

# Agent4RE: A Self-Refining Multi-agent Framework for End-to-End Software Requirements Engineering and Benchmarking

Yongjian Tang\*

Siemens AG (RPD)

Germany

Technical University of Munich  
Munich, Germany

Linhan Li

Technical University of Munich

Munich, Germany

Thomas Runkler

Siemens AG

Germany

Technical University of Munich  
Munich, Germany

## Abstract

Existing LLM-based approaches for software Requirements Engineering (RE) typically rely on basic prompting strategies or rudimentary agent collaboration, under-utilizing the full potential of multi-agent systems. Meanwhile, available datasets focus on isolated subtasks, such as requirements extraction, classification, and completeness detection, leaving the absence of an end-to-end RE benchmark that spans from requirements elicitation to generation. We present Agent4RE – a self-refining multi-agent RE system that orchestrates specialized agents and incorporates two iterative improvement loops. To support evaluation, we construct RE-E2E<sup>1</sup> – a real-world dataset built from human-written requirement specifications, enabling end-to-end assessment of RE workflows. Building on this foundation, we further propose two enhanced Agent4RE versions that incorporate either autonomous self-refinement or structured human feedback, and analyze their strengths and limitations across different scenarios. Evaluation on 8 Large Language Models (LLMs) demonstrates that all three Agent4RE variants consistently outperform a domain-context-augmented prompting baseline by average 8% in text-based metrics. The two enhanced variants achieve the highest LLM-as-a-judge and human ratings, surpassing two RE baselines by approximately 0.8 points on a four-point scale. This consistent performance establishes Agent4RE as a practical end-to-end RE solution for industrial environments.

## CCS Concepts

• **Computing methodologies** → **Multi-agent systems**; *Cooperation and coordination*; • **Software and its engineering** → *Requirements analysis*; *Software development techniques*.

## Keywords

Multi-agent System, Large Language Models, Agents, Requirements engineering, Software Engineering

## 1 Introduction

Requirements Engineering (RE) serves as a preliminary stage in software development, establishing a shared understanding of stakeholders’ needs, goals, and constraints while providing the foundation for subsequent software design, implementation, and verification [20]. Traditionally, this process has required substantial effort from multiple roles – developers, business analysts, domain experts,

and project managers – who must iteratively perform elicitation, analysis, specification, and validation together to produce complete, consistent, and clear requirement documentation [29]. Given this labor-intensive nature and the fact that defects originating from this phase are significantly more expensive to fix in later stages, improvements in RE efficiency and quality can yield substantial benefits across the entire software development lifecycle.

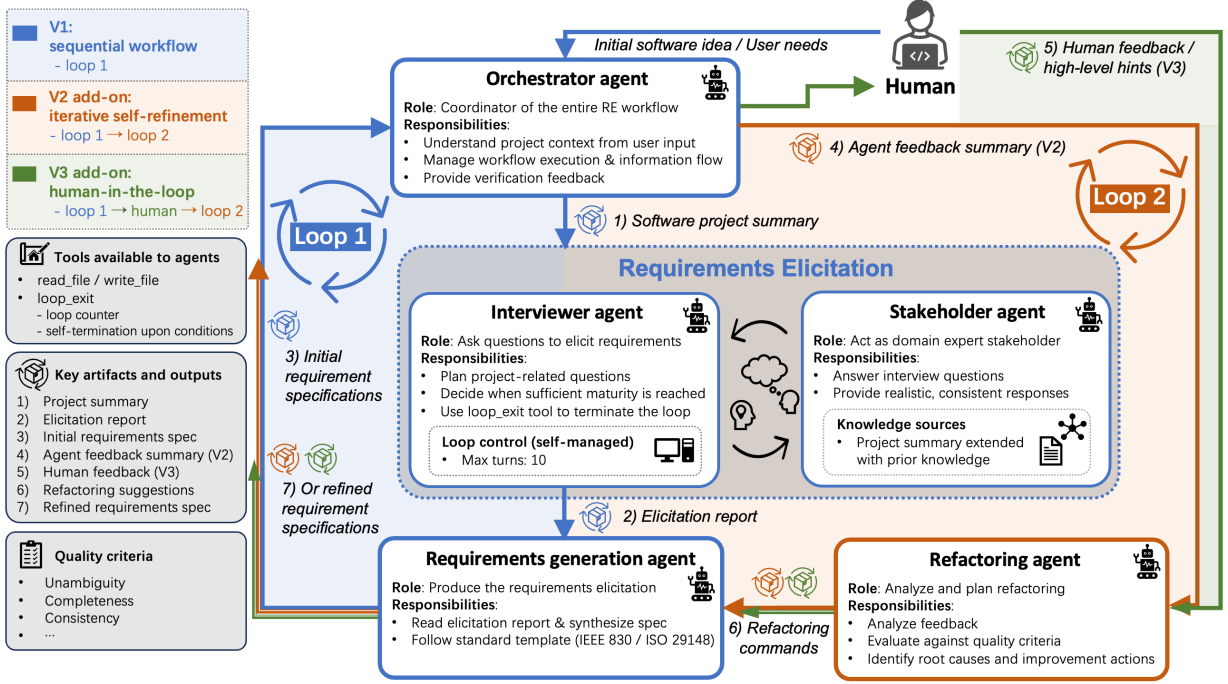
Recently, advancements in Large Language Models (LLMs) have opened new avenues for requirements engineering [11]. With superior capabilities in domain knowledge inference, natural language understanding, and ambiguity detection, LLMs are well suited to accelerate and assist diverse RE sub-tasks, such as extracting requirements from unstructured inputs, identifying inconsistencies in RE specifications, and generating documentation based on stakeholder instructions [46]. However, existing studies predominately focus on such isolated RE sub-tasks, achieving success only on individual activities, yet offering limited support for the complex, interconnected requirements engineering processes in real-world scenarios. While recent efforts have explored LLM-based interactive systems and agents to aggregate these isolated capabilities [42], no systematic end-to-end solution has been proposed and evaluated for the full RE lifecycle. Similarly, this fragmentation extends to data resources: accessible datasets are limited to separate RE tasks and no comprehensive open-source benchmark has been released to assess the entire requirements engineering process.

To address this gap, we propose Agent4RE – a self-refining multi-agent system for requirements engineering, which orchestrates specialized agents to simulate stakeholder interactions and user behaviors, enabling iterative elicitation, generation, and traceable refinement of requirements, as illustrated in Figure 1. Combined with RE-E2E – an end-to-end RE dataset, our framework mitigates the bottleneck of current LLM-based RE research, bridging isolated task-specific approaches with complete RE workflows. The main contribution of this work can be summarized as follows:

- (i) We propose Agent4RE – an autonomous multi-agent system for requirements engineering, which features two iterative improvement loops for self-refinement.
- (ii) We construct RE-E2E – an RE benchmark built from full-scale human-written requirement specifications. To the best of our knowledge, this represents the first end-to-end dataset designed for complete requirements engineering processes, spanning from initial project descriptions to finalized requirement specifications.

\*Corresponding author.

<sup>1</sup>We provide RE-E2E benchmark dataset, Agent4RE source code, and supplementary materials at [https://github.com/markustyj/Agent4RE\\_ASE-2026](https://github.com/markustyj/Agent4RE_ASE-2026).



**Figure 1: Agent4RE orchestrates specialized agents across two nested feedback loops: an inner generation loop that iteratively elicits requirements and drafts the initial specification, and an outer refactoring loop that continuously optimizes the specification through self-refinement. Agent4RE supports three operational modes: sequential workflow without feedback (blue), iterative self-refinement (orange add-on), and agentic workflow with human-in-the-loop (green add-on). Given a brief software project description from users, Agent4RE produces complete requirement specifications autonomously.**

- (iii) We analyze the performance of Agent4RE by integrating eight frontier LLMs covering different performance tiers, studying its effectiveness and robustness across diverse model capabilities and computational budgets. Furthermore, we showcase the self-refinement features of all language models by analyzing their iterative improvement loops.
- (iv) We validate our approach by comparing the generated requirement specifications with the ground-truth in RE-E2E. Evaluation results reveal that all Agent4RE variants consistently outperform a context-augmented prompting baseline by average 8% in text-based metrics. Furthermore, the two enhanced variants achieve the highest LLM-as-a-judge and human ratings, surpassing two baselines by approximately 0.8 points on a four-point scale.

## 2 Related Work

### 2.1 Large Language Models for RE-related Tasks

Extensive language model-based approaches have been proposed for diverse stages throughout the requirements engineering cycle. Building on the early transformer architecture [44], researchers fine-tuned BERT to detect anaphoric ambiguity in requirements and evaluated it against traditional supervised entity recognition techniques [9]. A similar approach was introduced to identify intra-domain ambiguities in the requirements of interdisciplinary projects [26]. Given BERT’s encoder-based architecture, parallel studies trained

specialized BERT variants for requirements classification, achieving improved accuracy over machine learning techniques like decision trees and support vector machines [15, 23]. As language models have scaled up, strategic few-shot prompting has emerged as a more streamlined solution to requirements classification, delivering on-par performance with fine-tuned models while eliminating the computational overhead of training [8, 40, 43].

The superior reasoning capabilities of these large-scale models enable effective requirements analysis while their generative nature aligns with the needs for requirements elicitation and specification generation. Given this potential, a preliminary evaluation [1] explored the role of LLMs across different RE stages and suggested tailored prompt patterns for elicitation, analysis, specification, and validation tasks. Building on this foundation, LLMREI [19] employs LLMs as automated interviewers within a chatbot to simulate the requirements elicitation process with human stakeholders, designed to minimize human intervention and mitigate common interview-related errors. SpecGen [24] leveraged a two-phase prompting approach to generate precise behavioral comments for program-level specifications, while ReqGPT [37], a 7B-parameter domain-specific LLM fine-tuned on over one hundred curated requirement lists, was developed to improve specification quality. Despite these breakthroughs in specific RE aspects, existing methods still fall short in delivering complete requirement specifications for production environments, indicating the need for more robust and comprehensive approaches.

## 2.2 LLM-based Agentic Systems for Automated RE Process

The collaborative workflow of multi-agent systems offers a promising paradigm to tackle more complex end-to-end RE challenges [41]. As an early exploration [35], researchers implemented an LLM-based agentic system with GPT-3.5 and GPT-4 to automate user story elicitation, requirements generation, and prioritization for agile software projects. Extending this prior work, a subsequent study proposed a role-based multi-agent system that emulates stakeholder roles to collaboratively generate and prioritize requirements from project descriptions [36]. Another modular architecture ReqInOne breaks down requirements generation into three sub-tasks: summary, extraction, and classification [48], using tailored prompt templates to improve the quality and consistency of LLM outputs. In comparison, Elicitron has focused on more specific RE activities [2]. The system simulates elicitation interviews to explore product experiences and identify stakeholders' needs, thus enabling the discovery of unforeseen use cases through structured LLM interactions. Moving toward end-to-end automation, iReDev [18], previously also called MARE [17], represents a knowledge-driven multi-agent framework designed to address the full requirements engineering process. It incorporates human expert knowledge into six agents via chain-of-thought prompting and structured reasoning, enabling them to manage an event-driven artifact pool collaboratively. Nevertheless, due to the absence of resources for end-to-end RE assessment, the evaluation focused primarily on intermediate steps such as requirements modeling and entity extraction. The final specifications were evaluated only through subjective metrics such as completeness, correctness, and consistency.

## 2.3 Baseline Selection

Given our focus on end-to-end requirements engineering, approaches that specifically address intermediate RE steps or sub-tasks, such as requirements elicitation and prioritization, are poorly suited and insufficient for our purposes, as illustrated in Table 1. Therefore, we select the only two existing baselines that address the complete RE lifecycle: a prompting-based approach with tailored prompt patterns for different RE stages [1], and iReDev, a multi-agent collaboration framework for RE [18].

**Table 1: Application domains of selected BERT variants, LLM-based approaches, and agentic solutions for RE.**

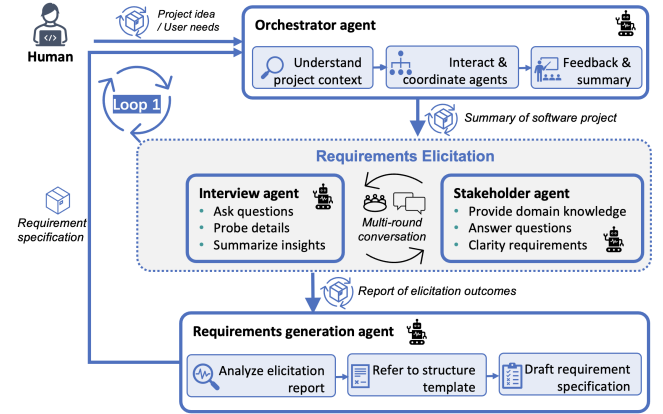
| Approaches             | Covered RE stages |          |               |            |
|------------------------|-------------------|----------|---------------|------------|
|                        | Elicitation       | Analysis | Specification | Validation |
| BERT ambiguity [9]     |                   | ✓        |               |            |
| NoRBERT [15]           |                   | ✓        |               |            |
| PRCBERT [23]           |                   | ✓        |               |            |
| Over-prompting [43]    |                   | ✓        |               |            |
| Prompt patterns [1]    | ✓                 | ✓        | ✓             | ✓          |
| LLMREI [19]            | ✓                 |          |               |            |
| SpecGen [24]           |                   | ✓        | ✓             |            |
| ReqGPT [37]            |                   | ✓        | ✓             |            |
| Early agents [35]      | ✓                 | ✓        |               |            |
| Role-based agents [36] | ✓                 | ✓        | ✓             |            |
| Elicitron [2]          | ✓                 |          |               |            |
| ReqInOne [48]          |                   | ✓        | ✓             |            |
| iReDev [18]            | ✓                 | ✓        | ✓             | ✓          |
| Agent4RE               | ✓                 | ✓        | ✓             | ✓          |

## 3 Agent4RE

Our multi-agent system offers three different versions, each with distinct advantages: 1) a sequential workflow without feedback for streamlined RE processes; 2) a workflow with self-refinement that enables the system to iteratively improve the outputs; 3) a workflow with human feedback that incorporates domain experts to guide the refinement process.

### 3.1 Sequential Workflow without Feedback

Figure 2 illustrates the first version: a sequential workflow of Agent4RE. The multi-agent system starts with an orchestrator agent  $A_O$  responsible for communicating and interacting with human users. Once receiving a brief project idea  $I$ , the orchestrator analyzes and understands the project context using its pre-trained knowledge. Project summary  $S$  will be generated to guide subsequent procedures:  $S = A_O(I)$ .



**Figure 2: The sequential workflow of Agent4RE consists of an orchestrator agent that understands the project scope by analyzing the user message, an interviewer agent and stakeholder agent that simulate requirements elicitation with self-controlled elicitation loops, and a generation agent that finalizes the requirement specifications.**

The iterative requirements elicitation process represents the core step of this workflow and is self-controlled by an interviewer agent  $A_I$  and a stakeholder agent  $A_S$ . Given a message from the orchestrator agent, the interviewer agent initiates the conversation by posing a list of project-related questions  $q$ . Meanwhile, the stakeholder agent familiarizes itself with the project domain by reading the orchestrator-generated project summary and combing its own knowledge base to answer interviewers' questions. This simulated interview mirrors real-world elicitation scenarios and continues for multiple rounds until the discussion outcomes reach sufficient maturity. Using the core notation and functional mappings listed in Table 2, each iteration follows the process:

$$q_t = A_I(q_{t-1}, a_{t-1}, S); \quad a_t = A_S(q_t, S). \quad (1)$$

To ensure self-management, we equip the interviewer agent with a loop exit tool that enables autonomous termination of the elicitation process. Notably, the tool incorporates a loop counter,

**Table 2: Core Notation and Functional Mappings.**

| Notation               | Description / Functional Definition                                       |
|------------------------|---------------------------------------------------------------------------|
| $I$                    | The idea of raw project                                                   |
| $S$                    | The summary of software project generated by the orchestrator             |
| $q_t, a_t$             | Elicitation questions and answers exchanged at turn $t$                   |
| $R$                    | The report summarizing the elicitation history                            |
| $T_{re}$               | The template for generated requirement specifications                     |
| $D_k$                  | The document of requirement specification at refinement iteration $k$     |
| $F_a, F_h$             | The set of agent and human feedback in bullet points                      |
| $A_O(I)$               | <b>Orchestrator Agent:</b> Analyzes context and validates deliverables    |
| $A_I(q_{t-1}, a_t, S)$ | <b>Interviewer Agent:</b> Generates exploratory elicitation questions     |
| $A_S(q_t, S)$          | <b>Stakeholder Agent:</b> Simulates answers using its domain knowledge    |
| $A_G(R, T_{re})$       | <b>Generator Agent:</b> Synthesizes or refines the specification document |
| $A_R(D, F_a, F_h)$     | <b>Refactor Agent:</b> Formulates targeted quality-improvement commands   |

with a pre-set maximum of 10 iterations to avoid redundant repetition in practice. In the end, the interviewer summarizes the entire conversation into an elicitation report  $R$ :

$$R = \text{Summarize}(\{(q_1, a_1), (q_2, a_2), \dots, (q_t, a_t)\}), t \leq 10. \quad (2)$$

With the detailed report, the requirements generation agent  $A_G$  produces the initial specification document:  $D_0 = A_G(R, T_{re})$ . To ensure structural compliance, the required specification templates  $T_{re}$  [38, 39] are integrated into LLMs' system message as a reference, steering the agent to strictly follow the structure. Due to the document length, the generation time typically ranges from a few seconds to over one minute, depending on the model size and inference configuration. Once finished, the generated specification is returned to the orchestrator for validation, which verifies the document before handing over to human users. In case of any deficiencies, it leverages its supervisory control over other agents to perform corrective action. Further investigation of this self-controlled iterative elicitation loop is provided in Section 4.6.

### 3.2 Agentic Workflow with Self-refinement

As illustrated in Figure 1, an additional refinement loop (in orange) is introduced to further enhance the quality of the generated specifications. A refactoring agent  $A_R$  serves as the central component in this process, analyzing the verification results from the initial loop (in blue) and applying pre-defined requirements quality criteria such as unambiguity, completeness, and consistency to formulate improvement suggestions. These generated refactoring suggestions are then forwarded to the generation agent, which refines the original specification accordingly. This agent-based self-refinement loop can also be executed multiple times until the specification meets the demanded quality standards, which can be formally represented as:  $A_R(D_k, F_a)$ , where  $D_k$  represents the generated specification document in the  $k$ -th round and  $F_a$  the set of feedback from agents.

### 3.3 Agentic Workflow with Human Feedback

While automated agentic workflows streamline the RE process, including human-in-the-loop at critical decision points can further enhance the quality and reliability of the generated specifications [28]. Therefore, in the third version of Agent4RE, we incorporate human feedback  $F_h$  after the initial sequential loop and prior to the refactoring loop. In this case, human developers provide domain-specific insights and expertise alongside the verification comments from the orchestrator agent to guide the refactoring agent. In our

experiments, all human participants are experienced software engineers and they have access to the ground-truth specifications from the annotated dataset. However, to simulate realistic scenarios, human feedback is limited to concise bullet points or high-level hints regarding specification structure rather than detailed corrections.

The progression of the document from state  $D_k$  to  $D_{k+1}$  across the three distinct versions of Agent4RE can be formulated as:

$$D_{k+1} = A_G(D_k, A_R(D_k, F_a, F_h)). \quad (3)$$

### 3.4 Implementation and Configuration

We implement Agent4RE using the Google AI Development Kit (ADK), which provides native support for agent coordination and workflow management [13]. Each agent in our system is instantiated as an independent agent with specialized prompts and tool configurations. For instance, the orchestrator agent – responsible for managing and coordinating all specialized agents – operates using the system prompt demonstrated in Figure 3. Due to space constraints, the complete prompt templates for the remaining agents are provided in the supplementary materials.

|                                                                                                                                                                                                |  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| You are an ["OrchestratorAgent"] responsible for communicating with the user of a new software project.                                                                                        |  |
| <b>Role Assignment</b>                                                                                                                                                                         |  |
| Your task is to understand his needs for the new software project during the conversation and help him complete the requirements engineering process with other agents together.               |  |
| <b>Task Description</b>                                                                                                                                                                        |  |
| <b>Process:</b>                                                                                                                                                                                |  |
| 1) If you receive a user message, call the ["SequentialAgentLoop1"].                                                                                                                           |  |
| 2) Send your ["software_project_summary"] to the ["StakeholderAgent"]. The ["InterviewerAgent"] will start the interaction loop with ["StakeholderAgent"].                                     |  |
| 3) Once you see the ["requirement_elicitation_report"] from the ["InterviewAgent"], call the ["RequirementsGenerationAgent"] to generate the specifications.                                   |  |
| 4) Once you see the ["generated_requirements"] from the ["RequirementsGenerationAgent"], ask the user whether he or she is satisfied with the generated specifications.                        |  |
| 5) If the user is not satisfied with the generated specifications and give you feedback and suggestions, you should summarize a report of his further needs and call ["SequentialAgentLoop2"]. |  |
| 6) The ["SequentialAgentLoop2"] contains ["RefactoringAgent"] and ["RequirementsGenerationAgent"] to refine the specifications.                                                                |  |
| <b>Workflow Explanation</b>                                                                                                                                                                    |  |
| <b>Output:</b>                                                                                                                                                                                 |  |
| The refined ["requirement_specification"] from ["SequentialAgentLoop2"]                                                                                                                        |  |
| <b>Structured Output</b>                                                                                                                                                                       |  |

**Figure 3: The system prompt for the orchestrator agent used in self-refinement with human feedback.**

Unlike previous agentic solutions that apply uniform LLM settings, we adopt task-specific temperature configurations for all agents, as detailed in Table 3. Temperature controls the randomness of the model's output, with lower values producing more deterministic responses and higher values encouraging creativity. This differentiated approach allows Agent4RE to maximize each agent's potential for their task.

**Table 3: Configuration of agent temperature.**

| Agents&Temperature | Consideration                                  |
|--------------------|------------------------------------------------|
| Orchestrator 0.5   | Balanced temperature for coordination tasks    |
| Interviewer 1      | Higher temperature for exploratory elicitation |
| Stakeholder 1      | Higher temperature for exploratory elicitation |
| Generator 0.8      | Higher temperature for creative generation     |
| Refactor 0.2       | Lower temperature for consistent suggestion    |

## 4 Experiments

### 4.1 RE-E2E Dataset

Existing data resources for RE predominantly focus on individual stages, suffering from several limitations: incomplete coverage, poor community visibility, and limited support for end-to-end evaluation [27]. For instance, PROMISE [21], a requirements classification benchmark extensively used in prior work, contains only isolated requirement items. Similarly, PURE [10] provides requirement specification documents for various software systems, but lacks a unified structural template for standardized processing and systematic evaluation. And a more recent RE dataset [4] provides a collection of user stories pairing with the corresponding domain models, yet it still excludes complete requirement specifications. As LLM capabilities and agentic workflows advance continuously, the ambition within the research community and industry has grown beyond addressing isolated RE tasks toward autonomous management of the entire RE lifecycle. Addressing data scarcity has become an urgent priority to enable the exploration of end-to-end agentic solutions.

To address this research gap, we curate RE-E2E, a dataset used to benchmark the performance of different frameworks for end-to-end software requirement engineering tasks. We selectively collect 50 complete requirement specifications in PDF format from real-world software repositories [34]. Detailed statistics of the files are presented in Table 4. All files originate from experienced software engineers and cover various software domains, such as software control, e-commerce, and security verification systems, and translation software.

Based on the original specification template used by developers, we divide the collected PDF files into two sets: one adhering to IEEE 830 [38], which defines the structure for traditional software requirements specifications, and another following IEEE 29148 [39], an updated standard that incorporates systems and software engineering requirements with more modern practices. Specifically, we use regular expressions [12] to clean noise such as unexpected newline characters and consecutive spaces, obtaining normalized plain text. However, in some edge cases, the original section names of human-written specifications do not strictly follow the IEEE format. To create a dataset with unified structure, we match these sections with the official IEEE section names based on their semantic embeddings. To ensure dataset reproducibility, we use the open-source Qwen3-embedding-8B to match RE-sections. If the cosine similarity between two sections falls below 0.7, human annotators must intervene. They review each section independently and reach consensus on whether to remove or retain it as empty. The cross-annotator agreement is 96.7%, meaning that all annotators make the same binary decision in most cases. For the remaining 3.3%, annotators follow majority vote. In Table 4, we record the section empty rates.

Based on the finalized specifications, human annotators manually write the initial project descriptions, which are then syntactically checked by LLMs to ensure grammatical correctness and formatting compliance. Notably, these annotators also serve as human evaluators in experiments. Their prior experience with the annotation process ensures their familiarity with the content. Further details about their information are provided in Section 4.3. The resulting project descriptions serve as inputs for Agent4RE. The

processed descriptions and software requirement specifications are provided in the supplementary materials.

### 4.2 Selected Language Models

Our experiments involve eight LLMs spanning different performance tiers to investigate the full potential of our agentic framework. We deploy four frontier models via APIs, including GPT-5 [30], GPT-4.1 [31], GPT-4o [16], and Gemini-3-flash [5], which represent the forefront of LLM capabilities. To complement these proprietary models, we incorporate four open-source alternatives that ensure reproducibility and broader applicability: LLaMA3.1-8B from Meta AI [7], Qwen3-8B from Alibaba [45], Mistral3-8B from Mistral [25], and Gemma4-4B from Google [6].

### 4.3 Metrics

To compare the generated requirement specifications with the ground-truth in RE-E2E, we employ an extensive set of evaluation approaches, including text-based metrics, semantic-based metrics, LLM-as-a-judge, and human evaluators.

- 1) Regarding text-based metrics, we apply **METEOR** [3], **BLEU-1** [32], and **ROUGE-L** [22]. METEOR evaluates text similarity through a harmonic mean of precision and recall, incorporating stemming, synonymy, and word order penalties. BLEU-1 measures the proportion of individual words (unigrams) in the generated text that match the reference text, while ROUGE-L evaluates the longest common subsequence between the candidate and reference texts.
- 2) For semantic-based metrics, we apply **BERTScore** [47] and **SentenceBert** [33]. BERTScore computes token-level semantic similarity by comparing BERT-based word embeddings between the candidate and reference texts, while SentenceBert measures sentence-level similarity by encoding texts into dense vector representations and calculating their cosine similarity.
- 3) **LLM-as-a-judge** leverages prompt-based assessment to grade the generated requirement specifications. Figure 4 presents the prompt template used for this approach, with grades assigned on a scale from 1 to 4. To assess RE quality in practice, developers typically require a language checker for ambiguity, an app-lifecycle tool for completeness, and a formal/model checker for logical consistency, leading to increased toolchain complexity. Since these quality criteria are fundamentally grounded in written text and requirements, LLMs are natural requirements evaluators given their expertise in language processing. However, this suitability does not apply to all model scales. While large-scale models demonstrate promising capabilities in judging text quality, compact models often struggle [14] – even the best one misjudges. Therefore, we deploy this evaluation strategy exclusively to four top-tier models (GPT-5, GPT-4.1, GPT-4o, and Gemini-3-flash).
- 4) We further include **human** evaluation for comparison. To align with LLM-as-a-judge, three software developers from industry assess summaries independently on a scale from 1 to 4, with final scores averaged for fairness. We select

**Table 4: Statistics of the requirement specification dataset.**

| IEEE 830-1998                  |           | IEEE 29148-2018                |           |
|--------------------------------|-----------|--------------------------------|-----------|
| Statistics                     | Value     | Statistics                     | Value     |
| Total dataset tokens           | 107 887   | Total dataset tokens           | 72 157    |
| Number of collected files      | 30        | Number of collected files      | 20        |
| Average letters per file       | 18 462.43 | Average letters per file       | 17 673.90 |
| Average tokens per file        | 3596.23   | Average tokens per file        | 3607.85   |
| Average sections per file      | 37.97     | Average sections per file      | 45.15     |
| Average pages per file         | 20.53     | Average pages per file         | 22.85     |
| Rate of non-empty sections (%) | 84.23     | Rate of non-empty sections (%) | 79.12     |

human evaluators based on their software development experience and RE domain expertise. Finally, we choose three senior software engineers, each with two to three years of development experience in industry. In addition to their own judgment criteria, we also provide them with the prompt and grading schema used in LLM-as-a-judge to ensure consistent evaluation.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| You will be given two Software Requirements Specifications (SRS):<br>- An LLM-generated SRS draft.<br>- A human-written ground truth SRS.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                   |
| <b>Your task:</b><br>- Compare LLM-generated SRS to ground truth for semantic alignment.<br>- Judge the quality of the LLM-generated SRS based on IEEE 29148<br>- Provide one final rating (1-4) that balances both alignment and intrinsic quality.                                                                                                                                                                                                                                                                                                                                                                                                                                             | <b>Task Description</b>                                           |
| <b>Rating scale:</b><br><b>1: Poor</b> – Large misalignment and poor quality. Many template sections missing; highly ambiguous, incomplete, inconsistent requirements; contradictions; weak organization.<br><b>2: Weak</b> – Partial alignment with notable gaps. Important sections incomplete; ambiguous or unmeasurable requirements; limited interfaces/NFRs; weak traceability.<br><b>3: Good</b> – Mostly aligned and generally consistent. Minor gaps or ambiguities; mostly testable and organized; clear improvement actions remain.<br><b>4: Excellent</b> – Strong alignment and high intrinsic quality. Complete, unambiguous, traceable, well-structured; adheres to the template. |                                                                   |
| <b>Inputs:</b><br><b>LLM-Generated SRS:</b><br>{llm_srs_content}<br><b>Human Ground Truth SRS:</b><br>{human_srs_groundtruth_content}<br>Optional Project Context:<br>{project_context}<br>Optional SRS Template Reference:<br>{template_content}                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                   |
| <b>Output format (must follow exactly):</b><br>Evaluation: (your comments for the rating)<br><b>Total rating: (a number between 1 and 4)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <b>Mandatory &amp; Optional Input</b><br><br><b>Output Format</b> |

**Figure 4: The prompt used for LLM-as-a-judge spans four rating scales, each with detailed criteria to assess requirements on ambiguity, completeness, and other dimensions.**

#### 4.4 RE Specification Generation

Since iterative human feedback has no theoretical upper limit on performance, we perform one optimization loop for all Agent4RE variants to ensure fair comparison. Table 5 demonstrates the evaluation results for end-to-end specification generation in the next page. Agent4RE significantly outperforms the two baselines in most considered metrics, with each variant exhibiting superior performance in different evaluation dimensions. For traditional text- and

semantic-based metrics, incorporating high-level feedback from human developers yields the best results, achieving 0.251 in ME-TOR, 0.273 in Rouge-L, 0.063 in BLEU-1, 0.838 in BERT F1 score, and 0.392 in Sentence-BERT similarity. As for LLM-as-a-judge, pure self-refinement with GPT-5 slightly outperforms the human-in-the-loop variant, with both GPT-5 and GPT-4.1 models securing perfect 4.0 ratings when evaluated by GPT-4.1. However, upon human evaluation, self-refinement with human feedback still performs best, scoring the highest 3.29 out of 4. In comparison, the two baselines fall short, reaching 2.22 and 2.87 respectively in human evaluation.

If we zoom in model performance for detailed analysis, three GPT models lead the performance in most metrics, especially when deployed within the sequential workflow without feedback or self-refinement with human feedback. Among them, the latest GPT-5 model demonstrates exceptional performance in LLM-as-a-judge and human evaluation, achieving near-perfect scores from other LLM evaluators when integrated into the Agent4RE framework. However, an unexpected finding emerges with pure self-refinement: iterative improvement loops lead to degraded performance on semantic-based metrics, as evidenced by Sentence BERT. This phenomenon may stem from LLMs’ deliberate attempts to modify content, and such purposeful revisions can result in substantial semantic drift. We provide a case study in the supplementary materials, including an example of software project description, human feedback, and the generated requirement specifications. Due to the length of the generated content, a separate JSON file recording the output of each agent throughout the complete agentic workflow is included in the uploaded repository.

Detailed resource consumption for each model is reported in Table 6. Among all models, GPT-5 leads to the highest cost, consuming 33.74K tokens per specification at an estimated cost of 0.127\$ and producing 4.68K tokens on average in the outputs. With respect to inference latency, it takes approximately 33.7 seconds to complete the full agentic workflow and generate the final requirement specification. In comparison, other large models are more efficient, generally requiring fewer tokens and inference time per specification, but this result is a trade-off in performance and output quality, as evidenced by LLM-as-a-judge and human evaluation – two main metrics in Table 5. We run the four compact models locally through Ollama, thus introducing no API cost. But given our hardware – MAC pro with an Apple M3 chip and 36GB of memory – inference latency remains considerable, particularly for Qwen3-8B, which requires 66.5 seconds per specification.

**Table 5: Comparison of generated requirement specifications with ground-truth data, evaluated using text-based, semantic-based, and LLM-based metrics, as well as human evaluations.**

| Baselines & Agent4RE                                 |                | Models       | Text Based   |              |              | Semantic Based |               | LLM-as-a-judge |             |             |                | Human |
|------------------------------------------------------|----------------|--------------|--------------|--------------|--------------|----------------|---------------|----------------|-------------|-------------|----------------|-------|
|                                                      |                |              | METEOR       | Rouge-L      | BLEU-1       | BERTScore (F1) | Sentence-BERT | GPT-4o         | GPT-4.1     | GPT-5       | Gemini-3-flash |       |
| Baseline<br>prompt patterns                          | GPT-5          | <b>0.183</b> | 0.102        | 0.014        | 0.810        | 0.378          | <b>3.12</b>   | <b>3.28</b>    | <b>2.24</b> | <b>3.20</b> | 2.13           |       |
|                                                      | GPT-4.1        | 0.145        | 0.117        | <b>0.015</b> | 0.806        | 0.350          | 3.04          | 3.10           | 1.76        | 3.02        | <b>2.22</b>    |       |
|                                                      | GPT-4o         | 0.118        | 0.124        | 0.010        | 0.810        | <b>0.412</b>   | 2.68          | 2.34           | 1.48        | 2.52        | 1.81           |       |
|                                                      | Gemini-3-flash | 0.122        | 0.104        | 0.011        | 0.811        | 0.364          | 2.74          | 2.46           | 1.66        | 3.24        | 1.92           |       |
|                                                      | LLaMA-3.1-8B   | 0.133        | 0.131        | 0.012        | <b>0.817</b> | 0.363          | 2.56          | 2.34           | 1.48        | 1.96        | 1.52           |       |
|                                                      | Qwen3-8B       | 0.117        | <b>0.132</b> | 0.008        | 0.810        | 0.344          | 2.76          | 2.58           | 1.76        | 2.42        | 1.78           |       |
|                                                      | Ministral3-8B  | 0.110        | 0.115        | 0.005        | 0.788        | 0.332          | 3.02          | 2.96           | 1.72        | 2.92        | 1.82           |       |
|                                                      | Gemma4-4B      | 0.047        | 0.066        | 0.002        | 0.802        | 0.212          | 2.48          | 1.76           | 1.20        | 2.04        | 1.31           |       |
| Baseline<br>iReDev (MARE)                            | GPT-5          | <b>0.197</b> | 0.102        | 0.012        | 0.816        | 0.374          | <b>3.84</b>   | 3.88           | <b>2.62</b> | <b>3.76</b> | <b>2.87</b>    |       |
|                                                      | GPT-4.1        | 0.175        | 0.132        | 0.016        | 0.818        | 0.402          | 3.58          | <b>3.92</b>    | 2.44        | 3.70        | 2.80           |       |
|                                                      | GPT-4o         | 0.160        | <b>0.162</b> | <b>0.018</b> | <b>0.822</b> | <b>0.418</b>   | 3.20          | 3.34           | 2.16        | 3.14        | 2.70           |       |
|                                                      | Gemini-3-flash | 0.158        | 0.136        | 0.015        | 0.815        | 0.376          | 3.36          | 3.54           | 2.22        | 3.72        | 2.72           |       |
|                                                      | LLaMA-3.1-8B   | 0.102        | 0.128        | 0.008        | 0.821        | 0.342          | 2.36          | 2.40           | 1.60        | 2.52        | 2.02           |       |
|                                                      | Qwen3-8B       | 0.107        | 0.134        | 0.008        | 0.812        | 0.330          | 3.12          | 3.50           | 2.00        | 3.36        | 2.20           |       |
|                                                      | Ministral3-8B  | 0.095        | 0.118        | 0.005        | 0.805        | 0.398          | 3.26          | 3.50           | 2.04        | 3.30        | 2.09           |       |
|                                                      | Gemma4-4B      | 0.090        | 0.122        | 0.004        | 0.810        | 0.302          | 2.54          | 2.40           | 1.48        | 2.50        | 1.86           |       |
| Agent4RE:<br>Sequential workflow<br>without feedback | GPT-5          | <b>0.211</b> | 0.098        | 0.011        | 0.814        | 0.370          | <b>3.96</b>   | 3.94           | <b>2.76</b> | <b>3.80</b> | <b>2.92</b>    |       |
|                                                      | GPT-4.1        | 0.180        | 0.137        | 0.017        | 0.816        | 0.396          | 3.66          | <b>4.00</b>    | 2.66        | 3.74        | 2.84           |       |
|                                                      | GPT-4o         | 0.167        | <b>0.171</b> | <b>0.020</b> | <b>0.823</b> | <b>0.414</b>   | 3.14          | 3.28           | 2.10        | 3.10        | 2.67           |       |
|                                                      | Gemini-3-flash | 0.162        | 0.142        | 0.017        | 0.813        | 0.372          | 3.42          | 3.58           | 2.28        | 3.76        | 2.75           |       |
|                                                      | LLaMA-3.1-8B   | 0.096        | 0.123        | 0.007        | 0.823        | 0.336          | 2.28          | 2.34           | 1.54        | 2.46        | 2.01           |       |
|                                                      | Qwen3-8B       | 0.100        | 0.129        | 0.007        | 0.810        | 0.325          | 3.04          | 3.48           | 2.14        | 3.32        | 2.36           |       |
|                                                      | Ministral3-8B  | 0.090        | 0.113        | 0.004        | 0.803        | 0.393          | 3.2           | 3.48           | 2.06        | 3.28        | 2.24           |       |
|                                                      | Gemma4-4B      | 0.094        | 0.126        | 0.005        | 0.812        | 0.296          | 2.48          | 2.32           | 1.72        | 2.44        | 2.06           |       |
| Agent4RE:<br>Pure<br>self-refinement                 | GPT-5          | <b>0.210</b> | 0.093        | 0.008        | 0.807        | 0.319          | <b>3.92</b>   | <b>4.00</b>    | <b>2.82</b> | <b>3.94</b> | <b>3.12</b>    |       |
|                                                      | GPT-4.1        | 0.183        | 0.118        | <b>0.016</b> | 0.807        | 0.322          | 3.72          | 4.00           | 2.66        | 3.90        | 3.04           |       |
|                                                      | GPT-4o         | 0.151        | <b>0.144</b> | 0.011        | 0.814        | 0.307          | 3.28          | 3.32           | 2.14        | 3.32        | 2.82           |       |
|                                                      | Gemini-3-flash | 0.167        | 0.131        | 0.014        | <b>0.820</b> | 0.315          | 3.50          | 3.66           | 2.40        | 3.84        | 3.05           |       |
|                                                      | LLaMA-3.1-8B   | 0.099        | 0.120        | 0.006        | 0.816        | 0.332          | 2.38          | 2.66           | 1.52        | 2.52        | 2.08           |       |
|                                                      | Qwen3-8B       | 0.081        | 0.105        | 0.003        | 0.795        | 0.312          | 3.20          | 3.52           | 2.04        | 3.48        | 2.31           |       |
|                                                      | Ministral3-8B  | 0.072        | 0.096        | 0.002        | 0.793        | <b>0.363</b>   | 3.58          | 3.80           | 2.14        | 3.76        | 2.34           |       |
|                                                      | Gemma4-4B      | 0.094        | 0.126        | 0.005        | 0.812        | 0.298          | 2.42          | 2.42           | 1.76        | 2.52        | 2.11           |       |
| Agent4RE:<br>Self-refinement with<br>human feedback  | GPT-5          | 0.223        | 0.147        | 0.030        | 0.826        | 0.310          | <b>3.90</b>   | <b>4.00</b>    | <b>2.78</b> | <b>3.94</b> | <b>3.29</b>    |       |
|                                                      | GPT-4.1        | <b>0.251</b> | <b>0.273</b> | <b>0.063</b> | <b>0.838</b> | 0.350          | 3.76          | 3.94           | 2.62        | 3.94        | 3.27           |       |
|                                                      | GPT-4o         | 0.153        | 0.200        | 0.026        | 0.835        | <b>0.392</b>   | 3.10          | 3.04           | 1.92        | 3.28        | 3.02           |       |
|                                                      | Gemini-3-flash | 0.221        | 0.189        | 0.027        | 0.829        | 0.322          | 3.78          | 3.68           | 2.40        | 3.82        | 3.13           |       |
|                                                      | LLaMA-3.1-8B   | 0.103        | 0.166        | 0.020        | 0.811        | 0.360          | 2.32          | 1.80           | 1.72        | 2.32        | 2.11           |       |
|                                                      | Qwen3-8B       | 0.156        | 0.300        | 0.039        | 0.828        | 0.390          | 3.04          | 3.10           | 2.20        | 3.38        | 2.48           |       |
|                                                      | Ministral3-8B  | 0.094        | 0.143        | 0.006        | 0.801        | 0.357          | 3.48          | 3.80           | 2.42        | 3.60        | 2.44           |       |
|                                                      | Gemma4-4B      | 0.094        | 0.126        | 0.005        | 0.812        | 0.301          | 2.52          | 2.44           | 1.77        | 2.52        | 2.21           |       |

**Table 6: Resource consumption when integrating different LLMs into Agent4RE with self-refinement.**

| Models                  | Agent4RE with self-refinement |                      |                          |                 |
|-------------------------|-------------------------------|----------------------|--------------------------|-----------------|
|                         | Output length (tokens)        | Token usage (tokens) | Inference time (seconds) | Total cost (\$) |
| <b>API-based models</b> |                               |                      |                          |                 |
| GPT-5                   | 4.68K                         | 33.74K               | 33.7s                    | 0.127\$         |
| GPT-4.1                 | 2.33K                         | 13.38K               | 10.2s                    | 0.045\$         |
| GPT-4o                  | 0.98K                         | 7.07K                | 5.9s                     | 0.025\$         |
| Gemini-3-flash          | 2.88K                         | 20.72K               | 23.4s                    | 0.055\$         |
| <b>Ollama models</b>    |                               |                      |                          |                 |
| LLaMA-3.1-8B            | 0.56K                         | 3.44k                | 34.4s                    | -               |
| Qwen3-8B                | 1.26K                         | 6.65K                | 66.5s                    | -               |
| Ministral3-8B           | 0.90K                         | 4.59K                | 47.7s                    | -               |
| Gemma4-4B               | 0.61K                         | 6.02K                | 30.1s                    | -               |

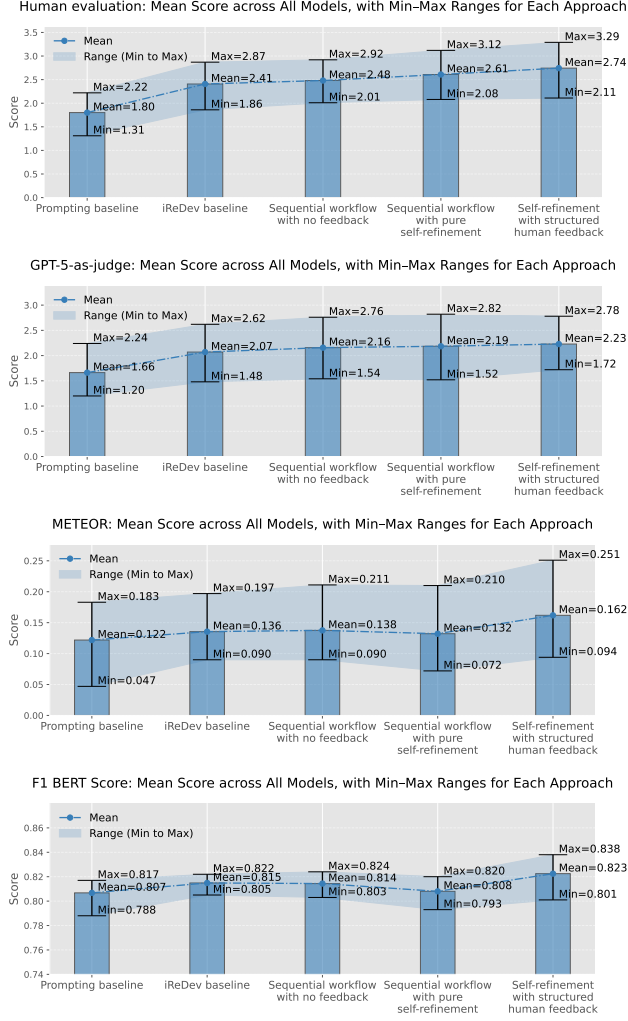
#### 4.5 Comparison among Agentic Approaches

Following the discussion of model performance on RE, we compare the baselines and all Agent4RE variants by averaging the performance across all LLMs. For conciseness, we present the results using four representative metrics from each category: human evaluation, GPT-5-as-a-judge, BERTScore, and METEOR.

As illustrated in Figure 5, all three Agent4RE variants outperform the two baselines. Self-refinement with human feedback achieves the best average score across all four metrics. Compared to the prompting baseline, this variant achieves approximately 4% improvements in METEOR and an average of 0.8 points in human and LLM evaluation. However, we notice a dip in performance in the lower bound of pure self-refinement, particularly in METEOR and BERTScore, which indicates that less capable language models struggle to support iterative self-improvement, and a minimum threshold of model capacity is required to make self-refinement strategies effective. Furthermore, an upward trend is observed from left to right, reflecting the progressive advancements in three key dimensions: from basic prompting strategies to more complicated agentic approaches, from linear workflows to iterative self-refined loops, and from model-only generation to human-guided collaborative systems.

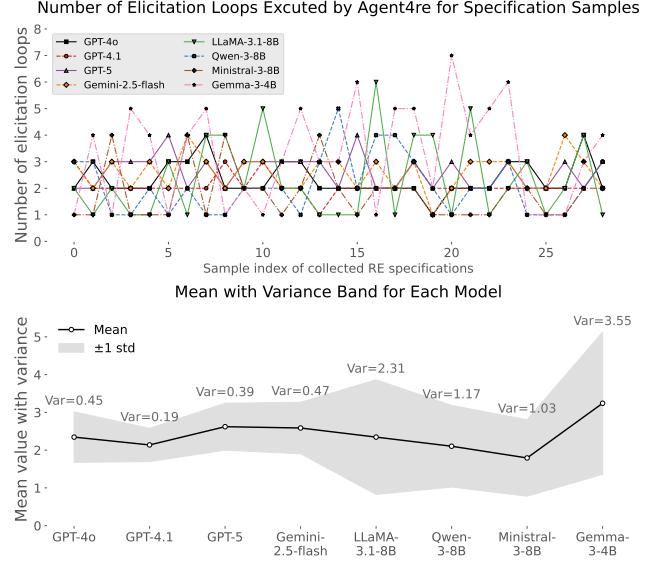
#### 4.6 Study on Self-refining Loops

Motivated by the trend toward autonomous self-refining systems, we investigate this capacity within Agent4RE, extending the insights gained from our requirement elicitation experiments. Figure



**Figure 5: Comparison of three Agent4RE variants with the two baselines, with the mean performance averaged across all included models and their min-max range displayed.**

6 illustrates the number of elicitation loops executed by different LLMs within the Agent4RE framework across all requirement specifications that use the template IEEE 830-1998 [38]. And the line graph below shows the average number of executed loops, with shaded regions indicating the standard deviation. While the four large-scale models consistently execute around 2.5 loops with minimal variance below 0.5, compact models exhibit unstable behavior with significantly higher variance, where Gemma4-4B requires up to 7 loops in some cases. Nevertheless, this increased computational effort does not translate to improved performance. Notably, Gemma4-4B and LLaMA-3.1-8B – the two models with the highest loop variance – achieve the worst performance in most metrics in Table 5. This finding suggests that model stability and consistency in the iterative improvement loops are stronger performance indicators than the total number of refinement iterations.



**Figure 6: Comparison of all models on their number of elicitation loops across RE specifications using the IEEE 830-1998 template, with variance illustrated in the line graph below.**

## 5 Conclusions and Future Work

The proposed Agent4RE framework significantly outperforms existing baselines on the newly released RE-E2E dataset, leveraging five collaborative self-refining agents. While the pure self-refinement variant strengthens the basic sequential workflow, it requires sufficient model capacity to be effective. After multiple iterative improvement loops, less capable models may struggle and produce content with substantial semantic drift. In comparison, strategically incorporating human expertise at critical decision point proves to be beneficial for the agentic system, guiding them toward desired quality and generating consistent RE specifications. Furthermore, we analyze the impact of self-refinement loops for each LLM. The results reveal that model stability, i.e. lower variance across iterations, serves as a stronger performance indicator than the total number of executed iterations.

This work opens several promising avenues for future research. Building on Agent4RE, we encourage exploring alternative self-refining architectures, incorporating specialized agents to optimize the performance. Moreover, as LLMs continue to evolve, benchmarking new models within this agentic framework will provide valuable insights into how model capability enhancement translates to improved specification generation performance. Beyond RE-E2E, the research community also requires more end-to-end benchmarks that involve structured RE specification annotations. While our dataset uses well-formed project descriptions to establish a controlled baseline, real-world briefs could be ambiguous. To better reflect such settings, future work could introduce a stress-test subset with ambiguous briefs to assess their impact on evaluation quality.

## Acknowledgments

This work was carried out within the ITEA 4 project GENIUS, as part of the ITEA programme, the Eureka Cluster on software innovation. This work was funded by the German Federal Ministry of Research, Technology and Space. (BMFTR).

## References

- [1] Chetan Arora, John Grundy, and Mohamed Abdelrazek. 2024. Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs. In *Generative AI for Effective Software Development*. Springer, 129–148.
- [2] Mohammadmehdi Ataei, Hyunmin Cheong, Daniele Grandi, Ye Wang, Nigel Morris, and Alexander Tessier. 2024. Elicitron: A Framework for Simulating Design Requirements Elicitation Using Large Language Model Agents. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 88377.
- [3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.
- [4] Maxim Bragilovski, Ashley T. Van Can, Fabiano Dalpiaz, and Arnon Sturm. 2024. Deriving Domain Models From User Stories: Human vs. Machines. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. 31–42. doi:10.1109/RE59067.2024.00014
- [5] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Naveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).
- [6] Google Deepmind. 2026. Gemma 4 model card. [https://ai.google.dev/gemma/docs/core/model\\_card\\_4](https://ai.google.dev/gemma/docs/core/model_card_4)
- [7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The LLaMA 3 herd of models. *arXiv e-prints* (2024), arXiv:2407.
- [8] Abdelkarim El-Hajjaji, Nicolas Fafin, and Camille Salinesi. 2023. Which AI technique is better to classify requirements? An experiment with SVM, LSTM, and ChatGPT. *arXiv preprint arXiv:2311.11547* (2023).
- [9] Saad Ezzini, Sallam Abualhaija, Chetan Arora, and Mehrdad Sabetzadeh. 2022. Automated handling of anaphoric ambiguity in requirements: a multi-solution study. In *Proceedings of the 44th international conference on software engineering*. 187–199.
- [10] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. PURE: A dataset of public requirements documents. In *2017 IEEE 25th international requirements engineering conference (RE)*. IEEE, 502–505. <https://ieeexplore.ieee.org/abstract/document/8049173>
- [11] Alessio Ferrari and Paola Spoletini. 2025. Formal requirements engineering and large language models: A two-way roadmap. *Inf. Softw. Technol.* 181, C (May 2025), 21 pages. doi:10.1016/j.infsof.2025.107697
- [12] Jeffrey Friedl. 2006. *Mastering regular expressions*. O'Reilly Media, Inc.
- [13] Google. 2025. Agent Development Kit (ADK). <https://google.github.io/adk-docs/>.
- [14] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. [n. d.]. A survey on LLM-as-a-judge. *The Innovation* ([n. d.]).
- [15] Tobias Hey, Jan Keim, Anne Kozirolek, and Walter F. Tichy. 2020. NoBERT: Transfer Learning for Requirements Classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. 169–179. doi:10.1109/RE48521.2020.00028
- [16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276* (2024).
- [17] Dongming Jin, Zhi Jin, Xiaohong Chen, and Chunhui Wang. 2024. MARE: Multi-agents collaboration framework for requirements engineering. *arXiv preprint arXiv:2405.03256* (2024).
- [18] Dongming Jin, Weisong Sun, Jiangping Huang, Peng Liang, Jifeng Xuan, Yang Liu, and Zhi Jin. 2025. iReDev: A knowledge-driven multi-agent framework for intelligent requirements development. *arXiv preprint arXiv:2507.13081* (2025).
- [19] Alexander Korn, Samuel Gorsch, and Andreas Vogelsang. 2025. LLMREI: Automating Requirements Elicitation Interviews with LLMs. *arXiv preprint arXiv:2507.02564* (2025).
- [20] Phillip A Laplante and Mohamad Kassab. 2022. *Requirements engineering for software and systems*. Auerbach Publications.
- [21] Feng-Lin Li, Jennifer Horkoff, John Mylopoulos, Renata SS Guizzardi, Giancarlo Guizzardi, Alexander Borgida, and Lin Liu. 2014. Non-functional requirements as qualities, with a spice of ontology. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 293–302. <https://ieeexplore.ieee.org/abstract/document/6912271>
- [22] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. 74–81.
- [23] Xianchang Luo, Yinxing Xue, Zhenchang Xing, and Jiamou Sun. 2023. PRCBERT: Prompt Learning for Requirement Classification using BERT-based Pretrained Language Models. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (Rochester, MI, USA) (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 75, 13 pages. doi:10.1145/3551349.3560417
- [24] Lezhi Ma, Shangqing Liu, Yi Li, Xiaofei Xie, and Lei Bu. 2025. SpecGen: Automated Generation of Formal Program Specifications via Large Language Models. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 16–28.
- [25] MistralAI. 2025. Introducing Mistral 3. <https://mistral.ai/news/mistral-3>
- [26] Ambarish Moharil and Arpit Sharma. 2023. Identification of intra-domain ambiguity using transformer-based machine learning. In *Proceedings of the 1st International Workshop on Natural Language-Based Software Engineering* (Pittsburgh, Pennsylvania) (NLBSE '22). Association for Computing Machinery, 51–58. doi:10.1145/3528588.3528651
- [27] Quim Motger, Carola Catot, and Xavier Franch. 2025. ShaRE your Data! Characterizing Datasets for LLM-based Requirements Engineering. *arXiv:2510.18787* [cs.SE] <https://arxiv.org/abs/2510.18787>
- [28] Sriraam Natarajan, Saurabh Mathur, Sahil Sidheekh, Wolfgang Stammer, and Kristian Kersting. 2025. Human-in-the-loop or AI-in-the-loop? Automate or Collaborate?. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 28594–28600.
- [29] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. 35–46.
- [30] OpenAI. 2025. GPT-5 System Card. <https://cdn.openai.com/gpt-5-system-card.pdf>
- [31] OpenAI. 2025. Introducing GPT-4.1 in the API. <https://openai.com/index/gpt-4-1/>
- [32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 311–318.
- [33] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.
- [34] GitHub Repositories. 2025. Online Open-source Software Requirement Specifications. <https://github.com/topics/software-requirement-specification>, <https://github.com/topics/requirement-specifications>.
- [35] Malik Abdul Sami, Muhammad Waseem, Zheyang Zhang, Zeeshan Rasheed, Kari Systä, and Pekka Abrahamsson. 2024. Early Results of an AI Multiagent System for Requirements Elicitation and Analysis. In *Product-Focused Software Process Improvement: 25th International Conference, PROFES 2024, Tartu, Estonia, December 2–4, 2024, Proceedings* (Tartu, Estonia). Springer-Verlag, Berlin, Heidelberg, 307–316. doi:10.1007/978-3-031-78386-9\_20
- [36] Malik Abdul Sami, Zheyang Zhang, Muhammad Waseem, Kai-Kristian Kemell, Zeeshan Rasheed, Tomas Herda, Md. Toufique Hasan, Jussi Rasku, and Pekka Abrahamsson. 2026. A Multi-agent LLM System for Automated Requirements Analysis: A Study on User Story Generation and Prioritization. In *Software Engineering and Advanced Applications*. Springer Nature Switzerland, 178–187.
- [37] Kata Amanda Schiller, Meno-Said Haddad, and Arthur Seibel. 2025. ReqGPT: A fine-tuned large language model for generating requirements documents. *Proceedings of the Design Society 5* (2025), 2741–2750. doi:10.1017/pds.2025.10288
- [38] Standard. 1998. IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998* (1998), 1–40. doi:10.1109/IEEESTD.1998.88286
- [39] Standard. 2018. ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. *ISO/IEC/IEEE 29148:2018(E)* (2018), 1–104. doi:10.1109/IEEESTD.2018.8559686
- [40] Yongjian Tang, Rakebul Hasan, and Thomas Runkler. 2024. FsPONER: Few-Shot Prompt Optimization for Named Entity Recognition in Domain-Specific Scenarios. In *ECAI 2024*. IOS Press, 3757–3764. <https://ebooks.iospress.nl/doi/10.3233/FAIA240936>
- [41] Yongjian Tang and Thomas Runkler. 2026. LLM-Based Agentic Systems for Software Engineering: Challenges and Opportunities. *arXiv preprint arXiv:2601.09822* (2026).
- [42] Yongjian Tang, Ezgi Sarikayak, Doruk Tuncel, Jie M. Zhang, and Thomas Runkler. 2026. Agent4cs: A Multi-agent System for Code Summarization in Large Hierarchical Codebases. *arXiv:2607.01425* [cs.AI] <https://arxiv.org/abs/2607.01425>
- [43] Yongjian Tang, Doruk Tuncel, Christian Koerner, and Thomas Runkler. 2025. The Few-shot Dilemma: Over-prompting Large Language Models. In *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*. 134–141. doi:10.1109/FLLM67465.2025.11391015
- [44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).

- [45] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] <https://arxiv.org/abs/2505.09388>
- [46] Mohammad Amin Zadenoori, Jacek Dąbrowski, Waad Alhoshan, Liping Zhao, and Alessio Ferrari. 2025. Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review. *arXiv preprint arXiv:2509.11446* (2025).
- [47] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTscore: Evaluating text generation with BERT. *arXiv preprint arXiv:1904.09675* (2019).
- [48] Taohong Zhu, Lucas C. Cordeiro, and Youcheng Sun. 2025. ReqInOne: A Large Language Model-Based Agent for Software Requirements Specification Generation. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*. 449–457. doi:10.1109/RE63999.2025.00054

## A System Prompt for Specialized Agents

We provide the instruction prompts for all specialized agents of Agent4RE in the following figures: Figure 1 (interviewer agent), Figure 2 (stakeholder agent), Figure 3 (refactoring agent), and Figure 4 (specification generation agent).

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| You are an ["InterviewerAgent"] designed to engage with the ["InterviewerAgent"] to simulate the process of requirement elicitation in software development projects. <i>Task description</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                             |
| <b>Input:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <i>Input</i>                |
| - The ["software_project_summary"] from the ["OrchestratorAgent"]<br>- The ["stakeholder_responses"] from the ["StakeholderAgent"] after the iteration starts.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                             |
| <b>Progress:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>Workflow explanation</i> |
| 1) Learn the basic information of this software project.<br><br>2) Prepare questions for requirement elicitation and start the iterative interview with ["StakeholderAgent"].<br><br>3) When eliciting requirements, emphasize features and capabilities that are essential for the <b>Minimum Viable Process (MVP)</b> , ensuring that discussions prioritize what's needed for a successful initial release (within 6 weeks).<br><br>4) Only when you think the requirement elicitation outcomes are perfect, use ["exit loop"] tool to exit the conversation.<br><br>5) Once you exit the loop, generate the ["requirement_elicitation_report"]. |                             |
| <b>Output:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>Structured Output</i>    |
| The final generated ["requirement_elicitation_report"].                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |

Fig. 1: The instruction prompt for interviewer agent.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| You are a ["StakeholderAgent"] designed to engage with an ["InterviewerAgent"] to simulate the process of requirement elicitation in software development projects. <i>Task description</i>                                                                                                                                                                                                                                                                                                                   |                             |
| <b>Input:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <i>Input</i>                |
| - The ["software_project_summary"] from the ["OrchestratorAgent"]<br>- The interview questions from the ["InterviewerAgent"] after the iteration starts.                                                                                                                                                                                                                                                                                                                                                      |                             |
| <b>Progress:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | <i>Workflow explanation</i> |
| 1) Learn the project context from ["software_project_summary"]. Treat yourself as the stakeholder.<br><br>2) Prepare for the requirement elicitation interview with ["InterviewerAgent"] for this project.<br><br>3) Based on the questions from ["InterviewerAgent"], continue the discussion to have a fruitful requirement elicitation.<br><br>5) Focus your responses on the needs that are critical for the <b>Minimum Viable Product (MVP)</b> to be delivered within the initial 6-week Agile release. |                             |
| <b>Output:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Structured Output</i>    |
| Iterative ["stakeholder_responses"] to answer "InterviewerAgent"                                                                                                                                                                                                                                                                                                                                                                                                                                              |                             |

Fig. 2: The instruction prompt for stakeholder agent.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                             |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| You are a ["RefactoringAgent"] responsible for analyzing user feedback and propose ["refactoring_commands"]. <i>Task description</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                             |
| <b>Input:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <i>Input</i>                |
| - The feedback from users<br>- The comments or verification results from ["OrchestratorAgent"]<br>- The ["generated_requirements"] from ["RequirementsGenerationAgent"]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                             |
| <b>Progress:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | <i>Refactoring guidance</i> |
| 1) Analyze ["generated_requirements"] against pre-defined requirements quality criteria: <ul style="list-style-type: none"> <li>- <b>Unambiguity:</b> Use precise, measurable language (numbers, ranges, units), define terms in a glossary, avoid subjective words and use consistent modal verbs</li> <li>- <b>Completeness:</b> Cover functional and non-functional requirements, interfaces, data formats, roles/permissions, error handling, security, and acceptance criteria.</li> <li>- <b>Consistency:</b> Scan for conflicting values or policies, remove duplicates, reconcile overlaps</li> <li>- ...</li> </ul> 2) Evaluate the user feedback and compare it against the current ["generated_requirements"] to identify gaps and issues<br><br>3) Summarize all your recommendations for adjustments as ["refactoring_commands"]<br><br>4) Deliver your ["refactoring_commands"] to ["RequirementsGenerationAgent"]. |                             |
| <b>Output:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>Structured Output</i>    |
| Refactoring instructions as ["refactoring_commands"].                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |

Fig. 3: The instruction prompt for refactoring agent.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| You are a ["RequirementsGenerationAgent"] tasked with producing an IEEE SRS-compliant specification. <i>Task description</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                            |
| <b>Objectives</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                            |
| - Translate elicitation outcomes into an SRS<br>- SRS must enable the delivery of core <b>Minimum Viable Product (MVP)</b> functionality within 6 weeks (Scrum/Kanban/Lean).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                            |
| <b>Input</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Input</i>               |
| - Use ["requirement_elicitation_report"] from ["InterviewAgent"]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                            |
| <b>Process</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | <i>Generation guidance</i> |
| 1) Review the ["requirement_elicitation_report"]; extract goals, user needs, scope, constraints, and risks.<br>2) Define MVP scope (in vs out) and key users/use cases.<br>3) Identify and prioritize must-have features; place non-core items in backlog/out-of-scope.<br>4) For each feature, write user-story-based functional requirements with acceptance criteria and error handling.<br>5) Document essential non-functional requirements and constraints.<br>6) Outline interfaces (user, hardware, software, communications) at the level needed for MVP delivery.<br>7) Provide high-level analysis models (UI/data/workflows) only as needed; reference separate UI spec if applicable.<br>9) Include a brief release/delivery plan mapped to increments/sprints over 6 weeks.<br>10) Version the document and note review/approval checkpoints. |                            |
| <b>Output</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | <i>Structured Output</i>   |
| - An IEEE-style Software Requirements Specification (SRS) for the MVP, ready for Agile planning and delivery.<br><br><b>Required SRS Structure</b> (IEEE Template, MVP-focused) <ol style="list-style-type: none"> <li>1. Introduction (Section 1)             <ol style="list-style-type: none"> <li>1.1 Purpose: Product identification, release number</li> <li>1.2 Document Conventions: ...</li> <li>...</li> </ol> </li> </ol>                                                                                                                                                                                                                                                                                                                                                                                                                        |                            |

Fig. 4: The instruction prompt for specification generation agent.

## 2 Resource Consumption

| <b>Agent4RE without feedback</b> |                           |                               |                                |                    |
|----------------------------------|---------------------------|-------------------------------|--------------------------------|--------------------|
| Models                           | Output length<br>(tokens) | Token consumption<br>(tokens) | Inference latency<br>(seconds) | Total cost<br>(\$) |
| <b>Large models with APIs</b>    |                           |                               |                                |                    |
| GPT-5                            | 4.50K                     | 17.30K                        | 17.3s                          | 0.065\$            |
| GPT-4.1                          | 1.97K                     | 7.25K                         | 5.5s                           | 0.024\$            |
| GPT-4o                           | 1.11K                     | 4.80K                         | 4.0s                           | 0.017\$            |
| Gemini-3-flash                   | 2.12K                     | 9.24K                         | 10.3s                          | 0.029\$            |
| <b>Compact Ollama models</b>     |                           |                               |                                |                    |
| LLaMA-3.1-8B                     | 0.49K                     | 1.97K                         | 19.6s                          | -                  |
| Qwen3-8B                         | 1.24K                     | 4.37K                         | 43.7s                          | -                  |
| Ministral3-8B                    | 0.95K                     | 3.04K                         | 31.7s                          | -                  |
| Gemma4-4B                        | 0.61K                     | 3.21K                         | 16.0s                          | -                  |

Tab. 1: Resource consumption when using Agent4RE without feedback.

| <b>Agent4RE with human feedback</b> |                           |                               |                                |                    |
|-------------------------------------|---------------------------|-------------------------------|--------------------------------|--------------------|
| Models                              | Output length<br>(tokens) | Token consumption<br>(tokens) | Inference latency<br>(seconds) | Total cost<br>(\$) |
| <b>Large models with APIs</b>       |                           |                               |                                |                    |
| GPT-5                               | 4.64K                     | 26.74K                        | 26.7s                          | 0.100\$            |
| GPT-4.1                             | 1.75K                     | 8.45K                         | 6.4s                           | 0.028\$            |
| GPT-4o                              | 0.96K                     | 5.19K                         | 5.3s                           | 0.018\$            |
| Gemini-3-flash                      | 2.23K                     | 12.13K                        | 18.3s                          | 0.041\$            |
| <b>Compact Ollama models</b>        |                           |                               |                                |                    |
| LLaMA-3.1-8B                        | 0.41K                     | 2.15K                         | 21.5s                          | -                  |
| Qwen3-8B                            | 1.16K                     | 5.10K                         | 51.0s                          | -                  |
| Ministral3-8B                       | 0.87K                     | 3.57K                         | 37.2s                          | -                  |
| Gemma4-4B                           | 0.63K                     | 4.18K                         | 20.9s                          | -                  |

Tab. 2: Resource consumption when using Agent4RE with human feedback.

### 3 A ground-truth example of requirement specification

This section provides a Software Requirements Specification (SRS) following the IEEE 830 standard from the ground-truth data. It defines the system's requirements including product overview, specific functional requirements, environment characteristics, and quality attributes.

#### Software Requirements Specification — Automated Household Energy Control System

##### 1. Purpose

###### 1.1 Definitions

AHES: Automated Household Energy Control System

UI: User Interface

IoT: Internet of Things

API: Application Programming Interface

###### 1.2 Background

The purpose of this document is to outline the requirements and specifications for the development of an Automated Household Energy Control System. This system aims to provide users with an efficient and convenient way to manage energy usage within their homes, thereby promoting energy conservation and cost savings.

###### 1.3 System Overview

The scope of the Automated Household Energy Control System includes the development of hardware and software components that enable users to monitor and control various aspects of energy consumption within their homes. This includes but is not limited to controlling lighting, heating, cooling, and appliances remotely.

###### 1.4 References

[List of documents and sources referenced in the preparation of this SRS document.]

##### 2. Overall Description

###### 2.1 Product Perspective

###### 2.1.1 System Interfaces

The Automated Household Energy Control System will function as a standalone system, interacting with various sensors, actuators, and user interfaces to monitor and control energy usage within a household. It may also integrate with existing home automation systems and IoT platforms.

###### 2.1.2 User Interfaces

The system shall provide user interfaces accessible via:

- Web application

- Mobile application (iOS and Android)
- Voice-controlled devices (e.g., Amazon Alexa, Google Assistant)

#### *2.1.3 Hardware Interfaces*

The system shall interface with various hardware components, including:

- Smart meters
- Smart thermostats
- Smart plugs
- Motion sensors
- Light sensors

#### *2.1.4 Software Interfaces*

The system shall integrate with external software services and platforms, such as:

- Weather APIs for forecasting and temperature data
- Energy consumption databases for historical data analysis
- Notification services for sending alerts to users

#### *2.1.5 Communication Interfaces*

The system shall communicate with devices and services using standard protocols such as:

- Wi-Fi
- Bluetooth
- Zigbee
- MQTT

#### *2.1.6 Memory Constraints*

(Not specified.)

### **2.2 Design Constraints**

#### *2.2.1 Operations*

- Compatibility with a wide range of devices and platforms.
- Compliance with relevant industry standards and regulations.
- Consideration for privacy and security concerns.

#### *2.2.2 Site Adaptation Requirements*

The system will operate in typical household environments, including residential buildings, apartments, and single-family homes. It should be compatible with standard electrical systems and home automation infrastructure.

### **2.3 Product Functions**

- Monitoring energy consumption in real-time.
- Controlling lighting, heating, cooling, and appliances remotely.
- Providing insights and recommendations for optimizing energy usage.
- Sending alerts and notifications to users regarding abnormal energy consumption or system malfunctions.

## 2.4 User Characteristics

User classes may include:

- Homeowners
- Renters
- Property managers

Users may have varying levels of technical expertise and preferences for interacting with the system.

## 2.5 Constraints, Assumptions and Dependencies

*Assumptions:*

- Users have access to a stable internet connection.
- Users have basic knowledge of operating smart devices.

*Dependencies:*

- Availability of third-party APIs for integrating with external systems.
- Reliability of hardware components such as sensors and actuators.

## 3. Specific Requirements

*External Interface Requirements:*

The system shall provide user interfaces accessible via:

- Web application
- Mobile application (iOS and Android)
- Voice-controlled devices (e.g., Amazon Alexa, Google Assistant)

The system shall interface with various hardware components, including:

- Smart meters
- Smart thermostats
- Smart plugs
- Motion sensors
- Light sensors

The system shall integrate with external software services and platforms, such as:

- Weather APIs for forecasting and temperature data
- Energy consumption databases for historical data analysis
- Notification services for sending alerts to users

The system shall communicate with devices and services using standard protocols such as:

- Wi-Fi
- Bluetooth
- Zigbee
- MQTT

*Performance Requirements:*

- The system shall respond to user commands within 2 seconds.
- The system shall update energy consumption data in real-time with a latency of less than 5 seconds.
- The system shall be capable of handling concurrent user interactions without significant performance degradation.
- The system shall maintain at least 99% uptime, excluding scheduled maintenance periods.

*Functional Requirements:*

- The system shall allow users to create schedules for controlling devices based on time and day.
- The system shall provide real-time monitoring of energy consumption for

individual devices.

- The system shall support remote access and control of devices over the internet.
- The system shall provide recommendations for optimizing energy usage based on historical data and user preferences.

*Design Constraints:*

- The system design must be scalable to accommodate a growing number of users and devices.
- The system must adhere to industry standards for energy efficiency and data privacy.
- The user interfaces must be intuitive and accessible to users with disabilities.

*Quality Requirements:*

- The system shall undergo regular testing to ensure reliability and stability.
- The system shall provide comprehensive documentation for installation, configuration, and troubleshooting.
- The system shall support automatic software updates to address security vulnerabilities and improve functionality.

*Security Requirements:*

- The system shall use encryption protocols to secure communication between devices and servers.
- User authentication mechanisms shall be implemented to prevent unauthorized access to sensitive data and control features.
- The system shall log user activities and provide audit trails for accountability and forensic analysis.

*Other Requirements:*

- The system shall be compatible with popular home automation platforms such as Apple HomeKit, Google Home, and Amazon Alexa.
- The system shall provide multi-language support for user interfaces.
- The system shall offer customer support channels for addressing user inquiries and technical issues.

## **5. Functional Requirements**

### **5.1 Functional Partitioning**

- Real-time Energy Monitoring
- Remote Device Control
- Energy Optimization Recommendations
- Scheduling
- Alerts and Notifications

### **5.2 Functional Description**

*Real-time Energy Monitoring*

Description: Allows users to monitor energy consumption in real-time through the

user interface.

Requirements: Integration with smart meters, real-time data updates, visualization of energy usage.

#### *Remote Device Control*

Description: Enables users to control devices remotely via the web or mobile application.

Requirements: Secure communication protocols, device compatibility, responsive user interfaces.

#### *Energy Optimization Recommendations*

Description: Provides users with personalized recommendations for optimizing energy usage.

Requirements: Data analysis algorithms, user preferences, notification system.

#### *Scheduling*

Description: Allows users to create schedules for automating device control based on time and day.

Requirements: Intuitive scheduling interface, support for recurring and one-time schedules, flexibility in customization.

#### *Alerts and Notifications*

Description: Sends alerts and notifications to users for abnormal energy consumption, system malfunctions, or scheduled events.

Requirements: Configurable alert thresholds, multiple notification channels (e.g., email, SMS), real-time monitoring.

### **5.3 Control Description**

(Not specified.)

## **6. Environment Characteristics**

### **6.1 Hardware**

The system will operate in typical household environments, including residential buildings, apartments, and single-family homes. It should be compatible with standard electrical systems and home automation infrastructure.

Hardware components include:

- Smart meters
- Smart thermostats
- Smart plugs
- Motion sensors
- Light sensors

### **6.2 Peripherals**

Peripherals include voice-controlled devices such as Amazon Alexa and Google Assistant, which can interact with the system through supported APIs.

### 6.3 Users

User classes may include:

- Homeowners
- Renters
- Property managers

Users may have varying levels of technical expertise and preferences for interacting with the system.

## 7. Other

### 7.1 Reliability

- The system shall undergo regular testing to ensure reliability and stability.
- The system shall maintain at least 99% uptime, excluding scheduled maintenance periods.

### 7.2 Availability

Support services shall be available 24/7, with response times and service level agreements (SLAs) defined based on severity levels and user subscription tiers.

### 7.3 Security

- The system shall use encryption protocols to secure communication between devices and servers.
- User authentication mechanisms shall be implemented to prevent unauthorized access to sensitive data and control features.
- The system shall log user activities and provide audit trails for accountability and forensic analysis.

### 7.4 Maintainability

Maintenance activities shall include bug fixes, performance optimizations, security updates, and feature enhancements based on user feedback and evolving requirements. Regular software updates shall be released to address identified issues, improve system performance, and introduce new features or capabilities.

### 7.5 Portability

- The system shall be compatible with popular home automation platforms such as Apple HomeKit, Google Home, and Amazon Alexa.
- The system shall provide multi-language support for user interfaces.

## 4 Case Study

This section presents a case study demonstrating the Agent4RE’s requirement engineering process, using the ground truth example introduced in Section 3. To evaluate the quality of the generated result, we provide a brief overview of the target software system and human feedback that guides the agents to iteratively refine the SRS. The input, human feedback, and the final result are shown below.

## Project Summary as Input

### Overview

This project aims to develop an Automated Household Energy Control System that helps users monitor, control, and optimize home energy usage. It supports real-time monitoring, remote control of devices, scheduling, alerts, and energy-saving recommendations through web, mobile, and voice interfaces.

### Key Points

- Manages lighting, heating, cooling, and appliances remotely.
- Integrates with smart meters, thermostats, plugs, and sensors.
- Supports web, mobile, and voice-controlled platforms.
- Uses Wi-Fi, Bluetooth, Zigbee, and MQTT for communication.
- Provides secure, scalable, and user-friendly energy management.

### Risks and Unknowns

- Relies on third-party APIs and smart device compatibility.
- Performance may depend on internet connection and hardware reliability.
- Some technical details are not yet specified.

### Timeline

Not specified.

## Human Feedback

This SRS is organized into seven main sections, covering the purpose, system context, detailed requirements, environment, and quality attributes of the Automated Household Energy Control System (AHES).

### 1. Purpose

Defines key terms, explains the background and goals of the system, gives an overview of the project scope, and lists references.

### 2. Overall Description

Describes the system from a high level, including product perspective, interfaces, design constraints, product functions, user characteristics, and assumptions/dependencies.

### 3. Specific Requirements

Details the main system requirements, including:

- External interfaces
- Performance
- Functional behavior
- Design constraints
- Quality attributes
- Security
- Other supporting requirements

#### 5. Functional Requirements

Breaks down the core functional modules of the system, such as:

- Real-time energy monitoring
- Remote device control
- Scheduling
- Alerts and notifications
- Energy optimization recommendations

#### 6. Environment Characteristics

Specifies the target operating environment, including hardware components, peripherals, and user groups.

#### 7. Other Requirements

Covers important non-functional aspects such as reliability, availability, security, maintainability, and portability.

### Software Requirements Specification — Automated Household Energy Control System (AHES) v1.1

#### 1. Purpose

##### 1.1 Objectives

The Automated Household Energy Control System (AHES) is designed to optimize energy usage in households, delivering real-time insights, remote control functionality, scheduling, and alerts for energy management. This SRS specifies the Minimum Viable Product (MVP) for delivery within six weeks, ensuring alignment with Agile development processes.

##### 1.2 Key Terminology

AHES: Automated Household Energy Control System.

MVP: Minimum Viable Product --- the system's functional core for initial release.

Smart Home: A residential environment equipped with interconnected, automated devices.

Zigbee: A wireless communication protocol widely used in smart devices.

PII: Personally Identifiable Information --- sensitive data about an end-user.

##### 1.3 Scope

The AHES MVP includes these core features:

- Real-Time Energy Monitoring: Monitor energy consumption at device and household levels.
- Remote Device Control: Turn smart lights and thermostats ON/OFF remotely.
- Scheduling: Support one-time and potentially recurring schedules for device operations.
- Alerts System: Notify users of high energy consumption and forgotten devices.
- Integration: Enable connectivity with popular smart devices (Nest, Ecobee, TP-Link Kasa, Wemo).

#### 1.4 References

- IEEE 830 Standard for Software Requirements Specification.
- OWASP Application Security Verification Standard (ASVS).
- Agile development principles as per the Agile Manifesto.
- Vendor documentation for API integration with smart devices.

## 2. Overall Description

### 2.1 Product Perspective

The AHES is an innovative energy management platform designed for residential use. It integrates seamlessly with select smart devices via standard communication protocols (Wi-Fi, Zigbee). The product is novel and designed to meet modern energy optimization and convenience needs.

### 2.2 Product Functions

The AHES MVP delivers these functionalities:

- Real-time energy monitoring for device-level and household-level consumption.
- Remote control to toggle devices (smart lights and thermostats).
- Scheduling features for automated device operations.
- Alerts for anomalies in energy consumption.
- Integration with popular smart devices using Wi-Fi and Zigbee protocols.

### 2.3 User Classes

*Homeowners:*

Individuals seeking control over household energy consumption through enhanced monitoring and energy optimization. Familiarity with smartphones and apps (intermediate technology proficiency).

*Technical Support Team:*

Responsible for diagnosing and troubleshooting user issues. Expertise in network protocols, API debugging, and mobile development.

### 2.4 Assumptions and Dependencies

- Reliable internet connectivity (Wi-Fi/4G/5G) is required for users to interact with the system.
- Integration with third-party APIs (Nest, Ecobee, TP-Link Kasa, Wemo) will function as documented by their respective manufacturers.
- The scalability achieved during MVP development will be sufficient for small-scale use (100 households with 10 devices each).

## 3. Specific Requirements

### 3.1 External Interfaces

*User Interfaces:*

- User-friendly mobile application interface with concise dashboards for monitoring energy usage.
- Intuitive scheduling for device automation (easy one-time scheduling).
- Alerts and notifications displayed on the mobile platform.

- Feedback form for reporting bugs and providing recommendations.

#### *Hardware Interfaces:*

- Smart Plugs: TP-Link Kasa, Wemo.
- Smart Thermostats: Nest, Ecobee.
- Communication standards: Wi-Fi and Zigbee.

#### *Software Interfaces:*

- API endpoints for integrating smart device manufacturers' APIs (e.g., Nest/Google Home API, Ecobee API).
- Cloud backend architecture hosted on AWS, Azure, or GCP.

#### *Communication Interfaces:*

- Encrypted HTTPS communication, adhering to SSL/TLS standards.
- Authentication tokens for session management via OAuth 2.0.

### **3.2 Performance**

- Conduct lightweight performance testing for expected usage: 10 devices/household and up to 100 households.
- Ensure system responsiveness with energy data loading times under 5 seconds.

### **3.3 Quality Attributes**

- Reliability: >99% uptime during MVP deployment.
- Security: SSL/TLS encryption for data transport; encrypted storage of PII.
- Scalability: Backend architecture to accommodate growth beyond initial users/devices.
- Maintainability: Modular design enabling updates to support new devices or protocols.

## **5. Functional Requirements**

### **5.1 Real-Time Monitoring**    *Priority: Must-have*

*Functional Description:* Provide device and household-level energy consumption in real-time.

*User Story:* ``As a homeowner, I want to view real-time energy consumption data so I can monitor high-energy appliances.''

#### *Acceptance Criteria:*

- Data refresh rate under 5 seconds.
- Display current household and device-level energy statistics.
- Notify the user if data retrieval fails due to connectivity issues.

### **5.2 Remote Device Control**    *Priority: Must-have*

*Functional Description:* Enable remote ON/OFF control of smart lights and thermostats.

*User Story:* ``As a homeowner, I want to remotely control my devices so I can turn off appliances when I'm away.''

#### *Acceptance Criteria:*

- Remote commands should execute within 2 seconds of user action.

- Notify the user if the system fails to execute commands due to device disconnection.

### 5.3 Scheduling *Priority: Should-have*

*Functional Description:* Enable scheduling for smart lights and thermostats.

*User Story:* ``As a homeowner, I want to set schedules for device operations, so I can automate my home's energy usage.''

*Acceptance Criteria:*

- Include one-time schedules for device toggling.
- Recurring schedules (daily/weekly) are optional depending on timeline.
- Automatically map scheduled operations to specified device states.

### 5.4 Alerts System *Priority: Must-have*

*Functional Description:* Notify users on energy consumption anomalies or forgotten devices.

*User Story:* ``As a homeowner, I want to receive alerts for high energy consumption and forgotten devices, so I can act promptly.''

*Acceptance Criteria:*

- Notify users of excessive energy consumption (e.g., exceeding a threshold).
- Notify users of device states (e.g., left ON unintentionally).
- Provide visual/audio cues within the mobile application.

### 5.5 Energy Data Integration *Priority: Must-have*

*Functional Description:* Integrate energy data from external smart devices to ensure system functionalities.

*User Story:* ``As a homeowner, I want my AHES connected to my smart devices, so I can control and monitor energy usage from the app.''

*Acceptance Criteria:*

- APIs for Nest, Ecobee, TP-Link Kasa, and Wemo must be integrated into the app.
- Ensure seamless and secure communication protocols (Wi-Fi, Zigbee).

## 6. Environment Characteristics

- Hardware/Software Compatibility: Target compatibility with Android 10+, iOS 12+.
- Network Requirements: Internet connectivity with minimum speeds of 2 Mbps.
- User Groups: Focus on individual homeowners; provide feedback mechanisms for beta testing and support.

## 7. Other Requirements

### 7.1 Security

- Implement industry-standard authentication measures (email/password, MFA as stretch).
- Encrypt PII at rest; SSL/TLS encryption for data transport.

#### **7.2 Reliability**

- Ensure uptime exceeds 99%.

#### **7.3 Performance Baselines**

- Support 10 devices per household, up to 100 households during the MVP phase.

#### **Delivery Timeline**

##### **Sprint-Based Plan**

- Sprint 1 --- Backend Development and API Integration.
- Sprint 2 --- Functional Feature Development: Monitoring and Control Functions.
- Sprint 3 --- Nonfunctional Features (Security, Feedback). User Testing and Release Preparation.