted to achieve moderate results. For open-source LLMs, such as OpenLLaMA, we instantiate them with their replication packages and load their weights from HuggingFace.⁷ Additionally, the parameter *max_new_tokens* is configured to 500 in the OpenLLaMA series, which is deemed an appropriate length for the newly generated acceptance criteria.

Towards SimAC, the settings of its backbone LLMs are kept as default. In the collaboration phase, the role-based prompt (P_{ro}) is constructed as Fig. 6 to initialize the LLMs. Additionally, Spacy⁸ is employed to compute the cosine similarity in the deduplication phase. Considering the diversity of text background generated, the vector package *en_core_web_lg* is adopted for calculating the similarity score. The package is trained with about 500k word vectors, whereas the calculation of US-AC similarity can benefit from a large vocabulary.

5.5 Evaluation

We employ both quantitative and qualitative methods to evaluate the proposed SimAC framework. Quantitatively, the evaluation covers quality and relevance metrics. Qualitatively, specific cases are selected for an in-depth examination to assess the effectiveness of the SimAC framework.

⁵ <https://ai.meta.com/>

⁶ <https://platform.openai.com/docs/models/gpt-3-5>.

⁷ <https://huggingface.co/openlm-research>.

⁸ <https://spacy.io/usage/linguistic-features>.

5.5.1 Metrics

Quality In agile practice, the product backlog items are commonly represented as user stories with the extension of acceptance criteria (Sedano et al. 2019). Following the principles behind the Agile manifesto (Manifesto 2001), the most well-known quality evaluation framework for product backlog items is the six attributes method called **INVEST** (Cohn 2004; Dimitrijević et al. 2015; Buglione and Abran 2013; Khanh et al. 2017). The metrics, introduced by Bill Wake,⁹ stands for **I**ndependent-**N**egotiable-**V**aluable-**E**stimable-**S**mall-**T**estable. The survey in Lucassen et al. (2016) reveals that a majority of practitioners leverage the INVEST framework as quality guidance when crafting agile product backlog items. Recognizing the absence of formalized evaluation metrics specifically tailored to US-AC and the significant relevance of user story and acceptance criteria introduced in Sect. 2.1, we follow the widely acknowledged agile product backlog item evaluation standards, namely INVEST, for assessment in this work. Here, we integrate the explanation introduced in INVEST⁹ and the measurement details discussed in Buglione and Abran (2013) to extend every attribute for evaluating the US-AC as follows.

- *Independent*: The US-AC should be as independent as possible. It can be implemented and delivered without being dependent on other stories.
- *Negotiable*: The US-AC is not an explicit contract and should leave space for discussion. It can be rewritten depending on business, market, technical, or any other type of requirement by team members.
- *Valuable*: The US-AC must deliver value to the stakeholders by clearly specifying solutions.
- *Estimable*: The US-AC should be estimable. It is well understood by team members enough to be able to determine its relative size (in development and cost).
- *Small*: The US-AC should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
- *Testable*: The US-AC must provide the necessary information to make test development possible.

Halme et al. (2021) operationalized the INVEST as binary evaluation. That is, the evaluators in Halme et al. (2021) consider whether a user story fulfilled the requirements of an INVEST attribute, scoring 0 or 1. Lombriser et al. (2016) employed a five-score scale (1–5) to evaluate the quality of requirements (i.e., user stories and acceptance test) generated from gamification practices. Compared with binary and Likert-scale evaluation mechanisms, the latter method offers a more comprehensive understanding of the contents. Therefore, we adopted Likert-type questionnaires to explore the INVEST mnemonic for the US-AC. Each attribute of INVEST was evaluated using expert opinions on a 5-point Likert scale, where scores of 0–4 represent the scale of *poorly*, *unsatisfactorily*, *reasonably*, *satisfactorily*, and *very satisfactorily*.

For each individual attribute of INVEST, we calculated the mean (m) and standard deviation (sd). The m provides an assessment of the average score derived from

⁹ <https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

the evaluated samples, indicating the central tendency of the measurements. Conversely, the *sd* quantifies the extent of dispersion in the observations, elucidating the divergence of individual data points from the mean value. The cumulative means (Avg.) of all the INVEST attributes are subsequently averaged, collectively offering a comprehensive indicator of the overall satisfaction level on the quality of US-AC.

Relevance To support agile RE effectively, the automatically generated US-AC must achieve a level of relevance that is as good as that achieved by a human practitioner. The primary metrics commonly employed to assess the relevance of generated results are completeness and validity. Given the absence of ground truth in the dataset, employing a gold standard as a benchmark for evaluation is considered appropriate under reasonable conditions (Cardoso et al. 2014). In a study focusing on deriving conceptual UML diagrams from user stories, Bragilovski et al. (2022) evaluated the completeness and validity of the student-generated diagrams against the gold standard crafted by professionals. Similarly, for generating candidate testable references for given requirements, Peng et al. (2021) assessed the precision and recall by examining the automatic abstraction matching with the manually created assumptions. Following these practices, three experienced practitioners were initially invited to craft a gold standard for a given user story within D_i . The practitioners were encouraged to thoroughly examine the project background and the referenced products in order to produce the gold standard as convincingly as possible. Subsequently, the relevance of US-AC generated by SimAC was evaluated through manual comparison with the human-crafted gold standard.

Considering the potentially many-to-many mapping relationship between the generated acceptance criteria items in the gold standard and the SimAC-generated US-AC, we divided the true positives into two categories. True positives in the SimAC solution (TP_s) signify the number of instances present in the US-AC generated by the SimAC that also aligns with the instances in the gold standard. Conversely, false positive (*FP*) denotes the count of instances in the SimAC-generated US-AC that do not correspond to instances within the gold standard. The metric *validity* (*val*) is defined as the ratio of TP_s to the total instances in the LLM-based solution ($TP_s + FP$). In information retrieval terms, validity equates to precision.

$$val = \frac{|TP_s|}{|TP_s| + |FP|} \quad (3)$$

True positives in the gold standard (TP_g) refer to instances present in the gold standard that correspond to instances generated in the SimAC solution. The term false negative (*FN*) denotes the count of instances that exist in the gold standard but cannot be identified in the automatically generated US-AC. The metric *completeness* (*com*) is defined as the ratio between TP_g and the number of instances in the gold standard ($TP_g + FN$). In information retrieval terms, completeness equates to recall.

$$com = \frac{|TP_g|}{|TP_g| + |FN|} \quad (4)$$

Table 3 Description of the practitioners included in this study

Evaluator	Working experience	Agile experience	Role	Engagement in evaluation group
P1	13 years	8 years	QA	Baselines (OpenLLaMA)
P2	15 years	6 years	PM	Role combination
P3	15 years	8 years	PM	Similarity threshold
P4	8 years	5 years	BA	Baselines (OpenLLaMA, GPT-3.5)
P5	6 years	4 years	PO	Role Combination
P6	9 years	5 years	Dev	Baselines (GPT-3.5)
P7	10 years	8 years	BA	K-shot Prompting, Similarity Threshold
P8	10 years	6 years	QA	K-shot Prompting

5.5.2 Methodologies

Practitioner engagement Getting practitioners involved in research is widely acknowledged as a challenging task due to time and other restrictions (Falessi et al. 2018). Zhao et al. (2021) further underscore the issue by revealing that a mere 7.18% of NLP-based RE studies evaluate within an industrial context, highlighting a general lack of industrial evaluation of NLP for RE research results. Agile RE, in particular, heavily relies on domain expertise, necessitating increased practitioner engagement for comprehensive evaluation.

In our study, we actively involved industrial practitioners in various aspects, including scoring the INVEST, crafting the gold standard, and calibrating true positives mapping. In line with the methodology adopted in Ahmed et al. (2023), which involves industrial experts to label the user story, we extended invitations to eight well-trained agile practitioners from different IT software companies to participate in the evaluation process. The backgrounds of these evaluators are presented in Table 3, demonstrating their extensive experience. Each of them has more than four years of experience in agile practices in the development team.

The quality evaluation is divided into five groups of solutions: role combination, similarity threshold, K-shot prompting, GPT-3.5 baselines, and OpenLLaMA baselines. Each evaluation group includes two evaluators from distinct roles to ensure comprehensive evaluations. This approach facilitates the collection of insights from a variety of role perspectives, allowing for an in-depth analysis.

Considering the workload of human evaluation and the time restrictions imposed by practitioner evaluations, the evaluation sample size was intentionally diminished to facilitate a more meticulous and profound analysis. We randomly picked and delivered ten comparative groups of US-AC from each selected project. In total, we delivered 30 sets of US-AC for each evaluation group.

To mitigate bias arising from the distribution of roles within various evaluation groups, it was stipulated that one of the evaluators in each group must be an RA, who is the primary participant in requirement activities. Moreover, two of the authors meticulously reviewed the scores across all evaluation groups. In cases

where groups were either underestimated or overestimated, evaluators were convened for discussion to ensure alignment in estimation levels.

Questionnaire: INVEST scoring The quality evaluation was conducted using a questionnaire method. Prior to the evaluation, evaluators were provided with instructions within the questionnaire package and participated in an online demonstration. Appendix A presents the instructions delivered for evaluation, while Appendix B furnishes a sample of a group of US-AC that needs to be assessed in the questionnaire. In accordance with the protocol, each set of questionnaires was allocated a three-week window for evaluators to score the US-AC with a comprehensive and rigorous analysis.

The sequence in which outputs from different models are presented for review can impact the judgments made by evaluators. To minimize potential bias in the evaluation process, evaluators were directed to conduct a comprehensive review of the entire evaluation corpus to ensure accurate scoring.

To mitigate the inherent subjectivity associated with the evaluations conducted by two raters, an in-depth analysis was undertaken to assess the degree of congruence between their assigned scores. For each metric in INVEST, the difference in scores assigned by the two evaluators was calculated for each instance. A significant deviation exceeding a threshold of 2 points was considered indicative of a substantial inconsistency in their assessments, particularly concerning satisfaction levels. Subsequently, to enhance the reliability of the evaluation, a third expert was enlisted to assess the specific instance independently. The third expert was selected from the pool of experts (i.e., P1 to P8), excluding individuals whose roles were already involved in the current evaluation group. Finally, the two scores demonstrating the least disparity among all three assessments were identified and used to calculate the final average score.

The inter-rater agreement between two evaluators is measured by Cohen's Kappa (Cohen 1960). Following the established categorization scheme proposed by Landis and Koch (1977), the values of the kappa coefficient are categorized into specific ranges, denoting the degree of agreement. Specifically, kappa scores falling within the intervals of 0.21 to 0.40, 0.41 to 0.60, and 0.61 to 0.80 are interpreted as indicative of *fair*, *moderate*, and *substantial* agreement, respectively. For each evaluation group, the level of inter-rater agreement is reviewed. Instances where the agreement falls below the threshold for fair agreement are subjected to discussion, and scores are adjusted accordingly to achieve agreement.

Manual matching: gold standard The gold standard was collaboratively developed by three invited practitioners from different roles (i.e., P7, P2, P8), drawing upon their comprehensive comprehension of the entire project. Leveraging their comprehensive understanding of the entire project, the practitioners engaged in a multi-step process.

Initially, they familiarized themselves with the background of the selected projects by thoroughly analyzing all user stories to form a conceptual understanding of the intended product. Subsequently, they delved into understanding the referenced product of each requirement. For Project NSF, practitioners utilized an available

online repository package¹⁰ to establish a local environment and comprehend the business flow. For Project UniBath, the researchers showcased the data repository business flow of our university to enhance practitioners' understanding. In the case of Project Zooniverse, practitioners registered and interacted with the platform¹¹ to grasp its functional flows. Drawing from this comprehensive understanding, the practitioners collaborated in drafting acceptance criteria. Finally, they engaged in discussions to distill key acceptance criteria gleaned from their background analysis and experience, culminating in the production of a reconciled version, i.e., the gold standard, which served as a benchmark for the relevance evaluation.

In the relevance evaluation process, two of our authors, who are proficient in requirement analysis, compared the auto-generated US-AC with the gold standard. Instances displaying similar meanings to those outlined in the gold standard were categorized as TP_s and TP_g . In cases where instances exhibited ambiguous connotations differing from those in the gold standard, our authors sought suggestions from practitioners to facilitate evaluation and discussion, thereby ensuring informed decisions regarding their classification as true positives or otherwise.

5.5.3 Statistical analysis

To investigate whether the improvement of SimAC against each backbone is statistically significant, we perform the t -test (Sullivan and Feinn 2012) with a confidence level of 95% to make pairwise comparisons in all experiments. If the obtained p -value is less than 0.05, there is a statistically significant difference between the two methods. While a p -value value can inform whether an effect exists, it cannot reveal the size of the effect (Sullivan and Feinn 2012). Subsequently, we further conduct Cohen's d (a standardized mean difference) analysis (Cohen 2013) to measure the effect size, which is measured by the standardized difference between two means. Following the classification in Sullivan and Feinn (2012), the effect size can be classified as *small* ($0.2 \leq d < 0.5$), *medium* ($0.5 \leq d < 0.8$), and *large* ($0.8 \leq d$). According to Cohen, a medium effect indicates the difference is visible to the naked eye of a careful observer, while a small size presents a limited effect (Coe 2002). A large effect size means that the result has meaningful practical significance. The combination of t -test and Cohen's d analysis offers us an in-depth review of the improvement on different evaluation metrics.

6 Experimental results

In this section, we present the experimental results to answer the research questions.

¹⁰ <https://github.com/nsf-open/nsf/tree/master>.

¹¹ <https://blog.zooniverse.org/>.

Table 4 Average satisfaction on INVEST score, comparative results on US-AC generated by different LLMs

LLM	Prompt	I		N		V		E		S		T		Avg.	
		m	sd	m	sd	m	sd	m	sd	m	sd	m	sd		
GPT-3.5	-gpt3.5-turbo	P_{re}	2.87	0.47	2.90	0.44	3.07	0.41	2.97	0.49	2.57	0.50	2.95	0.41	2.89
		P_{ro}	3.10	0.44	3.13	0.50	3.65	0.48	3.40	0.62	2.67	0.54	3.30	0.62	3.21
	text-davinci-003	P_{re}	2.80	0.48	2.63	0.52	2.57	0.56	2.63	0.58	2.77	0.50	2.60	0.53	2.67
OpenLLaMA		P_{ro}	3.00	0.52	2.83	0.49	2.98	0.43	3.08	0.53	3.05	0.57	3.08	0.67	3.01
	OpenLLaMA-7B	P_{re}	0.83	0.67	0.57	0.65	0.62	0.67	0.70	0.70	0.77	0.77	0.65	0.61	0.69
		P_{ro}	1.12	0.76	0.97	0.78	1.07	0.90	1.05	0.85	0.88	0.76	1.10	0.84	1.03
OpenLLaMA-13B		P_{re}	0.82	0.60	0.60	0.62	0.67	0.66	0.78	0.58	0.70	0.59	0.87	0.60	0.74
		P_{ro}	1.15	0.68	1.10	0.75	1.13	0.87	1.15	0.76	1.03	0.74	1.15	0.86	1.12

m - mean, *sd* - standard deviation
P_{re} - regular prompt, *P_{ro}* - role-based prompt
Bold indicates better performance

Table 5 Significance analysis of quality evaluation on different LLMs

p value	P_{re} Versus P_{ro}					
Cohen's d	I	N	V	E	S	T
gpt-3.5-turbo	<i>5.73e-03</i>	<i>7.83e-03</i>	<i>6.82e-11</i>	<i>3.86e-05</i>	<i>2.96e-01</i>	<i>6.63e-04</i>
	+Medium	+Small	+Large	+Medium	+Small	+Medium
text-davinci-003	<i>3.07e-02</i>	<i>3.26e-02</i>	<i>1.32e-05</i>	<i>2.11e-05</i>	<i>4.33e-03</i>	<i>2.52e-02</i>
	+Small	+Small	+Large	+Large	+Medium	+Large
OpenLLaMA-7B	<i>3.22e-02</i>	<i>2.78e-03</i>	<i>2.32e-03</i>	<i>1.52e-02</i>	<i>4.05e-01</i>	<i>1.01e-03</i>
	+Small	+Medium	+Medium	+Small	+Small	+Medium
OpenLLaMA-13B	<i>5.25e-03</i>	<i>1.18e-04</i>	<i>1.23e-04</i>	<i>3.57e-03</i>	<i>7.17e-03</i>	<i>3.81e-02</i>
	+Medium	+Medium	+Medium	+Medium	+Small	+Small

p value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of P_{ro}

6.1 RQ1: generality of SimAC on different LLMs

Quality Table 4 presents the quality evaluation results of SimAC (implemented with P_{ro}) compared to its different underlying LLM baselines (utilized with P_{re}), revealing a notable improvement in satisfaction (s) with the quality of the US-AC generated by SimAC. It is evident that the GPT-3.5 series has already achieved a commendable level of performance, rating as *satisfactorily* ($s > 2$). When collaboratively embedded with the SimAC, these LLMs attain a higher level of satisfaction categorized as *very satisfactorily* ($s > 3$). Particularly, *gpt-3.5-turbo* initialized by P_{ro} consistently performs the best, obtaining the highest average score of 3.21. Notably, such integration leads to a rating of *very satisfactorily* ($s > 3$) across all other INVEST metrics, except for S . In the case of OpenLLaMA family models, the US-AC generated by P_{re} falls into the category of *poorly* ($s < 1$). Consistently, the utilization of P_{ro} does yield some improvement, albeit it remains within the realm of *unsatisfactorily* ($1 < s < 2$).

To examine whether the improvement of SimAC against each backbone LLM is statistically significant, we perform statistical analysis combining t -test and Cohen's d as discussed in Sect. 5.5. The results presented in Table 5 illustrate the improvements are statistically significant and non-trivial across various metrics, specifically in N , V , E , and T . In the case of I , a significant improvement is evident within three out of the evaluated baselines, except for *OpenLLaMA-13B*. Furthermore, the analysis in S reveals a constrained enhancement, as evidenced by p -value and Cohen's d observed in *gpt-3.5-turbo* and *OpenLLaMA-7B*.

Considering the agreement between two evaluators, Table 6 illustrates that all of the assessment outcomes reside within the categories of *fair* and *moderate*. Such agreement adds credibility to the human evaluation results. When comparing the scores from evaluators of different roles, no substantial discrepancy is discerned, with an average difference of less than 0.2 observed for the GPT-3.5

Table 6 Kappa score for assessing inter-rater agreements on the baseline evaluations

kappa (difference)	Pt	I	N	V	E	S	T	Avg.	Agreement
gpt-3.5-turbo	P_{re}	0.65 (-0.13)	0.22 (-0.20)	0.34 (-0.20)	0.33 (-0.27)	0.47 (-0.13)	0.37 (-0.23)	0.40 (-0.19)	Moderate
	P_{ro}	0.41 (-0.13)	0.37 (-0.13)	0.35 (-0.10)	0.25 (-0.27)	0.22 (0.20)	0.53 (0.00)	0.35 (-0.07)	Fair
text-davinci-003	P_{re}	0.51 (0.00)	0.38 (0.33)	0.41 (0.27)	0.26 (0.07)	0.40 (-0.07)	0.61 (0.13)	0.43 (0.12)	Moderate
	P_{ro}	0.69 (0.07)	0.36 (0.13)	0.48 (0.10)	0.48 (0.10)	0.47 (0.10)	0.61 (0.10)	0.51 (0.10)	Moderate
OpenLLaMA-7B	P_{re}	0.33 (-0.13)	0.41 (-0.07)	0.49 (0.10)	0.78 (0.00)	0.27 (-0.33)	0.27 (-0.17)	0.42 (-0.04)	Moderate
	P_{ro}	0.50 (0.03)	0.29 (-0.13)	0.53 (0.07)	0.46 (-0.03)	0.43 (-0.17)	0.42 (0.00)	0.44 (-0.04)	Moderate
OpenLLaMA-13B	P_{re}	0.56 (-0.03)	0.29 (0.00)	0.20 (0.07)	0.31 (-0.17)	0.27 (-0.20)	0.36 (0.00)	0.33 (-0.06)	Fair
	P_{ro}	0.29 (0.03)	0.38 (0.13)	0.62 (0.13)	0.44 (-0.10)	0.27 (0.13)	0.27 (0.03)	0.38 (0.02)	Fair

Pt: Prompt

Negative sign (–) indicates higher score received from RA (vs. QA and Dev)

Table 7 Completeness and validity of US-AC generated by different LLMs

(%)	Pt	NSF		UniBath		Zooniverse		Avg.	
		com	val	com	val	com	val	com	val
gpt-3.5-turbo	P_{re}	49.69	42.64	48.26	39.92	42.44	50.58	46.80	44.38
	P_{ro}	60.53	54.93	64.92	48.64	56.33	50.28	60.60	↑29.48
text-davinci-003	P_{re}	31.97	39.96	33.78	44.58	32.25	51.86	32.67	45.47
	P_{ro}	38.25	41.15	38.70	45.45	34.60	50.52	37.18	↑13.83
OpenLLaMA-7B	P_{re}	16.28	48.50	8.97	30.83	9.27	45.83	11.50	41.72
	P_{ro}	21.00	51.31	8.97	30.50	12.54	46.83	14.17	↑23.16
OpenLLaMA-13B	P_{re}	15.31	40.28	14.19	40.83	9.43	43.19	12.97	41.43
	P_{ro}	20.44	34.70	19.50	28.89	11.19	47.00	17.05	↑31.41

Pt: Prompt

Bold indicates better completeness

series. The discrepancies are even smaller when assessing the OpenLLaMA series, with an average difference of less than 0.06.

Relevance Table 7 further proves the superior performance of SimAC in enhancing LLMs' GAC capability in terms of completeness and validity. The results underscore a substantial improvement in relevance attributed to the

Table 8 Significance analysis of relevance evaluation for different LLMs

<i>p</i> -value (Cohen's <i>d</i>)	<i>P_{re}</i> Versus <i>P_{ro}</i>	
	com	val
<i>gpt-3.5-turbo</i>	<i>4.49e-04</i> (+Large)	<i>6.19e-02</i> (+Small)
<i>text-davinci-003</i>	<i>1.94e-01</i> (+Small)	<i>9.64e-01</i> (+Small)
<i>OpenLLaMA-7B</i>	<i>2.65e-01</i> (+Small)	<i>8.99e-01</i> (+Small)
<i>OpenLLaMA-13B</i>	<i>2.12e-01</i> (+Small)	<i>5.86e-01</i> (-Small)

p-value less than 0.05 is highlighted with italics
 Bold indicates large effect size
 Positive sign (+) indicates better performance in setting of *P_{ro}*

Table 9 Average satisfaction on INVEST score, evaluating by combinations of different role groups

<i>R</i>	<i>I</i>		<i>N</i>		<i>V</i>		<i>E</i>		<i>S</i>		<i>T</i>		Avg.
	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	
1	2.78	0.58	2.63	0.55	2.87	0.65	2.87	0.60	2.92	0.42	2.90	0.60	2.83
2	2.97	0.52	2.92	0.42	3.08	0.67	3.07	0.55	2.70	0.50	3.27	0.63	3.00
3	3.03	0.55	3.12	0.52	3.60	0.59	3.08	0.53	2.60	0.49	3.45	0.62	3.15

m - mean, *sd* - standard deviation

R - Groups of roles, Avg. - Average score over INVEST attributes

Bold indicates highest score

US-AC generated by SimAC. Particularly, SimAC embedding with *gpt-3.5-turbo* consistently performs the best, receiving 60.60% completeness and 51.29% validity, marking enhancements of 29.48% and 15.56%, respectively. SimAC slightly enhances *text-davinci-003* in GAC, showing an improvement of 13.83% in completeness and 0.53% in validity.

Despite the underperformance of the OpenLLaMA series, receiving less than 20% in completeness, SimAC still contributes to improving their completeness with improvements of 23.16% for *OpenLLaMA-7B* and 31.41% for *OpenLLaMA-13B*. However, embedding SimAC into the OpenLLaMA series leads to a decrease in validity, suggesting that SimAC might introduce some irrelevant information into the output.

The results in Table 8 reveal a statistically significant improvement in completeness upon integrating SimAC into *gpt-3.5-turbo*. However, the enhancement of completeness becomes less evident when introducing SimAC into other LLMs. Meanwhile, the enhancement in validity across all LLMs is not readily discernible (i.e., with *small* effect size).

Answering RQ1: In summary, SimAC delivers substantial improvements for different LLMs of diverse parameter sizes in terms of both quality and relevance, demonstrating a robust generality across various LLMs. Specifically, the collaboration facilitated by role-based prompts yields high satisfactory quality and

Table 10 Significance analysis of INVEST score for different settings on combinations of role groups

<i>p</i> -value	<i>R</i> = 3					
<i>Cohen's d</i>	I	N	V	E	S	T
<i>R</i> = 1	<i>1.75e-02</i>	<i>2.79e-06</i>	<i>2.21e-09</i>	<i>3.74e-02</i>	<i>2.57e-04</i>	<i>2.84e-06</i>
	+Small	+Large	+Large	+Small	+Medium	+Large
<i>R</i> = 2	<i>4.97e-01</i>	<i>2.32e-02</i>	<i>1.70e-05</i>	<i>8.66e-01</i>	<i>2.71e-01</i>	<i>1.13e-01</i>
	+Small	+Small	+Large	+Small	+Small	+Small

p-value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of *R* = 3

Negative sign (−) indicates worse performance in setting of *R* = 3

Table 11 Kappa score and differences for assessing inter-rater agreements on role group combination evaluations

kappa (difference)	I	N	V	E	S	T	Avg.	Agreement
<i>R</i> = 1	0.68 (−0.10)	0.74 (−0.07)	0.54 (−0.13)	0.62 (−0.13)	0.28 (0.10)	0.75 (−0.07)	0.60 (−0.07)	Substantial
<i>R</i> = 2	0.38 (0.00)	0.26 (−0.03)	0.61 (−0.10)	0.43 (−0.07)	0.71 (0.07)	0.65 (−0.07)	0.51 (−0.04)	Moderate
<i>R</i> = 3	0.43 (0.00)	0.19 (0.03)	0.31 (−0.07)	0.48 (−0.17)	0.72 (0.07)	0.58 (−0.17)	0.45 (−0.05)	Moderate

Negative sign (−) indicates higher score received from RA (vs. PM)

achieves completeness rates exceeding 60% when applied to LLMs trained with extensive parameter sizes, exemplified by the GPT-3.5 series. Moreover, this approach enhances the output quality in GAC even when embedded with LLMs trained with relatively smaller parameter sizes, such as those within the OpenL-LaMA model family.

6.2 RQ2: Ablation study for role groups in SimAC

Quality Table 9 demonstrates satisfaction scores obtained from introducing various role groups to SimAC. The results notably indicate that SimAC, incorporating three distinct role groups, attains the highest satisfaction score of 3.15 on average. However, the inclusion of QA and Others results in the generation of more informative acceptance criteria, making the size of US-AC slightly exceed the scope compatible with sprint implementation. Therefore, the highest score of *S* is observed in scenarios where only RA is introduced. For the other metrics, the introduction of additional roles yields consistent enhancement. Notably, when

Table 12 Completeness and validity in combinations of different role groups

R (%)	NSF		UniBath		Zooniverse		Avg.	
	com	val	com	val	com	val	com	val
R = 1	38.42	39.96	40.94	46.24	36.39	44.21	38.58	43.47
R = 2	38.64	38.52	45.75	41.19	43.82	44.05	42.74	41.25
R = 3	60.53	54.93	64.92	48.64	56.33	50.28	60.60	51.29

Bold indicates highest completeness

Table 13 Significance analysis of completeness and validity for different settings on combinations of role groups

p-value (Cohen's d)	R = 3			
	com		val	
R = 1	<i>2.16e-07</i>	(+Large)	<i>4.11e-02</i>	(+Medium)
R = 2	<i>5.03e-05</i>	(+Large)	<i>3.41e-03</i>	(+Medium)

p-value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of R = 3

Table 14 Average satisfaction on INVEST score, evaluating by K-shot prompting

K	I		N		V		E		S		T		Avg.
	m	sd	m	sd	m	sd	m	sd	m	sd	m	sd	
0	2.72	0.64	2.43	0.72	2.78	0.72	3.15	0.66	2.15	0.73	3.05	0.75	2.71
1	2.95	0.34	3.02	0.60	3.12	0.52	3.27	0.63	2.67	0.57	3.15	0.55	3.03
3	3.57	0.56	3.63	0.49	3.58	0.59	3.37	0.64	3.23	0.59	3.47	0.50	3.48
4	3.60	0.53	3.48	0.50	3.48	0.60	3.25	0.57	3.08	0.67	3.33	0.54	3.37

m - mean, sd - standard deviation

K - Number of shots, Avg. - Average score over INVEST attributes

Bold indicates highest score

three distinct role categories are introduced, the resulting outcomes reach a rating of *very satisfactorily* ($s > 3$).

From a statistical perspective, the US-AC generated by all three role groups ($R = 3$) proves significantly enhanced compared with the inclusion of RA only ($R = 1$), as demonstrated in Table 10. The involvement of QA and Others significantly enriches the comprehensiveness of requirement details and introduces additional valuable items (V) to US-AC, indicated by *large* effect size of enhancement observed. It is noteworthy that the augmentation of N and T attain significant enhancement with three role groups incorporated.

Cohen's kappa scores for this set of evaluations are presented in Table 11, which are situated within the 0.4–0.6, indicative of *moderate* to *substantial* agreement.

Table 15 Significance analysis of INVEST score for different settings on K-shot prompting

<i>p</i> -value	<i>K</i> = 3					
<i>Cohen's d</i>	I	N	V	E	S	T
<i>K</i> = 0	4.17e-12	4.78e-19	8.27e-10	6.97e-02	7.67e-15	4.89e-04
	+Large	+Large	+Large	+Small	+Large	+Medium
<i>K</i> = 1	4.61e-11	8.22e-09	1.17e-05	3.91e-01	4.86e-07	1.28e-03
	+Large	+Large	+Large	+Small	+Large	+Medium
<i>K</i> = 4	7.39e-01	9.96e-02	3.58e-01	2.93e-01	1.97e-01	1.65e-01
	-Small	+Small	+Small	+Small	+Small	+Small

p-value less than 0.05 is highlighted with itaics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of *K* = 3

Negative sign (−) indicates worse performance in setting of *K* = 3

Reviewing the difference between the two evaluators, it is evident that the RA evaluator consistently assigns slightly higher scores, with the biggest discrepancy of 0.17 compared to the PM evaluator.

Relevance Table 12 illustrates the impact of introducing different role groups into the SimAC on relevance. The findings reveal enhancements in both completeness and validity as more roles are integrated. Evidently, the inclusion of three groups of roles results in the highest level of completeness. Conversely, the incorporation of only the RA role or the combination of RA and QA leads to completeness and validity falling below 50%. Upon reviewing by project, the improvement remains consistent across all three projects.

The statistical analysis presented in Table 13 indicates the enhancement for both completeness and validity is significant and non-negligible (i.e., *p*-value less than 0.05 with a *large* or *medium* effect size) upon introducing more roles.

Answering RQ2: The most effective practice to approach SimAC is to engage the collaboration involving three distinct groups of roles: requirement analyst, quality analyst, and other team members.

6.3 RQ3: Analysis of different setting on K-shot prompting

Quality The efficacy of incorporating different examples in the prompt is presented in Table 14. The results demonstrate that, with a setting of *K* = 3, the INVEST score reaches its zenith. The human evaluation consistently maintains *very satisfactorily* (*s* > 3) across all INVEST metrics, both with *K* = 3 and *K* = 4. Concerning the increment of *K*, an overall enhanced level of satisfaction is observed when an increased number of scenarios are encompassed within the prompt examples. Specifically, the setting of *K* = 3 accommodates more scenarios involving different engagement levels, where participants acting in diverse roles can thereby contribute outputs that leverage their respective expertise. Consequently, these detailed

Table 16 Kappa score and differences for assessing inter-rater agreements on k-shot settings

kappa (difference)	I	N	V	E	S	T	Avg.	Agreement
$K = 1$	0.25 (0.43)	0.47 (0.33)	0.28 (−0.03)	0.31 (0.43)	0.33 (0.30)	0.34 (0.30)	0.33 (0.29)	Fair
$K = 2$	0.22 (0.10)	0.22 (0.37)	0.33 (−0.17)	0.32 (0.20)	0.25 (0.07)	0.23 (−0.17)	0.26 (0.07)	Fair
$K = 3$	0.29 (0.33)	0.30 (0.13)	0.40 (−0.03)	0.30 (0.00)	0.27 (0.20)	0.46 (0.00)	0.34 (0.11)	Fair
$K = 4$	0.21 (0.27)	0.27 (0.43)	0.32 (−0.03)	0.43 (−0.17)	0.28 (0.10)	0.35 (0.00)	0.31 (0.10)	Fair

Negative sign (−) indicates higher score received from RA (vs. QA)

Table 17 Completeness and validity in different settings on K-shot prompting

(%)	NSF		UniBath		Zooniverse		Avg.	
	com	val	com	val	com	val	com	val
$K = 1$	34.94	38.24	43.01	28.13	42.69	30.40	40.21	32.25
$K = 2$	35.47	40.76	45.33	37.00	47.57	48.73	42.79	42.16
$K = 3$	60.53	54.93	64.92	48.64	56.33	50.28	60.60	51.29
$K = 4$	43.11	39.27	52.37	43.01	45.03	46.03	46.84	42.77

Bold indicates highest completeness

acceptance criteria encompass a broader spectrum of negotiable (*N*), valuable (*V*), and testable (*T*) components. In terms of *I*, the setting of $K = 4$ receives a higher satisfaction score since it benefits from a greater number of examples to generate fewer dependency-related elements.

Compared with zero-shot ($K = 0$) and one-shot ($K = 1$) prompting, the findings in Table 15 suggest that implementing the few-shot prompting strategy yields significant improvement in other metrics except for *E*. The evaluators perceive that the acceptance criteria generated by *gpt-3.5-turbo* maintain a consistently high level of satisfaction ($s > 3$) in representing estimable items. Thereby, no significant improvement is observed. Transitioning from $K = 3$ to $K = 4$ does not result in statistically significant differences across all INVEST metrics.

Cohen's kappa scores for this particular set of evaluations, as depicted in Table 16, fall within the range of 0.2–0.4, suggesting a *fair* level of agreement. Notably, in this evaluation, the RA consistently assigns tighter scores across various metrics, except for *V*. Conversely, when evaluating the value of US-AC, the QA evaluator tends to assign lower scores. One potential explanation for this difference is their respective focus on different aspects of the US-AC.

Relevance Reviewing Table 17, the highest completeness is observed for $K = 3$, which remains consistent across all three projects. Remarkably, the completeness exceeds 50% for each project individually, implying that the US-AC

Table 18 Significance analysis of completeness and validity for different settings on K-shot prompting

<i>p</i> -value (Cohen's <i>d</i>)	<i>K</i> = 3			
	com		val	
<i>K</i> = 1	<i>1.45e-05</i>	(+Large)	<i>1.92e-05</i>	(+Large)
<i>K</i> = 2	<i>7.61e-06</i>	(+Large)	<i>2.01e-02</i>	(+Medium)
<i>K</i> = 4	<i>4.64e-04</i>	(+Large)	<i>9.27e-03</i>	(+Medium)

p-value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of *K* = 3

Table 19 Average satisfaction on INVEST score, evaluating by semantic similarity thresholds

<i>t</i>	I		N		V		E		S		T		Avg.
	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	<i>m</i>	<i>sd</i>	
0.95	2.75	0.65	2.35	0.61	3.53	0.50	3.50	0.50	2.48	0.68	3.47	0.54	3.01
0.90	2.12	0.64	2.12	0.74	2.15	0.86	2.27	0.82	2.37	0.64	2.15	0.73	2.19
0.85	1.63	0.80	1.58	0.98	1.37	0.86	1.22	0.85	1.70	0.98	1.17	0.96	1.44

m - mean, *sd* - standard deviation

t - Threshold, Avg. - Average score over INVEST attributes

Bold indicates highest score

Table 20 Significance analysis of INVEST score for different settings on similarity threshold

<i>p</i> -value	<i>t</i> = 0.95					
Cohen's <i>d</i>	I	N	V	E	S	T
<i>t</i> = 0.90	<i>4.20e-7</i>	<i>4.16e-02</i>	<i>3.25e-19</i>	<i>3.13e-17</i>	<i>3.33e-01</i>	<i>2.27e-20</i>
	+Large	+Large	+Large	+Large	+Small	+Large
<i>t</i> = 0.85	<i>1.44e-13</i>	<i>4.82e-07</i>	<i>4.36e-33</i>	<i>1.46e-35</i>	<i>1.32e-06</i>	<i>8.43e-32</i>
	+Large	+Large	+Large	+Large	+Large	+Large

p-value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of *t* = 0.95

generated by SimAC, prompted with three sets of examples, possesses the capability to match half of those written by humans. There is a general trend that a higher number of examples leads to better completeness. However, for *K* = 4, its coverage scenario is narrower than that of *K* = 3, resulting in lower completeness.

Regarding validity, there is no consistent trend indicating that an increasing number of examples yields higher validity. However, the setting of *K* = 3 yields the best validity. One assumption is that the introduction of more scenarios covering

Table 21 Kappa score and differences for assessing inter-rater agreements on similarity threshold

kappa (difference)	I	N	V	E	S	T	Avg.	Agreement
$t = 0.95$	0.49 (0.17)	0.88 (0.07)	0.21 (-0.13)	0.23 (-0.20)	0.69 (-0.10)	0.25 (-0.13)	0.46 (-0.05)	Moderate
$t = 0.90$	0.26 (-0.17)	0.64 (-0.03)	0.45 (-0.17)	0.60 (-0.20)	0.31 (0.07)	0.22 (0.03)	0.41 (-0.07)	Moderate
$t = 0.85$	0.23 (-0.13)	0.49 (0.17)	0.51 (-0.07)	0.65 (-0.03)	0.42 (0.07)	0.62 (0.07)	0.49 (0.01)	Moderate

Negative sign (-) indicates higher score received from RA (vs. PM)

Table 22 Completeness and validity in different settings on similarity threshold

kappa (%)	NSF		UniBath		Zooniverse		Avg.	
	com	val	com	val	com	val	com	val
$t = 0.95$	60.53	54.93	64.92	48.64	56.33	50.28	60.60	51.29
$t = 0.90$	32.00	65.12	32.42	57.67	28.07	56.02	30.83	59.60
$t = 0.85$	17.39	59.17	21.42	73.33	16.53	51.67	18.45	61.39

Bold indicates highest completeness

Table 23 Significance analysis of completeness and validity for different settings on similarity threshold

p -value (Cohen's d)	$t = 0.95$	
	com	val
$t = 0.90$	<i>2.03e-18</i>	(+Large)
$t = 0.85$	<i>1.69e-11</i>	(+Large)

p -value less than 0.05 is highlighted with italics

Bold indicates large effect size

Positive sign (+) indicates better performance in setting of $t = 0.95$

different proportions of role group engagement introduces more relevant information, thereby reducing the false positives generated.

Statistical analysis in Table 18 indicates significant improvements in both completeness and validity upon incorporating examples covering three engagement scenarios.

Answering RQ3: Leveraging few-shot prompting, which covers three typical collaboration scenarios, SimAC gains higher satisfaction levels on output quality. The resulting US-AC demonstrates greater alignment with the human-crafted gold standard, rendering them more pertinent and applicable.

6.4 RQ4: Analysis of different settings on similarity thresholds

Quality Table 19 presents the effectiveness of employing varying similarity thresholds. Obviously, when t is set as 0.95, the highest average INVEST score is achieved. The increment of the threshold demonstrates a consistent improvement in INVEST.

Reviewing from Table 20, it is evident the utilization of higher t preserves more significant valuable information within acceptance criteria. The comprehensive elaborations of details enhance the understandability and testability of the user story, consequently yielding the highest score in E and T . However, the intricate elaborations have the consequence of reducing the size of the user story, potentially extending it beyond the scope of a single sprint. Consequently, in terms of S , the influence of varying t from 0.9 to 0.95 exhibits a relatively less pronounced effect.

The Cohen's kappa scores (Table 21) for this set of evaluations range between 0.4 and 0.5, reflecting a *moderate* level of agreement. When comparing the scoring for two evaluators, no significant discrepancies are evident, with the largest difference being less than 0.2.

Relevance The results presented in Table 22 indicate that decreasing the similarity threshold results in lower completeness. A decrease in the threshold entails fewer acceptance criteria being included, thereby reducing the amount of information available to match items in the gold standard.

Conversely, a decrease in the threshold leads to higher validity in some cases. With a lower threshold, SimAC yields a smaller total instance, consequently resulting in higher validity.

Statistical analysis (Table 23) reveals a significant improvement in completeness. However, the improvement in validity is not as pronounced, with only a *small* effect size observed.

Answering RQ4: It is evident that redundant information is generated by different roles in the process of GAC within SimAC. The subsequent deduplication process can streamline the refined US-AC to a size that fits in appropriate sprint planning and discussion. Importantly, the careful selection of an optimal similarity score threshold can significantly impact the relevance and quality of the US-AC.

7 Case studies

This section presents three case studies exemplifying outputs produced by SimAC, which integrate various LLMs for qualitative analysis. Cases 1 and 2 depict the successful scenarios of SimAC employed in GPT-3.5 and OpenLLaMA series, alongside offering a detailed breakdown of role-related output in Case 1. Case 3 discusses a failed scenario that SimAC employed in LLMs with fewer parameters. Such an approach enables a thorough exploration of the strengths and limitations inherent in SimAC. To ensure a comprehensive analysis infused with industrial perspectives,

our researchers collaborated closely with practitioners to scrutinize the intricacies presented within these specific cases.

7.1 Case 1: SimAC implemented on GPT-3.5

For the LLMs trained with massive parameter sizes, the LLM initiated with regular prompts has already received highly satisfactory results. The incorporation of role-based prompts in SimAC can elaborate the US-AC with role-related details. Case 1 is a typical example, as shown in Fig. 9. In Case 1, the goal specified in the user story is to search and discover music or ringback tones based on content similarity.

7.1.1 Introduction on informative details

In Fig. 9, the middle-left panel presents the acceptance criteria generated by SimAC incorporating with the regular prompt, while the middle-right panel presents the output generated based on LLM utilizing the role-based prompt. The bottom panel showcases the gold standard meticulously crafted by practitioners. Within the middle panels, the underlined items correspond to the acceptance criteria aligned with the gold standard, while the points highlighted in red denote aspects deemed valuable by practitioners for inclusion in the discourse. In the bottom panels, underlined items denote those matched by LLM-generated solutions, with the item highlighted in blue, indicating its exclusive matching by the SimAC solution.

As can be seen, Point 2 on both solutions defines the criteria for the system to display the results based on the content similarity. However, the SimAC solution further extends its elaboration to define the facets and considerations for content similarity. Moreover, compared with Point 7 in the regular LLM solution, Point 10 in the SimAC solution elucidates the scenario (e.g., a large volume of contents) necessitating consideration during performance handling. SimAC enables the LLM to delineate the consideration factors, which serves as a valuable point of reference. The RA experts endorse the inclusion of such supplemental content, which aids in formulating the comprehensive requirements and resulting in a highly satisfactory degree in terms of E .

Furthermore, in the SimAC solution, Point 9 updates the acceptance criteria by introducing considerations on software compatibility in different platforms, which provides an alignment for development. Points 8 and 11 furnish detailed considerations regarding functionalities warranting discussion or clarification with stakeholders, thereby serving as valuable reference points. Therefore, these supplements contribute to the high score received in V .

In terms of relevance, both approaches receive a TP_s value of 7, as underlined in the middle panels presented in Fig. 9. However, SimAC generates additional acceptance criteria, which slightly reduces the validity of the case. The TP_g associated with the criteria generated from regular LLM and that produced by the SimAC are 7 and 8, respectively, as underlined in the gold standard panel. In the SimA solution, Point 6 integrates the filter and sort functionalities, whereas they are separately addressed

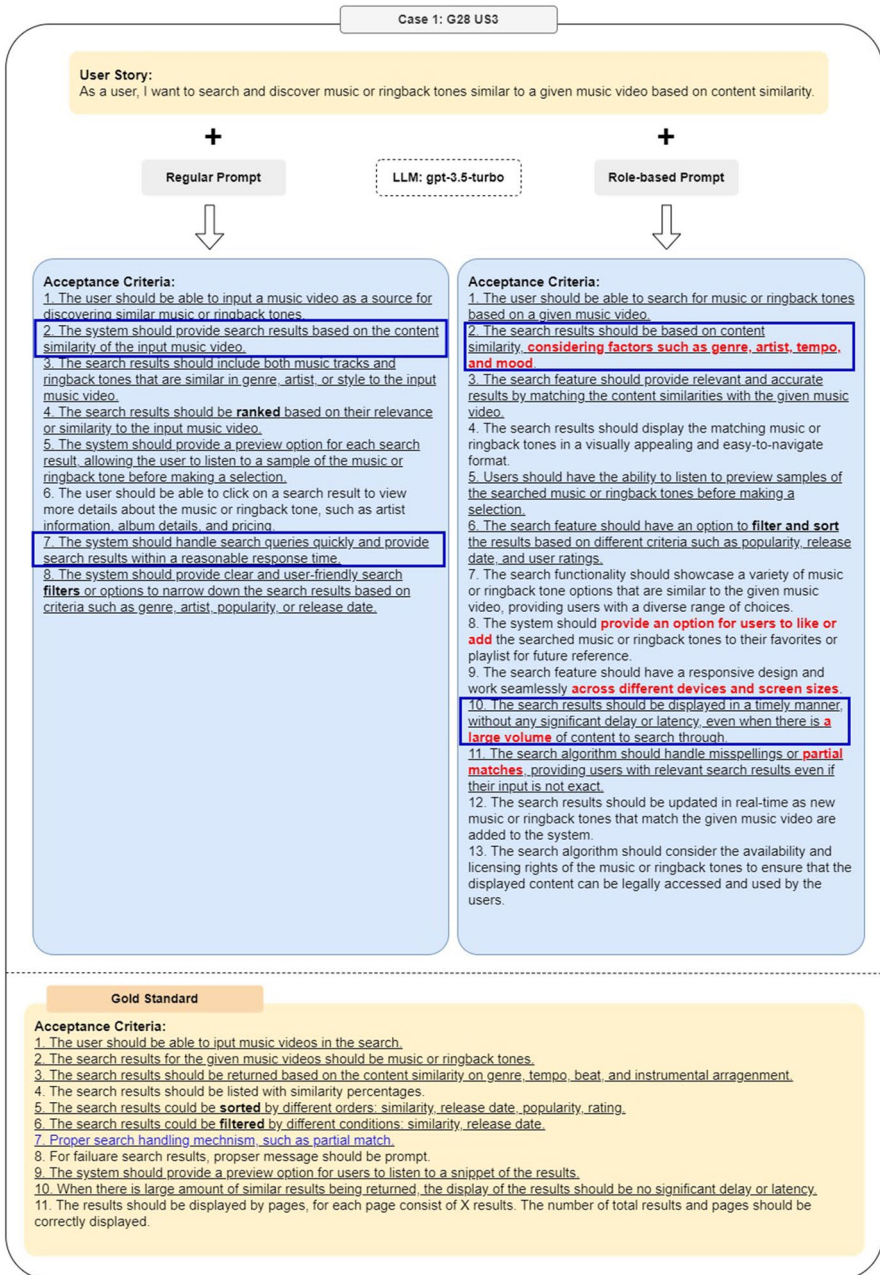


Fig. 9 Case 1: Generate acceptance criteria with the regular and role-based prompt. (G28 US3)

in Points 4 and 8 in the regular LLM solution. Furthermore, the SimAC solution features one additional true positive against the gold standard (i.e., Point 7 in the gold standard panel), as highlighted in Point 11. The support for the partial match mechanism is exclusively mentioned in the SimAC solution, contributing to a TP_g of 8 with higher completeness for this case.

7.1.2 Supplements on role-related insights

Figure 10 illustrates the detailed acceptance criteria generated or updated by different roles in Case 1. The acceptance criteria updated by the QA role place a particular emphasis on the performance issue and error handling within the search function (as presented in the middle panel). As underlined in Points 10 and 11, these criteria align with the human-crafted gold standard, thus augmenting the completeness. Incrementally, the LLM acts in other roles to address the legacy issue. In this context, the additional considerations primarily stem from the perspective of a project manager. As highlighted in red, these additional considerations serve as valuable points of reference for stakeholder deliberations, even their absence in the original gold standard. It is noteworthy that SimAC utilizes role-playing techniques, facilitating the generation of detailed insights from role-specific perspectives, thereby introducing valuable points for reference and further discussion.

Finding 1: In contrast to initiating LLMs with regular prompts, SimAC incorporates collaborative procedures into the LLM, thereby introducing more informative acceptance criteria and augmenting insights from role-specific perspectives. These enriched criteria and role-specific enhancements are considered valuable references for practitioners.

7.2 Case 2: SimAC implemented on OpenLLaMA

SimAC boosts the efficacy of GAC, particularly for LLMs with smaller sizes. Figure 11 showcases the acceptance criteria generated by *OpenLLaMA-7B* with the regular prompt and the role-based prompt in the middle-left and middle-right panels, respectively. The gold standard for this case is presented in the bottom panel. In Case 2, the goal specified in the user story is to allow the user to deposit arbitrarily large files into the system.

In Fig. 11, the acceptance criteria generated by the regular LLM solution elaborate the user story with limited information. The output provides insights into deposit files of any format, as repeatedly presented in Points 2 and 4. However, it erroneously specifies file limitations (highlighted in blue on the middle-left panel) despite the user story suggesting the ability to deposit arbitrarily large files. The integration of role-based prompts in SimAC, exemplified in the middle-right panel, rectifies the mistake and extends the depth of elaboration. The output supplements the narrative with specifics regarding file storage location and acceptable compressed file formats. QA experts consider the keywords highlighted in red as more informative criteria, which offer valuable references for designing test strategies.

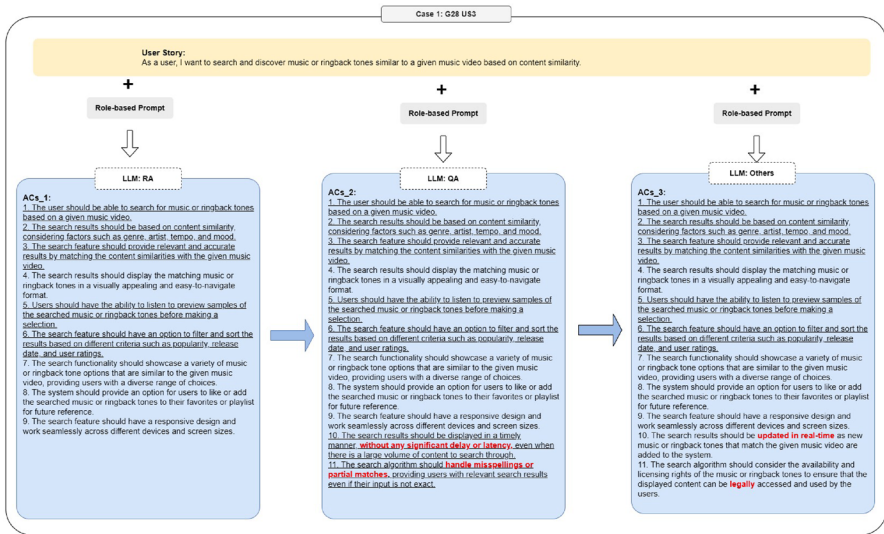


Fig. 10 Case 1: Generate acceptance criteria from different roles. (G28 US3)

Considering the relevance, the TP_g remains at 1 (i.e., Point 2 underlined in the bottom panel) for the regular LLM solution, as underlined, where Points 2 and 4 in the middle-left panel convey similar information. In contrast, TP_g stands as 2 in the SimAC solution, reflecting SimAC's capacity to generate more precise and informative criteria, as highlighted in blue at the bottom.

Finding 2: SimAC enhances the capacity of LLMs trained with constrained parameters to generate more correct and informative details in GAC. Additionally, SimAC slightly improves the completeness of the output. Nonetheless, these improvements fall short of reaching a satisfactory level.

7.3 Case 3: failures of SimAC

For some of the user stories, such as Case 3 presented in Fig. 12, it is evident that LLMs trained with more limited parameters do not exhibit any improvement even when employing SimAC. The goal specified in Case 3 is to integrate the user-centered design into the recommendation.

In Case 3, no additional information is generated when introducing the role-based prompt on *OpenLLaMA-7B*. One probable explanation is that the LLM lacks project-specific knowledge, thereby impeding its ability to elaborate on particular scenarios. In this case, the given user story includes vague descriptions, such as “our recommendation”. This kind of description does not furnish sufficient contextual information for LLMs to understand what type of recommendation should be integrated with a user-centered design. Even utilizing SimAC, the

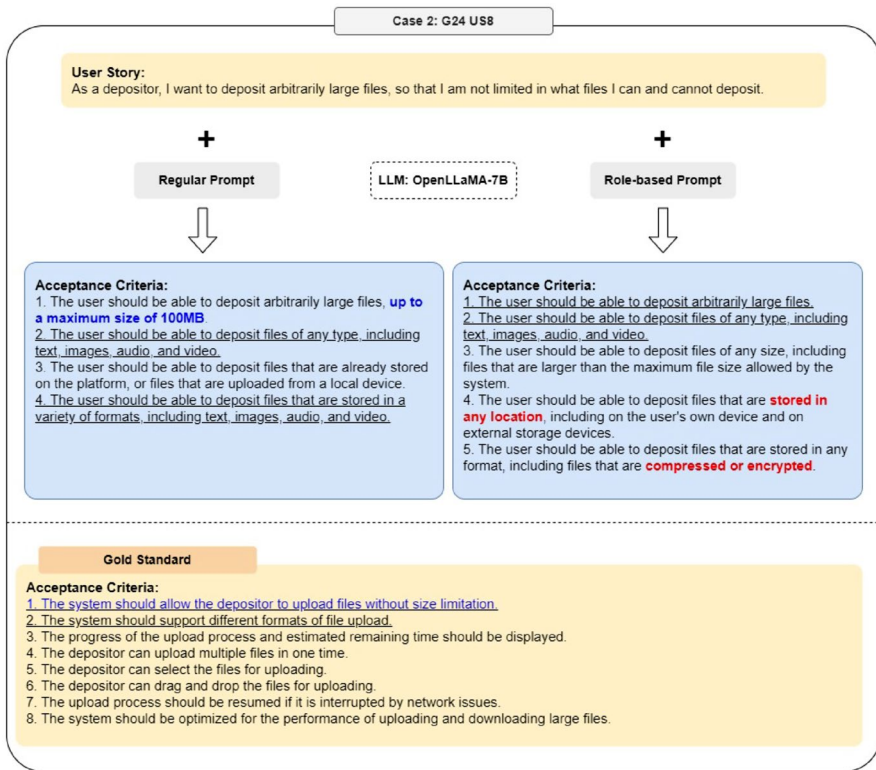


Fig. 11 Case 2: Generate acceptance criteria with the regular and role-based prompt. (G24 US8)

LLMs remain unable to extract additional relevant information from their knowledge base.

This limitation persists even when employing LLM trained with larger parameters, such as *OpenLLaMA-13B*. No supplementary acceptance criteria are generated, as presented in the top-right portion in Fig. 12. Although the incorporation of SimAC into *OpenLLaMA-13B* yields a more informative outcome, no valuable or pertinent information emerges. Among the above four solutions generated in this case, only one basic criterion mentions platform access rights, which is in line with Point 2 in the gold standard. This particular scenario underscores the diminished effectiveness of SimAC when it is introduced to LLMs with less proficiency in understanding the context of the user story.

When embedding SimAC with *text-davinci-003*, the resulting US-AC for this case is more reasonable. In the solution generated by the regular prompt, the US-AC places emphasis on the report, a component not explicitly mentioned in the original user story. However, with the introduction of the role-based prompt, the focus of the US-AC shifts towards a demonstration of user-centered design for discussion, a theme more closely aligned with the gold standard. As highlighted in red, experts

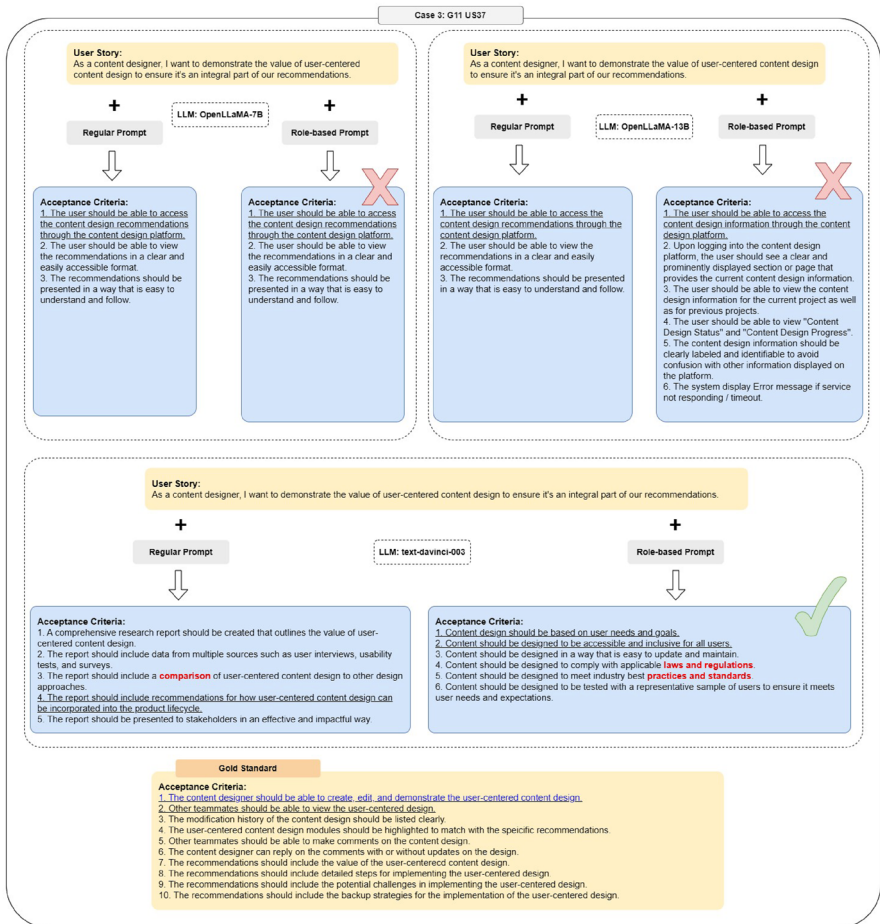


Fig. 12 Case 3: Generate acceptance criteria with the regular and role-based prompt in different parameter sizes (G11 US37)

suggest that these points also offer valuable references for further discussion during sprint planning. Additionally, the SimAC solution employed in *text-davinci-003* fulfills an additional gold standard item, as indicated by Point 1 highlighted in blue at the bottom, contributing to higher completeness with a larger TP_g .

Finding 3: In scenarios where the provided user story offers an indistinct depiction of the project context, the effectiveness of SimAC is diminished when implemented on LLMs trained with fewer parameters.

8 Threats to validity

Our results need to be considered in view of threats to validity. We follow (Wohlin et al. 2012) to discuss the *internal*, *external*, *conclusion*, and *construct* validity of our study.

8.1 Internal validity

Internal validity concerns the external factors that might affect dependent variables (Wohlin et al. 2012), which may be due to individual factors in the proposed method. In the proposed SimAC, a fixed set of examples is utilized, while different sets of examples may yield diverse results. To mitigate the impact of these threats, we purposefully crafted some examples for common projects demonstrated in online resources rather than relying solely on the dataset D itself. This strategy minimizes the overlap of project contexts between prompting examples and the testing dataset, thereby enhancing the resilience of our proposed framework.

8.2 External validity

External validity addresses the generalizability of the results (Wohlin et al. 2012). The absence of first-hand information about these projects leads to an inevitable deviation between the gold standard crafted by our invited practitioners and the ground truths gained in real-world applications. This gap is more evident in agile practices, which necessitates more insights from the project environment (Conboy and Fitzgerald 2004). To mitigate this threat, during the evaluation process and drafting of the gold standard, we involved practitioners to familiarize themselves with the project background. Furthermore, we provided a three-week window for evaluators to review results. To gain more convincing results, one of the future research directions is to collaborate with industry projects and apply our approach in a real-world context to obtain first-hand results.

Another threat arises from the limited experimented models in this work. As mentioned in Sect. 2.2.2, encoder-only models are only suitable for text understanding without text generation capability, while most encoder-decoder models are in relatively small parameter sizes, such as CodeT5 (Wang et al. 2021) and UniXcoder (Guo et al. 2022), leading their poor performance in training-free task adaptation. As such, we focus on prevalent decoder-only models, including a series of GPT-3.5 and OpenLLaMA, where the former is closed-source LLMs while the latter is open-source. Particularly, we select *OpenLLaMA-7B* and *OpenLLaMA-13B* to make comparisons among LLMs of the same family but different sizes. Besides, we select *gpt-3.5-turbo* and *text-davinci-003* to make comparisons among LLMs of similar parameter sizes but diverse pre-training foundations. Hence, diverse LLMs are tested in the work, and we believe such an approach can minimize the identified threat.

8.3 Conclusion validity

Conclusion validity concerns the relationship between the proposed method and the outcome (Wohlin et al. 2012). We followed the various assumptions of the statistical tests when analyzing the results. The t – test combined with Cohen’s d was employed to assess the statistical significance of improvements in each set of evaluation results, including quality INVEST scoring and relevance measurement. In addition, the kappa score was utilized to assess the inter-rater agreement, with only fair or above agreement considered valid, reinforcing our analysis.

Another potential threat arises in the construction of the gold standard, which is intended to serve as a benchmark for evaluation. However, it may involve subjective elements. The possibility of presenting an alternative gold standard may change the results (Bragilovski et al. 2022). To mitigate bias, practitioners were invited to collaboratively construct the gold standard and engage in discussions to reach an agreement on its contents. During the evaluation process, two researchers meticulously compared the results with the gold standard. In cases where ambiguity arose, practitioners were involved in discussions to reach a consensus on the final decision. These threats are partially mitigated by practitioner engagement, but they still exist.

8.4 Construct validity

Construct validity primarily concerns the relationships between theory and observations, and these threats are mainly due to the methods employed to evaluate task outcomes (Wohlin et al. 2012; Bragilovski et al. 2022), which poses three potential threats in our study. In this study, we examined if SimAC improves LLMs’ capability in GAC. The dataset selection in the evaluation may affect the results. To mitigate the threat, we selected three projects that are easy to understand and cover different domains. As presented in Sect. 5.2, these three projects offer representative examples encompassing diverse levels of user story complexity for the purposes of our experimental evaluations.

The property of evaluation metrics presents another significant threat. To the best of our knowledge, there is no widely acknowledged metric designed specifically to assess the quality of US-AC. INVEST mnemonic is widely recognized as a guiding principle for crafting high-quality product backlog items (Cohn 2004; Dimitrijević et al. 2015; Buglione and Abran 2013; Khanh et al. 2017). Considering the significant relevance of the user story and acceptance criteria (Sedano et al. 2019) and following (Lombriser et al. 2016) to score INVEST on agile requirement quality, we employ INVEST scoring to evaluate the quality of the auto-generated US-AC. Furthermore, we invited experienced practitioners to make a more accurate assessment of the completeness and validity of auto-generated US-AC. This serves as a remedy for a series of potential problems caused by using the INVEST metric. Therefore, we contend that our evaluation approach is comprehensive.

Another potential threat arises from the evaluation sequence. This sequence may introduce bias into the evaluation process, as perceptions of outputs might be influenced by the order in which they are observed. To mitigate this bias, we offer a three-week evaluation window to the evaluators, enabling them to review their scoring based on a comprehensive understanding of the project background and outputs. Furthermore, to alleviate their workload and improve focus, we divided the evaluations into four groups, ensuring that each evaluator was assigned to fewer than two groups.

9 Conclusion

In this paper, we propose the SimAC framework incorporated with the role-based prompt to streamline the iterative discussions in GAC, aiming to reduce human effort in agile RE. To evaluate the effectiveness of SimAC, we applied it to GPT-3.5 and OpenLLaMA models to conduct empirical studies. To gain in-depth insights from SE and RE communities, we invited eight agile practitioners to participate in the evaluation process. The evaluation leverages INVEST metrics to assess the quality of auto-generated US-AC, particularly focusing on their quality across various attributes. The findings indicate that US-AC generated by SimAC with the *gpt-3.5-turbo* model achieve 60.60% completeness and 51.29% validity, representing significant improvements of 29.48% and 15.56%, respectively. Additionally, through an in-depth examination of specific cases, practitioners emphasized the refined US-AC as a valuable reference for subsequent user story elaboration. This study lays the groundwork for further exploration of automation in agile RE.

Implications for practitioners An increasing number of practitioners express interest in leveraging LLMs to aid in complex tasks. Although LLMs like the GPT-3.5 series exhibit acceptable outputs, our experimental findings indicate that they tend to yield less relevant results compared to those crafted by humans. SimAC effectively enhances LLM performance, producing higher-quality and more pertinent outcomes. Despite the potential risk of automating this task leading to team members having a limited understanding of requirements, generating high-quality acceptance criteria at the initial stage establishes a robust foundation for subsequent refinement, thereby promoting more effective collaboration.

Implications for researchers This study represents the first empirical investigation into the capabilities and limitations of recent LLMs in the automated generation of acceptance criteria. Subsequently, insightful case studies were conducted to highlight successes and shortcomings, offering potential avenues for further enhancement. Researchers are encouraged to devise more effective methods to harness the potential of LLMs. While the prompts meticulously crafted at each phase of the SimAC framework have been optimized, the possibility remains that alternative prompt designs may offer superior efficacy. Hence, we call for researchers to delve deeper into prompt design based on SimAC, thereby propelling automation in agile requirements engineering forward.

Instruction of questionnaire

Background introduction

Questionnaire for the effectiveness of different models in the task of generating acceptance criteria to elaborate the given user stories. The User Story associated with Acceptance Criteria is marked as "US-AC".

Please score each US-AC by the five attributes of INVEST evaluation criteria. The score of each attribute is 0 to 4.

- 0 = US-AC fulfilled the X attribute poorly.
- 1 = US-AC fulfilled the X attribute unsatisfactorily.
- 2 = US-AC fulfilled the X attribute to some extent.
- 3 = US-AC fulfilled the X attribute satisfactorily.
- 4 = US-AC fulfilled the X attribute very satisfactorily.

INVEST evaluation attributes:

- Attribute 1 (Independent): The US-AC should be as independent as possible. It can be implemented and delivered without being dependent on other stories.
- Attribute 2 (Negotiable): The US-AC is not an explicit contract and should leave space for discussion. It can be rewritten depending on business, market, technical, or any other type of requirement by team members.
- Attribute 3 (Valuable): The US-AC must deliver value to the stakeholders by clearly specifying solutions.
- Attribute 4 (Estimable): The US-AC should be estimable. It is well enough understood by team members to be able to determine its relative size (in development and cost).
- Attribute 5 (Small): The US-AC should not be so big as to become impossible to plan/task/prioritize within a level of accuracy.
- Attribute 6 (Testable): The US-AC must provide the necessary information to make test development possible.

The generations for different models are randomly ordered.

And US-ACs are randomly picked from 3 projects (e.g., g11, g24, g28). Brief description of three projects:

- g11 - New version of the NSF website: redesign and content discovery.
- g24 - Institutional data repository for the University of Bath.
- g28 - Citizen science platform that allows anyone to help in research tasks.

Actions

Please fill in the Tab Scores (Columns D to H) in ScoreSheets.xlsx.

- Column "Group No." - The number of projects.
- Column "US No." - The number of given user stories.
- Column "Model #" - The model number of the generated acceptance criteria.
- Column "Independent" - The score of Independent. (0 to 4)
- Column "Negotiable" - The score of Negotiable. (0 to 4)
- Column "Valuable" - The score of Valuable. (0 to 4)
- Column "Estimable" - The score of Estimable. (0 to 4)
- Column "Small" - The score of Small. (0 to 4)
- Column "Testable" - The score of Testable. (0 to 4)

Please fill in the Tab Information (Column B) in ScoreSheets.xlsx. Column A - Description of your information. Column B - Details of your information.

Sample of US-AC for evaluation

User story

As an NSF employee, I want to have editorial-planning recommendations so that I can understand how they might impact an eventual CMS / future web presence.

Acceptance criteria: (generated from Model #1)

- (1) The editorial-planning tool should be available to employees with access to the current CMS or future web presence.
- (2) The tool should provide recommendations for content that is likely to attract the most traffic or engagement.
- (3) The recommendations should be based on data insights and analysis.
- (4) The tool should be user-friendly and intuitive, enabling employees to easily input editorial content and receive recommendations.
- (5) Recommendations should include keywords or search terms that are likely to generate traffic.

Acceptance criteria: (generated from Model #2)

- (1) The editorial planning recommendations should be based on data-driven analysis, not just assumptions or opinions.
- (2) The CMS should provide a dashboard or interface for displaying the editorial planning recommendations for easy access for NSF employees.
- (3) The editorial planning recommendations should be displayed in an easy-to-understand format that presents the relevant information and makes the recommendations actionable.

- (4) The recommendations should address any potential SEO considerations.

Acknowledgements This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029).

Author contributions Yishu Li, Jacky Keung, and Zhen Yang designed and conducted the main experiments. Yishu Li, Zhen Yang, and Jacky Keung wrote the main manuscript text. Yishu Li and Zhen Yang designed the questionnaires. Yishu Li and Xiaozue Ma manual match the LLM solutions against with the gold standard. Xiaoxue Ma and Jingyu Zhang conducted comparative experiments. Xiaoxue Ma, Jingyu Zhang, and Shuo Liu consolidated the human evaluation results. All authors reviewed the manuscript.

Declarations

Conflict of interest The authors declare no competing interests.

References

- Achananuparp, P., Hu, X., Shen, X.: The evaluation of sentence similarity measures. In: Data Warehousing and Knowledge Discovery: 10th International Conference, DaWaK 2008 Turin, Italy, September 2-5, 2008 Proceedings 10, pp. 305–316. Springer (2008)
- Ahmed, M., Khan, S.U.R., Alam, K.A.: An NLP-based quality attributes extraction and prioritization framework in agile-driven software development. *Autom. Softw. Eng.* **30**(1), 7 (2023)
- Ali, N., Cai, H., Hamou-Lhadj, A., Hassine, J.: Exploiting parts-of-speech for effective automated requirements traceability. *Inf. Softw. Technol.* **106**, 126–141 (2019)
- Almanaseer, A.M., Alzyadat, W., Muhairat, M., Al-Showarah, S., Alhroob, A.: A proposed model for eliminating nonfunctional requirements in agile methods using natural language processes. In 2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA), pp. 1–7. IEEE (2022)
- Bjarnason, E., Unterkalmsteiner, M., Engström, E., Borg, M.: An industrial case study on test cases as requirements. In: Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings 16, pp. 27–39. Springer (2015)
- Bragilovski, M., Dalpiaz, F., Sturm, A.: Guided derivation of conceptual models from user stories: a controlled experiment. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 131–147. Springer (2022)
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Adv. Neural. Inf. Process. Syst.* **33**, 1877–1901 (2020)
- Buglione, L., Abran, A.: Improving the user story agile technique using the invest criteria. In: 2013 joint conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, pp. 49–53. IEEE (2013)
- Cardoso, J.R., Pereira, L.M., Iversen, M.D., Ramos, A.L.: What is gold standard and what is ground truth? *Dental Press J. Orthod.* **19**, 27–30 (2014)
- Carreño, L.V.G., Winbladh, K.: Analysis of user comments: an approach for software requirements evolution. In: 2013 35th international conference on software engineering (ICSE), pp. 582–591. IEEE (2013)
- Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al.: Universal sentence encoder. (2018) arXiv preprint [arXiv:1803.11175](https://arxiv.org/abs/1803.11175)
- Chipman, H. A., George, E. I., McCulloch, R. E.: Bart: Bayesian additive regression trees *The Annals of Applied Statistics* (2010). <https://doi.org/10.1214/09-AOAS285>
- Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. (2014) arXiv preprint [arXiv:1409.1259](https://arxiv.org/abs/1409.1259)

- Coe, R.: It's the effect size, stupid. *British Educ. Res. Assoc. Ann. Conf.* **12**, 14 (2002)
- Cohen, J.: A coefficient of agreement for nominal scales. *Educ. Psychol. Measur.* **20**(1), 37–46 (1960)
- Cohen, J.: *Statistical power analysis for the behavioral sciences*. Academic press, Cambridge (2013)
- Cohn, M.: *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, Boston (2004)
- Conboy, K., Fitzgerald, B.: Toward a conceptual framework of agile methods: a study of agility in different disciplines. In: *Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research*, pp. 37–44 (2004)
- Dalpiaz, F., Brinkkemper, S.: Agile requirements engineering with user stories. In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 191–200. IEEE (2012)
- Dalpiaz, F., Van Der Schalk, I., Brinkkemper, S., Aydemir, F.B., Lucassen, G.: Detecting terminological ambiguity in user stories: tool and experimentation. *Inf. Softw. Technol.* **110**, 3–16 (2019)
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. (2018) arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- Diebold, P., Ostberg, J.P., Wagner, S., Zender, U.: What do practitioners vary in using scrum? In: *Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25–29, 2015, Proceedings 16*, pp. 40–51. Springer (2015)
- Dimitrijević, S., Jovanović, J., Devedžić, V.: A comparative study of software tools for user story management. *Inf. Softw. Technol.* **57**, 352–368 (2015)
- Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via chatgpt. (2023) arXiv preprint [arXiv:2304.07590](https://arxiv.org/abs/2304.07590)
- Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. *Empir. Softw. Eng.* **23**, 452–489 (2018)
- Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., Zhang, J.M.: Large language models for software engineering: survey and open problems. (2023) arXiv preprint [arXiv:2310.03533](https://arxiv.org/abs/2310.03533)
- Ferrari, A., Abualhaija, S., Arora, C.: Model generation from requirements with llms: an exploratory study. (2024) arXiv preprint [arXiv:2404.06371](https://arxiv.org/abs/2404.06371)
- Ferrari, A., Gnesi, S.: Using collective intelligence to detect pragmatic ambiguities. In: *2012 20th IEEE International Requirements Engineering Conference (RE)*, pp. 191–200. IEEE (2012)
- Ferreira, A.M., da Silva, A.R., Paiva, A.C.: Towards the art of writing agile requirements with user stories, acceptance criteria, and related constructs. In: *ENASE*, pp. 477–484 (2022)
- Fischbach, J., Vogelsang, A., Spies, D., Wehrle, A., Junker, M., Freudenstein, D.: Specmate: Automated creation of test cases from acceptance criteria. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 321–331. IEEE (2020)
- Fried, D., Aghajanyan, A., Lin, J., Wang, S., Wallace, E., Shi, F., Zhong, R., Yih, W.t., Zettlemoyer, L., Lewis, M.: InCoder: A generative model for code infilling and synthesis. (2022) arXiv preprint [arXiv:2204.05999](https://arxiv.org/abs/2204.05999)
- Geng, X., Liu, H.: May. An open reproduction of llama, Openllama (2023). https://github.com/openlm-research/open_llama
- Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., Yin, J.: Unixcoder: unified cross-modal pre-training for code representation. (2022) arXiv preprint [arXiv:2203.03850](https://arxiv.org/abs/2203.03850)
- Gupta, A., Poels, G., Bera, P.: Creation of multiple conceptual models from user stories—a natural language processing approach. In: *Advances in Conceptual Modeling: ER 2019 Workshops FAIR, MREBA, EmpER, MoBiD, OntoCom, and ER Doctoral Symposium Papers, Salvador, Brazil, November 4–7, 2019, Proceedings 38*, pp. 47–57. Springer (2019)
- Hakala, K., Pyysalo, S.: Biomedical named entity recognition with multilingual bert. In: *Proceedings of the 5th workshop on BioNLP open shared tasks*, pp. 56–61 (2019)
- Halme, E., Vakkuri, V., Kultanen, J., Jantunen, M., Kemell, K.K., Rousi, R., Abrahamsson, P.: How to write ethical user stories? Impacts of the eccola method. In: *International Conference on Agile Software Development*, pp. 36–52. Springer International Publishing Cham (2021)
- Hey, T., Keim, J., Koziol, A., Tichy, W.F.: Norbert: Transfer learning for requirements classification. In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp.169–179. IEEE(2020)
- Hoang, M., Bihorac, O.A., Rouces, J.: Aspect-based sentiment analysis using bert. In: *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pp. 187–196 (2019)

- Hoda, R., Salleh, N., Grundy, J.: The rise and evolution of agile software development. *IEEE Softw.* **35**(5), 58–63 (2018)
- Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A., et al.: spacy: Industrial-strength natural language processing in python (2020). <https://spacy.io/>
- Hotomski, S., Glinz, M.: A qualitative study on using guidegen to keep requirements and acceptance tests aligned. In: 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 29–39. IEEE (2018)
- Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: Chatgpt for good? On opportunities and challenges of large language models for education. *Learn. Individ. Differ.* **103**, 102274 (2023)
- Kelly, A.: The Art of Agile Product Ownership: A Guide for Product Managers, Business Analysts, and Entrepreneurs, 93–123 (2019)
- Khanh, N.T., Daengdej, J., Arifin, H.H.: Human stories: a new written technique in agile software requirements. In: Proceedings of the 6th International Conference on Software and Computer Applications, pp. 15–22 (2017)
- Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. *Adv. Neural. Inf. Process. Syst.* **35**, 22199–22213 (2022)
- Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. *Biometrics* **33**, 159–174 (1977)
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. (2019) arXiv preprint [arXiv:1910.13461](https://arxiv.org/abs/1910.13461)
- Li, G., Hammoud, H.A.A.K., Itani, H., Khizbullin, D., Ghanem, B.: Camel: communicative agents for "mind" exploration of large scale language model society. (2023) arXiv preprint [arXiv:2303.17760](https://arxiv.org/abs/2303.17760)
- Li, Y., Keung, J., Ma, X., Chong, C.Y., Zhang, J., Liao, Y.: Llm-based class diagram derivation from user stories with chain-of-thought prompting. In: 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE (2024)
- Liu, F., Liu, Y., Shi, L., Huang, H., Wang, R., Yang, Z., Zhang, L.: Exploring and evaluating hallucinations in llm-powered code generation. (2024a) arXiv preprint [arXiv:2404.00971](https://arxiv.org/abs/2404.00971)
- Lombriser, P., Dalpiaz, F., Lucassen, G., Brinkkemper, S.: Gamified requirements engineering: model and experimentation. In: Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14–17, 2016, Proceedings 22, pp. 171–187. Springer (2016)
- Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: Improving agile requirements: the quality user story framework and tool. *Requir. Eng.* **21**, 383–403 (2016)
- Lucassen, G., Dalpiaz, F., Werf, J.M.E.v.d., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14–17, 2016, Proceedings 22, pp. 205–222. Springer (2016)
- Lucassen, G., Robeer, M., Dalpiaz, F., Van Der Werf, J.M.E., Brinkkemper, S.: Extracting conceptual models from user stories with visual narrator. *Requir. Eng.* **22**, 339–358 (2017)
- Ma, X., Keung, J.W., Yu, X., Zou, H., Zhang, J., Li, Y.: Attsum: a deep attention-based summarization model for bug report title generation. *IEEE Trans. Reliab.* **72**, 1663–1677 (2023)
- Manifesto, A.: Agile manifesto. *Haettu* **14**, 2012 (2001)
- Meredith, P., Summons, P., Park, M., Cheek, B.: What do employers expect from business analysts and is it captured by the "business analysis body of knowledge" (babok)? *ACIS 2019 Proceedings*. 93 (2019)
- Nema, P., Anthonysamy, P., Taft, N., Peddinti, S.T.: Analyzing user perspectives on mobile app privacy at scale. In: Proceedings of the 44th International Conference on Software Engineering, pp. 112–124 (2022)
- Nijkamp, E., Hayashi, H., Xiong, C., Savarese, S., Zhou, Y.: Codegen2: lessons for training llms on programming and natural languages. (2023) arXiv preprint [arXiv:2305.02309](https://arxiv.org/abs/2305.02309)
- Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., Xiong, C.: Codegen: an open large language model for code with multi-turn program synthesis. (2022) arXiv preprint [arXiv:2203.13474](https://arxiv.org/abs/2203.13474)
- Ozkaya, I.: Application of large language models to software engineering tasks: opportunities, risks, and implications. *IEEE Softw.* **40**(3), 4–8 (2023)

- Pandit, P., Tahiliani, S.: Agileuat: a framework for user acceptance testing based on user stories and acceptance criteria. *Int. J. Comput. Appl.* **120**(10), 16–21 (2015)
- Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., Abrahamsson, P.: Are software startups applying agile practices? the state of the practice from a large survey. In: *International Conference on Agile Software Development*, pp. 167–183. Springer, Cham (2017)
- Peng, Z., Rathod, P., Niu, N., Bhowmik, T., Liu, H., Shi, L., Jin, Z.: Environment-driven abstraction identification for requirements-based testing. In: *2021 IEEE 29th International Requirements Engineering Conference (RE)*, pp. 245–256. IEEE (2021)
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**(1), 5485–5551 (2020)
- Sedano, T., Ralph, P., Péraire, C.: The product backlog. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 200–211. IEEE (2019)
- Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving AI tasks with chatgpt and its friends in huggingface. (2023) arXiv preprint [arXiv:2303.17580](https://arxiv.org/abs/2303.17580)
- Spoletini, P., Ferrari, A.: The return of formal requirements engineering in the era of large language models. In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 344–353. Springer (2024)
- Subramanian, S., Mudumba, S.R., Sordoni, A., Trischler, A., Courville, A.C., Pal, C.: Towards text generation with adversarially learned neural outlines. *Adv. Neural Inf. Process. Syst.* **31** (2018). <https://proceedings.neurips.cc/paper/2018/hash/aaaccd2766ec67aebce26459bb828d81-Abstract.html>
- Sullivan, G.M., Feinn, R.: Using effect size-or why the p value is not enough. *J. Grad. Med. Educ.* **4**(3), 279–282 (2012)
- Sverrisdottir, H.S., Ingason, H.T., Jonasson, H.I.: The role of the product owner in scrum-comparison between theory and practices. *Procedia Soc. Behav. Sci.* **119**, 257–267 (2014)
- Thakur, J.S., Gupta, A.: Anmodeler: a tool for generating domain models from textual specifications. In: *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp. 828–833 (2016)
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. (2023) arXiv preprint [arXiv:2302.13971](https://arxiv.org/abs/2302.13971)
- Wang, T., Roberts, A., Hesslow, D., Le Scao, T., Chung, H.W., Beltagy, I., Launay, J., Raffel, C.: What language model architecture and pretraining objective works best for zero-shot generalization? In: *International Conference on Machine Learning*, pp. 22964–22984. PMLR (2022)
- Wang, X., Zhao, L., Wang, Y., Sun, J.: The role of requirements engineering practices in agile development: an empirical study. In: *Requirements Engineering: First Asia Pacific Requirements Engineering Symposium, APRES 2014, Auckland, New Zealand, April 28–29, 2014. Proceedings*, pp. 195–209. Springer (2014)
- Wang, Y., Wang, W., Joty, S., Hoi, S.C.: Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. (2021) arXiv preprint [arXiv:2109.00859](https://arxiv.org/abs/2109.00859)
- Wautelet, Y., Heng, S., Kolp, M., Mirbel, I., Poelmans, S.: Building a rationale diagram for evaluating user story sets. In: *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pp. 1–12. IEEE (2016)
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural. Inf. Process. Syst.* **35**, 24824–24837 (2022)
- White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. (2023) arXiv preprint [arXiv:2302.11382](https://arxiv.org/abs/2302.11382)
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: *Experimentation in Software Engineering*. Springer, Berlin (2012)
- Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: talking, drawing and editing with visual foundation models. (2023) arXiv preprint [arXiv:2303.04671](https://arxiv.org/abs/2303.04671)
- Xia, C.S., Wei, Y., Zhang, L.: Automated program repair in the era of large pre-trained language models. In: *Proceedings of the 45th International Conference on Software Engineering (ICSE 2023)*. Association for Computing Machinery (2023)

- Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T.: Automated extraction of security policies from natural-language software documents. In: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, pp. 1–11 (2012)
- Xue, P., L. Wu, Z. Yu, Z. Jin, Z. Yang, X. Li, Z. Yang, and Y. Tan.: Automated commit message generation with large language models: An empirical study and beyond. (2024) arXiv preprint [arXiv:2404.14824](https://arxiv.org/abs/2404.14824).
- Yang, Z., F. Liu, Z. Yu, J.W. Keung, J. Li, S. Liu, Y. Hong, X. Ma, Z. Jin, and G. Li.: Exploring and unleashing the power of large language models in automated code translation. (2024) arXiv preprint [arXiv:2404.14646](https://arxiv.org/abs/2404.14646).
- Zhang, J., Chen, Y., Niu, N., Liu, C.: A preliminary evaluation of chatgpt in requirements information retrieval. (2023) arXiv preprint [arXiv:2304.12562](https://arxiv.org/abs/2304.12562)
- Zhang, Y., Jin, Z., Xing, Y., Li, G.: Steam: simulating the interactive behavior of programmers for automatic bug fixing. (2023) arXiv preprint [arXiv:2308.14460](https://arxiv.org/abs/2308.14460)
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K.J., Ajagbe, M.A., Chioasca, E.V., Batista-Navarro, R.T.: Natural language processing for requirements engineering: a systematic mapping study. *ACM Comput. Surv. (CSUR)* **54**(3), 1–41 (2021)
- Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.: A survey of large language models. (2023) arXiv preprint [arXiv:2303.18223](https://arxiv.org/abs/2303.18223)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Yishu Li¹ · Jacky Keung¹ · Zhen Yang² · Xiaoxue Ma¹ · Jingyu Zhang¹ · Shuo Liu¹

✉ Zhen Yang
zhenyang@sdu.edu.cn

Yishu Li
yishuli5-c@my.cityu.edu.hk

Jacky Keung
Jacky.Keung@cityu.edu.hk

Xiaoxue Ma
xiaoxuema3-c@my.cityu.edu.hk

Jingyu Zhang
jzhang2297-c@my.cityu.edu.hk

Shuo Liu
sliu273-c@my.cityu.edu.hk

¹ Department of Computer Science, City University of Hong Kong, Hong Kong, China

² School of Computer Science and Technology, Shandong University, Qingdao, China