MARE: Multi-Agents Collaboration Framework for Requirements Engineering

Dongming Jin[1,2], Zhi Jin[1,2], Xiaohong Chen[3], Chunhui Wang[4]

[1] School of Computer Science, Peking University, China
[2] Key Lab of High-Confidence of Software Technologies (PKU), Ministry of Education, China
[3] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, China
[4] College of Computer Science and Technology, Inner Mongolia Normal University, China
dmjin@stu.pku.edu.cn, zhijin@pku.edu.cn

Abstract—Requirements Engineering (RE) is a critical phase in the software development process that generates requirements specifications from stakeholders' needs. Recently, deep learning techniques have been successful in several RE tasks. However, obtaining high-quality requirements specifications requires collaboration across multiple tasks and roles. In this paper, we propose an innovative framework called *MARE*, which leverages collaboration among large language models (LLMs) throughout the entire RE process. *MARE* divides the RE process into four tasks: elicitation, modeling, verification, and specification. Each task is conducted by engaging one or two specific agents and each agent can conduct several actions. *MARE* has five agents and nine actions. To facilitate collaboration between agents, *MARE* has designed a workspace for agents to upload their generated intermediate requirements artifacts and obtain the information they need. We conduct experiments on five public cases, one dataset, and four new cases created by this work. We compared *MARE* with three baselines using three widely used metrics for the generated requirements models. Experimental results show that *MARE* can generate more correct requirements models and outperform the state-of-the-art approaches by 15.4%. For the generated requirements specifications, we conduct a human evaluation in three aspects and provide insights about the quality.

Index Terms—Requirements Engineering; Large Language Model; Deep Learning

I. INTRODUCTION

Requirements Engineering (RE) is one of the most critical phases in the software development process. It aims to generate a well-defined set of requirements by eliciting, analyzing, specifying, and verifying the needs and expectations of stakeholders for a software system [1]. As the complexity and scale of the software system [2] continue to grow, developers will spend a lot of effort on various tasks in RE to create high-quality requirements models and write requirements specifications.

Nowadays, deep learning (DL) techniques have been successfully applied to automate various RE tasks [3]. For example, DL has been used to mine stakeholders' needs [4] [5], extract requirements model [6] [7], and deal with ambiguity in requirements [8]. However, to obtain high-quality requirements specifications, different RE tasks need to be interwoven and iterated. The automation of only a few of these tasks limits further increase in effectiveness and efficiency.

Recently, large language models (LLMs) have achieved significant performance in the fields of software engineering (SE) [9] and natural language processing (NLP) [10]. There have also been some early explorations of the application of LLMs to RE [11] [12]. For example, [13] used LLMs to generate requirements elicitation interview scripts. [14] used ChatGPT to detect inconsistency in natural language requirements. [15] provided prompt strategies for requirements traceability. These efforts demonstrate the effectiveness of using LLMs for specific requirements tasks and do not take full advantage of LLMs for collaboration among RE tasks.

Inspired by the multi-agents collaboration [16] and team design, researchers started to investigate the possibility of designing the communicative agents mechanisms [17] for facilitating the accomplishment of complex tasks. [18] presented a virtual chat-powered software development company to unveil fresh possibilities for integrating LLMs into the realm of software development. [19] proposed an meta-programming framework incorporating efficient human workflows into LLM-based multi-agent collaborations, allowing agents with human-like domain expertise to verify intermediate results and reduce errors. Studies show that multi-agent collaboration mechanisms are interrelated with specific tasks.

Along this line, in this paper, we propose a novel framework, named *MARE* (**M**ulti-**A**gent collaboration for **R**equirements **E**ngineering) to automate the entire RE process based on collaboration between multiple tasks and roles. Specifically, given a very rough idea of requirements, *MARE* autonomously and iteratively performs the basic tasks of RE, i.e., elicitation, modeling, verification, and specification, to achieve end-to-end requirements specifications generation. Each task is accomplished with one or two specific agents and *MARE* contains five agents, i.e., the stakeholders, the collector, the modeler, the checker, and the documenter. Each agent is responsible for a requirements task and performs predefined actions to accomplish that task. Nine actions are designed for this purpose. *MARE* also offers a shared workspace to these agents for supporting the collaboration among agents.

We conduct extensive experiments to evaluate *MARE*. (1) To evaluate its modeling ability, we compare *MARE* with three baselines for automated requirements modeling on five public cases, one public dataset, and four new cases created in this paper. The three baselines are state-of-the-art (SOTA) methods for use case diagrams, problem diagrams, and goal

models, respectively. Three widely used evaluation metrics (Precision(P), Recall(R), and F1) are used for the evaluation. Results demonstrate the impressive performance of *MARE* on requirements modeling. In terms of F1, *MARE(gpt-3.5-turbo)* outperforms the three SOTA baselines by up to 15.4%, 23.9%, and 0.6%, respectively. (2) To evaluate its specification generation ability, we conduct a human evaluation to evaluate the quality of generated requirements specifications in three aspects (correctness, completeness, and consistency) and provide insights about their quality. (3) We conduct an ablation study comparing individual LLMs with *MARE* on requirements modeling task. Results prove the contributions of multi-agents collaboration framework.

We summarize our contributions as follows.

- To the best of our knowledge, this is the first study exploring end-to-end automation of the entire RE process. This study unifies main tasks through LLMs, eliminating the need for specialized models at each task.
- We propose *MARE*, a multi-agents collaboration framework for RE. By merely specifying a rough idea about the requirements, *MARE* iteratively handles requirements elicitation, modeling, verification, and specification.
- We conduct extensive experiments on nine cases and one dataset. Compared with three SOTA baselines, *MARE* shows superiority in modeling. We further manually evaluate the generated specifications and provide insights about the quality of them in three aspects.

In the remainder of the paper, Section II presents the approach. Section III sets up the experiments. Section IV describes the results and analysis. Section V introduces the related work. Section VI concludes our work.

II. APPROACH

This section presents the multi-agent collaboration framework, named *MARE*, for end-to-end RE based on LLMs.

A. Overview

Requirements Engineering is characterized by discussions and brainstorming to define the scope, functionality, and quality of a product. The RE process is challenging because it usually consists of multiple tasks, e.g., requirements elicitation, requirements modeling, requirements verification, and specification. Traditionally, requirements engineers as well as stakeholders actively iteratively collaborate with each other to establish a common understanding of the envisioned product. We investigate the possibility of automating the end-to-end RE process from a novel perspective by proposing a generative framework, i.e. to create a generative model $G(S|X)$ that generates a requirement specification S based on a rough idea of requirements X.

Concretely, we design a multi-agent framework, *MARE*, that leverages LLMs in the RE process. We design prompt-engineering-driven LLM agents to allow these agents with abilities to do RE tasks and introduce a shared workspace in which the artifacts can be stored as the communication structure. To obtain requirements specifications that meet quality standards, these agents work together iteratively and collaboratively to complete the RE process, just like the Human Requirements Engineering team does. Figure 1 depicts this framework, in which, we assume the following four tasks in the RE process:

- **Elicitation.** Given a rough idea of requirements X, this task collects stakeholders' needs U and generates a requirements draft D.
- **Modeling.** Based on the generated requirements draft D, this task generates a requirements model M as required by the 'Metamodel'.
- **verification.** The task detects requirements smells S in the requirements draft D and requirements model M with 'accept criteria'.
- **Specification.** If requirements smells S don't exist, this task generates a requirement specification R based on the requirements draft D, and requirements model M by following a 'Template'. Otherwise, this task generates an Error report E.

B. Tasks

As shown in Figure 1, the RE-Agent Collaboration team conducts iteratively four tasks in *MARE*.

The purpose of the requirements elicitation task is to get stakeholders' needs U corresponding to the rough idea of system requirements X. Various **Stakeholders** can be involved to express their requirements for this system. The requirements **Collector** interviews these stakeholders to further delve into their needs and preferences and write down their needs into a requirement draft D. The information **Collector** agent and a set of **Stakeholders** agents are designed to complete this task.

The purpose of the requirements modeling task is to elaborate the requirements models, including the extraction of the requirements entities and relations. Following the previous study [6], the requirements **Modeler** identifies requirements entities from the requirements draft D, e.g., the *actor*s in use case diagrams and the *machine* in problem diagrams. The **Modeler** then decides whether a certain relationship exists between any pair of entities and determines the type of the relationship, e,g., the *include* in use case diagrams and the *requirements reference* in problem diagrams. Finally, the **Modeler** combines these entities and relationships to form the requirements model M. The **Modeler** agent is designed to accomplish this modeling task.

The purpose of the requirements verification task is to confirm that the system requirements contain all the necessary elements of well-written requirements, e.g., correctness, completeness, and consistency. In the real world, the quality **Checker** first figures out the acceptance criterion, then reads the requirements draft D and requirements model M to assess the quality of the current requirements draft. The **Checker** agent is designed in *MARE* to finish this quality-checking task.

Finally, if the quality of the current version of the requirements document meet the quality criteria, the specification task write the software requirements specification (SRS), otherwise presents an error report. The **Documenter** writes SRS R based
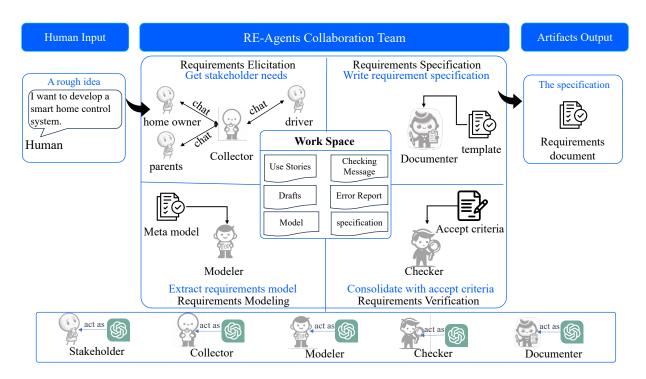
Fig. 1. The overview of our MARE

on requirements draft D and requirements model M according to the pre-defined SRS template or presents the Error Report. The **Documenter** agent is designed for this task.

When an error report appears in the shared workspace, **Collector** and **Modeler** will work sequentially, improving the requirements draft D and requirements model M, respectively, and then **Checker** will check their quality again.

C. Agent Definition

The necessary agents in *MARE* have been identified in Section II-B. They are **Stakeholders**, **Collector**, **Modeler**, **Checker**, and **Documenter**. Here we describe each agent and the interactions among agents.

Agent Specialization. In *MARE*, any agent has a role definition, which includes the name, the profile, the goal, and the actions. The name and profile indicate which role the agents play. The goal describes tasks for which the agent. The action gives the skills that the agent has. For instance, **Collector** can interview with **Stakeholders** and write the requirement draft based on user stories. Thus, the **Collector** agent is equipped with *ProposeQuestion* action and *WriteReqDraft* action. Figure 2 shows an example of the **Collector** agent.

The other agents are defined via the same way based on the analysis in section II-B. The **Stakeholders** agent can express their expectations for the systems and answer questions from **Collector**. Then, So the **Stakeholders** agent can do actions such as *SpeakUserStories* and *AnswerQuestion*. The **Modeler** agent splits the requirements modeling task into two steps, which are extracting requirements entities and determining the relationships among the entities. So the **Modeler** agent can do actions such as *ExtractEntity* and *ExtractRelation*.
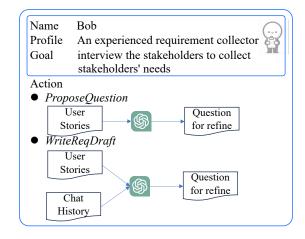

Fig. 2. The profile of the **Collector** agent.

The **Checker** agent can do *CheckRequrement* action. The **Documenter** agent can write requirements specifications or present the error report. So **Documenter** agent is equipped with *WriteSRS* and *WriteChceckReport* action. The role definition of each agent is shown in Table I.

Interaction between Agents. *MARE* adopts a shared workspace mechanism [19] to facilitate effective interaction and collaboration between different agents. As shown in figure 3, the shared workspace stores various requirements artifacts which can be operated by agents, such as the use stories, the requirements drafts, the requirements models, and the requirements specifications, etc.. The requirements artifacts

Agent	Name	Profile	Goal	Action
Stakeholders	Alice	An experienced requirement stakeholder	express and task the stakeholders' need for system to be developed	a. *SpeakUserStories* b. *AnswerQuestion*
Collector	Bob	An experienced requirement collector	interview the stakeholders to collect stakeholders' needs	a. *ProposeQuestion* b. *WriteReqDraft*
Modeler	Carol	An experienced requirement modeler	extract requirement model, including entities and relations	a. *ExtractEntity* b. *ExtractRelation*
Checker	Dave	An experienced requirement checker	check the requirement quality based the requirement model.	a. *CheckRequirement*
Documenter	Eve	An experienced requirement documenter	write the requirement specification or or checking report.	a. *WriteSRS* b. *WriteCheckReport*

Fig. 3. The mechanism of shared workspace.

Action	Input	Ouput
SpeakUserStories	rough_idea_of_requirements, num_user_storeis	user_stories
ProposeQuestion	rough_idea_of_requirements user_stories	requirement_question
AnswerQuestion	rough_idea_of_requirements, requirement_question	stakeholders_answer
WriteReqDraft	rough_idea_of_requirements, user_stories, requirement_question, requirement_answer, draft template	requirement_draft
ExtractEntity	requirement_meta_model, requirement_draft	modeling_entities
ExtractRelation	requirement_meta_model, requirement_draft, modeling_entities	modeling_relationship
CheckRequirement	modeling_entities, modeling_relationship, requirement_draft	checking_message
WriteSRS	modeling_entities, modeling_relationship, requirement_draft, SRS_template	software requirement specification
WriteCheckReport	checking_message	error_report

in the workspace have five properties, which are content, role, caused_by, sent_from, and send_to. The content property represents the content of the requirements artifacts. The role property indicates which agent generates the requirements artifacts. The caused_by property describes which action generates the requirements artifacts, which can be used to perform action migration. The action migration mechanism will be described in detail in the next section. The sent_from and send_to properties give the flow of requirements artifacts among agents. For example, the content of use stories can be "*As a homeowner, I want to be able to control the temperature in my home remotely, so that I can adjust it to my desired level before I arrive.*", the role and cause_by of use stories can be stakeholders and *SpeakUserStories*. The sent_from and send_to can be **Stakeholders** and **Collector**.

D. Action Definition

Action Space. Section II-C describes the actions of each agent. There are a total of nine actions in *MARE*. Each action is implemented by designing a prompt for LLMs. Therefore, we designed multiple prompts of various actions for LLMs and the details about these prompts are at the open link [20]. Table II shows the input and output for each action. The input will be put in each designed prompt and the output is referred to generated requirements artifacts by each action. The input for these actions is generated by agents within *MARE*, except for the rough idea of requirements which is

provided by humans. For the requirements draft and software requirement specification in Table II, their generation follows our predefined templates. The details of these templates are written in the prompt of *WriteReqDraft* action and *WriteSRS* action, which are publicly available at the same open link.

Action Migration. *MARE* iteratively performs a series of actions to achieve the end-to-end RE process. There is a sequential relationship between these actions. For example, action *WriteDraft* happens after action *SpeakUserStories*. Figure 4 shows the whole action migration mechanisms in *MARE*. *HumanInteraction* is the start action and it can provide human feedback to MARE.

III. STUDY DESIGN

To assess the effectiveness of our *MARE*, we perform an extensive study to answer three research questions. In this section, we describe the details of our study, including evaluation cases, metrics, baselines, and experiment settings.
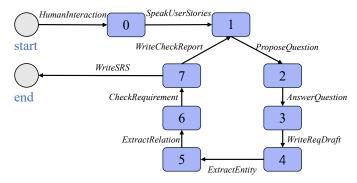
Fig. 4. The action migration diagram in MARE

A. Research Questions

Our study aims to answer three research questions(RQ). In RQ1, we compare our *MARE* to three SOTA automated requirements modeling baselines on 9 evaluation cases and 1 dataset to prove the modeling superiority of *MARE*. In RQ2, we conduct a human evaluation to evaluate the generated software requirements specifications on 9 software systems in three aspects. In RQ3, we conduct an ablation study to prove the contributions of multi-agent collaboration.

RQ1: How do the requirements models generated by *MARE* compared to the SOTA approaches? We first use *MARE* to generate requirements models on 9 evaluation cases. Then, we use multiple metrics to evaluate the generated requirements models and compare them with the existing three SOTA baselines. We select multiple LLMs to verify that *MARE* is effective for different LLMs.

RQ2: How do the requirements specifications generated by *MARE*?. We first use *MARE* to generate software requirements specifications on 9 evaluation cases. Then, we manually evaluate the generated requirements specifications in three aspects. Finally, we calculate the average results.

RQ3: How does *MARE* perform compared to individual LLM? Our *MARE* contains multiple LLMs-based agents. We assess the contributions of agents collaboration by comparing *MARE* with individual LLMs on requirements modeling.

B. Evaluation Cases

We conduct experiments on 5 public evaluation cases for generating use case diagrams, 1 public dataset for generating goal models, and 4 new evaluation cases for generating problem diagrams collected by this work.

Five public evaluation cases are proposed for automated modeling use case diagrams. The 5 cases are the ATM system(ATM) [21], the cafeteria ordering system (COS) [22], the library system (TLS) [23], the assembly system(TAS) [24] and the time monitor system (TMS) [25]. They are common and are often used to evaluate DL model to extract actors and actions in the use case. The original requirements descriptions for these cases can be obtained from the existing literature.

One public evaluation dataset are proposed for automating modeling goal models [26]. The origin requirements descrip-

tions in this dataset come from PURE [27], a dataset of 79 publicly available natural language requirements documents collected from the Web. In this paper, we refer to this dataset as GoalModelDataset.

Four new evaluation cases are proposed for automated modeling problem diagrams. The 4 cases are the smart home control system (TSHCS), the temperature control system (TTCS), the air conditioner control system (TACCS), and the humidistat control system (THCS). The origin requirements descriptions for these cases are shown in the same open link as the section II-D.

C. Evaluation Metrics

Evaluation for requirements models. We use three commonly used metrics to evaluate the performance, i.e., Precision, Recall, and F1. (1) Precision (P), which refers to the ratio of the number of correct predictions to the total number of predictions; (2) Recall (R), which refers to the ratio of the number of correct predictions to the total number of samples in the golden test set; and (3) F1, which is the harmonic mean of precision and recall. When comparing the performance, we care more about F1 since it is balanced for evaluation.

Evaluation for requirements specifications. Following the previous work [28], we manually evaluate the generated specifications by *MARE* in three aspects, including completeness, correctness, and consistency. For each aspect, the score is ranging from 0 to 2 (from bad to good). The evaluators are computer science master students and are not co-authors.

D. Baselines

We select 3 recently proposed automated requirements modeling approaches as baselines. They are state-of-the-art methods for automated modeling of problem diagrams, use case diagrams, and goal models, respectively.

- **IT4RE** [29]: is an approach to automatically identify actors and actions from natural language requirements' description using an NLP parser.
- **EPD** [6]: is a method for **E**xtracting requirements entities, e.g., machine domain, requirement domain and given domain, in **P**roblem **D**iagram based on BERT.
- **HAGM** [26]: is a **H**ybrid **A**pproach to requirements entities, e.g., role, agent, and relation, in **G**oal **M**odel based on machine learning and logical reasoning.

E. Experiment Settings

The implementation details of our *MARE* are as follows. We use the open-source multi-agent framework - MetaGPT [30] to build our *MARE*. For the LLM engine of the *MARE*, we use different versions of ChatGPT-3.5, named gpt-3.5-turbo [31], text-davinci-002 [31] and text-davinci-003 [31]. They are the closed-source model developed by OpenAI. So we experimented by purchasing the access api-key from OpenAI. Considering that the responses of the LLMs are somewhat random, in order to ensure the stability of the experimental results, we set temperature as 0, max_tokens as 3000, top_p

TABLE III
THE RESULTS OF MARE ON MODELING PROBLEM DIAGRAM

Cases	EPD			MARE(gpt-3.5-turbo)			MARE(text-davinci-002)			MARE(text-davinci-003)		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
TSHCS	76.4	67.9	72.7	85.7	92.3	83.8	86.9	79.3	82.9	80.6	82.8	81.7
TTCS	72.5	58.3	64.6	81.6	86.1	83.8	72.9	80.1	76.3	81.7	84.2	82.9
TACCS	71.3	62.4	66.6	81.3	88.4	84.7	86.6	85.7	86.1	85.3	82.5	83.9
THCS	74.7	67.2	70.8	81.8	86.5	84.1	84.3	80.1	82.1	82.2	77.9	80
Average	73.7	63.9	68.7	82.6(8.9)	88.3(24.4)	84.1(15.4)	82.6(8.9)	81.3(17.4)	81.9(13.2)	82.5(8.8)	81.9(18)	82.1(13.4)

TABLE IV
THE RESULTS OF MARA ON MODELING USE CASE DIAGRAM

Cases	IT4RE			MARE(gpt-3.5-turbo)			MARE(text-davinci-002)			MARE(text-davinci-003)		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
ATM	50	83	63	72.9	83.3	77.8	75.6	83.3	79.2	76.7	72.7	74.6
COS	63	56	59	79.2	80	79.6	72.7	79.6	76	75.7	81.8	78.6
TLS	50	56	53	78.3	77.8	78	74.8	74.8	74.8	71.4	78.9	75
TAS	17	100	29	74.9	82.8	78.7	72.3	78.7	75.3	77.8	81.1	79.4
TMS	56	100	71	81.6	79.1	80.3	80.7	77.6	79.1	82.2	77.9	80
Average	47.2	79	55	77.4(30.2)	80.6(1.6)	78.9(23.9)	75.2(28.0)	78.8(0.2)	76.8(21.8)	76.8(29.6)	78.5(0.5)	77.5(22.5)

TABLE V
THE RESULTS OF MARE ON MODELING GOAL MODEL

Datasets	Extracted Elements	HAGM			MARE(gpt-3.5-turbo)			MARE(text-davinci-002)			MARE(text-davinci-003)		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
GoalModelDataset	Role	80.8	83.9	82.3	81.3	84.3	82.7	81.1	83.4	82.3	82.3	83.5	82.9
	Action	93.2	90.4	91.7	93.5	91.4	92.4	92.5	91.2	91.8	92.4	91.2	91.8
Average		87	87.2	87	87.4	87.9	87.6	86.8	87.3	87.1	87.4	87.4	87.4

as 1, frequency_penalty as 0, presence_penalty as 0, best_of as 1. These are the hyper-parameters of the LLMs.

For requirements modeling, the experiments include generating modeling elements in problem diagrams, use case diagrams and goal models. To be specific, for the problem diagrams, we use *MARE* to generate the the machine domains, the requirements domains, the physical devices, the sharing phenomenon, and the requirements references. For the use case diagram, *MARE* is to generate the actors and the use cases. For the goal model, *MARE* is used to generate the roles and the agents.

IV. RESULTS AND ANALYSES

In our first research question, we evaluate the performance of the requirements modeling by *MARE* compared with the previous automated requirements modeling approaches.

RQ1:How do the requirements models generated by *MARE* compared to the SOTA approaches?

Setup. We evaluate baselines (Section III-D) and *MARE* on 9 evaluation cases and 1 public evaluation dataset (Section III-B). The evaluation metrics are described in Section III-C, i.e., the Precision(P), Recall(R), and F1. For all metrics, higher scores represent better performance. Given the name of each evaluation case, *MARE* can generate modeling elements. We compute these three metrics based on these modeling elements.

Results. Table III, Table IV and Table V show the experimental results of the problem diagrams, the use case diagrams, and the goal models compared with three baselines. Red indicates improved, and green indicates decreased.

Analyses. *MARE* achieves the best results among all baselines. For modeling problem diagrams, *MARE*(gpt-3.5-turbo) improves the SOTA approach, i.e., EPD, by 8.9% for average precision, by 24.4% for average recall and by 15.4% for average F1. Similarly, for modeling use case diagrams, *MARE*(gpt-3.5) outperforms the SOTA approach, i.e., IT4RE, by 30.1% for average precision, 1.6% for average recall, and 23.8% for average F1. For modeling goal models, *MARE*(gpt-3.5) improves the SOTA approach, i.e., HAGM, by 0.4% for average precision, 0.7% for average recall, and 0.6% for average F1. These results indicate *MARE* can more accurately generate modeling elements in various requirements models.

> **Answer to RQ1:** *MARE* achieves the best results among all baselines. The average F1 score of *MARE* is 84.1%, 78.9% and 87.6% on three requirement model, improving their SOTA approach by 15.4%, 23.8% and 0.6%

In RQ2, we aim to evaluate the performance of requirement specification generated by *MARE*.

RQ2: How do the requirements specifications generated by *MARE*?

Setup. We evaluate *MARE* on 9 evaluation cases (Section III-B). Given the name of these cases, we collect the requirements specifications generated by *MARE*. To guarantee the correctness of the evaluation, we built an inspection team, which consisted of three master students. All of them are fluent English speakers and have done intensive research work with RE. Each student scored each of the 9 requirements specifications from three aspects (Section III-C). Finally, the

Strategies	The ATM			The Cafeteria			The Library			The Asemantic			The Time Monitor		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Individual LLM	73.1	81.4	77.0	79.7	77.6	78.6	78.9	75.2	77.0	75.1	81.4	78.1	79.2	77.2	78.2
MARE	72.9	83.3	77.8	79.2	80.0	79.6	78.3	77.8	78.0	74.9	82.8	78.7	81.6	79.1	80.3

TABLE VII
THE SCORE OF REQUIREMENT SPECIFICATION

Evaluation Cases	completeness	correctness	consistency
TSHCS	0.87	1.53	1.54
TTCS	0.89	1.62	1.66
TACCS	0.78	1.69	1.87
THCS	1.04	1.71	1.74
ATM	0.98	1.66	1.95
COS	0.97	1.74	1.71
TLS	1.02	1.81	1.82
TAS	1.06	1.85	1.77
TMS	1.21	1.77	1.87
Average	0.98	1.92	1.98

average score was calculated.

Results. Table IV shows the results of evaluation for requirements specifications generated by *MARE* (gpt-3.5-turbo).

Analyses. *MARE* can generate requirement specifications for various software systems. For the 9 various software systems, the completeness score of the generated specification is between 0.78 and 1.21. The correctness score of them is between 1.53 and 1.85. Similarly, the consistency score of them is between 1.54 and 1.95. For average score, *MARE* achieves 0.98 for completeness, 1.92 for correctness, and 1.98 for consistency. These results imply that *MARE* can effectively generate requirements specifications. We also notice that the completeness aspect is slightly lower than the correctness and consistency. By investigating the causes, we believe that there are two reasons. First, the reason why evaluators give lower scores for the completeness is that they think each section of the generated requirement specifications is insufficient. Second, the length of text is reduced by LLMs when generating long text.

Answer to RQ2: *MARE* can effectively generates requirement specification. To be specific, the average score of *MARE* is 0.98, 1.92 and 1.98 on the three aspects.

RQ3:How does MARE perform compared to individual LLM on requirements modeling?

Setup. We compare *MARE* with an individual LLM to generate requirements models on 5 public cases. For *MARE*, we compute three metrics (Section III-C) based on extracted requirements modeling elements from the generated requirements drafts. For the individual LLM, we directly use one LLM to extract requirements modeling elements from the requirements drafts generated by *MARE* and computer the same metrics. Besides, we use gpt-3.5-turbo as the individual LLM and select the use case diagrams.

Results and Analyses. Table IV shows the comparative results between *MARE* and the individual LLM on 5 public cases. *MARE* can generate more correct requirements modeling elements compared with the individual LLM on the 5 public cases. To be specific, *MARE* outperforms the individual LLM by 0.2% for average precision, 2% for average recall and 1.1% for average F1.

Answer to RQ3: *MARE* achieves better results comparing with the individual LLM. This demonstrates the superiority of collaboration of multiple agents.

V. RELATED WORKS

Deep-Learning-based Requirements Engineering. With the rapid development of the deep learning technique, many researchers are devoted to applying DL to RE to improve the effectiveness of software development [32] [33] [34]. For *Requirement elicitation* task, recent work has more focus on mining user needs [35] from the open source community [36] and facilitating conversations [37]. For *Requirement modeling* task, some researchers use pre-defined rules to extract modeling entities and relations [38]. Others focus on building neural network models [39] [40] to extract the requirements models. Recent works also have attempted to explore the ability of LLMs for requirements modeling [12]. For *Requirement verification* task, [41] designed a tool to detect smell in requirements requests based on rules. Besides, many works focus on detecting ambiguous and inconsistent [33] requirements. Compared to these existing studies, our work leverages LLMs throughout the entire requirements engineering through collaboration, rather than just dealing with a single task in RE.

LLM-Based Multi-Agent Frameworks. LLMs have shown remarkable performance across a wide range of domains. [42] improves the reasoning accuracy by leveraging multi-agent debate. [18] proposed ChatDev, a virtual chat-powered software development company consisting of multiple agents based on LLMs. [19] introduce MetaGPT, an innovative meta-programming framework incorporating efficient human workflows into LLM-based multi-agent collaborations. Compared to these studies, our work focuses on tasks in RE.

VI. CONCLUSION

In this paper, we propose MARE, a multi-agent collaboration framework based on LLMs, to improve the efficiency of processing requirements engineering tasks. In MARE, different agents perform requirements elicitation, requirements modeling, requirements verification, and requirements specification, respectively, and collaborate to generate the requirement specification. When conducting these tasks, some external knowledge has been used, e.g. The Modeler extracts the requirements models according to the 'Meta model'; the

Checker checks the models based on 'Accept criteria'; and the Documenter produces the specification in terms of the 'Template'. MARE can contribute to multiple LLMs (such as gpt-3.5-turbo). We demonstrate the superiority of MARE's requirements modeling capabilities by comparing three SOTA automated requirements modeling approaches. Besides, we have provided insights into the quality of the requirements specification generated by MARE.

REFERENCES

[1] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative AI: assessing the role of llms," *CoRR*, vol. abs/2310.13976, 2023.

[2] J. Feng, W. Miao, H. Zheng, Y. Huang, J. Li, Z. Wang, T. Su, B. Gu, G. Pu, M. Yang, and J. He, "FREPA: an automated and formal approach to requirement modeling and analysis in aircraft control domain," in *28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1376–1386.

[3] M. Naumcheva, "Deep learning models in software requirements engineering," *CoRR*, vol. abs/2105.07771, 2021. [Online]. Available: https://arxiv.org/abs/2105.07771

[4] L. Shi, C. Chen, Q. Wang, and B. W. Boehm, "Automatically detecting feature requests from development emails by leveraging semantic sequence mining," *Requir. Eng.*, vol. 26, no. 2, pp. 255–271, 2021. [Online]. Available: https://doi.org/10.1007/s00766-020-00344-y

[5] L. Shi, M. Xing, M. Li, Y. Wang, S. Li, and Q. Wang, "Detection of hidden feature requests from massive chat messages via deep siamese network," in *42nd International Conference on Software Engineering*, 2020, pp. 641–653.

[6] D. Jin, C. Wang, and Z. Jin, "Automating extraction of problem diagrams from natural language requirement documents," in *31st IEEE International Requirements Engineering Conference Workshops*, 2023, pp. 199–204.

[7] C. Wang, L. Hou, and X. Chen, "Extracting requirements models from natural-language document for embedded systems," in *30th IEEE International Requirements Engineering Conference Workshops*, 2022, pp. 18–21.

[8] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh, and L. C. Briand, "Using domain-specific corpora for improved handling of ambiguity in requirements," in *43rd IEEE/ACM International Conference on Software Engineering*, 2021, pp. 1485–1497.

[9] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. C. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *CoRR*, vol. abs/2308.10620, 2023.

[10] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, and E. Chen, "Large language models for generative information extraction: A survey," *CoRR*, vol. abs/2312.17617, 2023.

[11] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," *CoRR*, vol. abs/2303.07839, 2023.

[12] K. Ruan, X. Chen, and Z. Jin, "Requirements modeling aided by chatgpt: An experience in embedded systems," in *31st IEEE International Requirements Engineering Conference Workshops*, 2023, pp. 170–177.

[13] B. Görer and F. B. Aydemir, "Generating requirements elicitation interview scripts with large language models," in *31st IEEE International Requirements Engineering Conference, RE 2023 - Workshops, Hannover, Germany, September 4-5, 2023*, 2023, pp. 44–51.

[14] A. Fantechi, S. Gnesi, L. C. Passaro, and L. Semini, "Inconsistency detection in natural language requirements using chatgpt: a preliminary evaluation," in *31st IEEE International Requirements Engineering Conference Workshop*, 2023, pp. 335–340.

[15] A. D. Rodriguez, K. R. Dearstyne, and J. Cleland-Huang, "Prompts matter: Insights and strategies for prompt engineering in automated software traceability," in *31st IEEE International Requirements Engineering Conference Workshops*, 2023, pp. 455–464.

[16] J. R. Katzenbach and D. K. Smith, *The wisdom of teams: Creating the high-performance organization*. Harvard Business Review Press, 2015.

[17] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C. Chan, Y. Qin, Y. Lu, R. Xie, Z. Liu, M. Sun, and J. Zhou, "Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents," *CoRR*, vol. abs/2308.10848, 2023.

[18] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *CoRR*, vol. abs/2307.07924, 2023.

[19] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, and C. Wu, "Metagpt: Meta programming for multi-agent collaborative framework," *CoRR*, vol. abs/2308.00352, 2023.

[20] Github. Supported materials. [Online]. Available: https://github.com/publicsubmission/MARE-support-material

[21] S. Vinay, S. Aithal, and P. Desai, "An approach towards automation of requirements analysis," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2009.

[22] A. Umber, I. S. Bajwa, and M. A. Naeem, "Nl-based automated software requirements elicitation and specification," in *Advances in Computing and Communications - First International Conference*, ser. Communications in Computer and Information Science, vol. 191, 2011, pp. 30–39.

[23] I. S. Bajwa and M. A. Choudhary, "From natural language software specifications to UML class models," in *Enterprise Information Systems - 13th International Conference*, vol. 102, pp. 224–237.

[24] I. S. Bajwa and M. A. Naeem, "On specifying requirements using a semantically controlled representation," in *Natural Language Processing and Information Systems - 16th International Conference on Applications of Natural Language to Information Systems*, vol. 6716, 2011, pp. 217–220.

[25] A. Umber and I. S. Bajwa, "A step towards ambiguity less natural language software requirements specifications," *Int. J. Web Appl.*, vol. 4, no. 1, pp. 12–21, 2012.

[26] Q. Zhou, T. Li, and Y. Wang, "Assisting in requirements goal modeling: a hybrid approach based on machine learning and logical reasoning," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, 2022, pp. 199–209.

[27] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *25th IEEE International Requirements Engineering Conference*, 2017, pp. 502–505.

[28] D. Xie, B. Yoo, N. Jiang, M. Kim, L. Tan, X. Zhang, and J. S. Lee, "Impact of large language models on generating software specifications," *CoRR*, vol. abs/2306.03324, 2023.

[29] A. Alhroob, A. T. Imam, and R. Al-Heisa, "The use of artificial neural networks for extracting actions and actors from requirements document," *Inf. Softw. Technol.*, vol. 101, pp. 1–15, 2018.

[30] DeepWisdom. Metagpt framework. [Online]. Available: https://github.com/geekan/MetaGPT/

[31] OpenAI. Chatgpt. [Online]. Available: https://platform.openai.com/docs/models/

[32] V. Seibert, "Towards dialogue based, computer aided software requirements elicitation," *CoRR*, vol. abs/2310.13953, 2023.

[33] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Automated handling of anaphoric ambiguity in requirements: A multi-solution study," in *44th IEEE/ACM 44th International Conference on Software Engineering*, 2022, pp. 187–199.

[34] W. Alhoshan, A. Ferrari, and L. Zhao, "Zero-shot learning for requirements classification: An exploratory study," *Inf. Softw. Technol.*, vol. 159, p. 107202, 2023.

[35] Y. Wang, J. Wang, H. Zhang, X. Ming, L. Shi, and Q. Wang, "Where is your app frustrating users?" in *44th International Conference on Software Engineering*, 2022, pp. 2427–2439.

[36] Google. Gitter. [Online]. Available: https://gitter.im/

[37] R. Dickler, S. Dudy, A. Mawasi, J. Whitehill, A. Benson, and A. Corbitt, "Interdisciplinary approaches to getting AI experts and education stakeholders talking," in *Artificial Intelligence in Education. Posters and Late Breaking Results*, ser. Lecture Notes in Computer Science, vol. 13356, 2022, pp. 115–118.

[38] M. Elbendak, P. Vickers, and B. N. Rossiter, "Parsed use case descriptions as a basis for object-oriented class model generation," *J. Syst. Softw.*, vol. 84, no. 7, pp. 1209–1223, 2011.

[39] M. Li, Y. Yang, L. Shi, Q. Wang, J. Hu, X. Peng, W. Liao, and G. Pi, "Automated extraction of requirement entities by leveraging LSTM-CRF and transfer learning," in *IEEE International Conference on Software Maintenance and Evolution*, 2020, pp. 208–219.

[40] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[41] F. Mu, L. Shi, W. Zhou, Y. Zhang, and H. Zhao, "NERO: A text-based tool for content annotation and detection of smells in feature requests," in *28th IEEE International Requirements Engineering Conference*, 2020, pp. 400–403.

[42] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," *CoRR*, vol. abs/2305.14325, 2023.