Leveraging LLMs for the Quality Assurance of Software Requirements

1st Sebastian Lubos
Graz University of Technology, Austria
slubos@ist.tugraz.at

2nd Alexander Felfernig
Graz University of Technology, Austria
afelfern@ist.tugraz.at

3rd Thi Ngoc Trang Tran
Graz University of Technology, Austria
ttrang@ist.tugraz.at

4th Damian Garber
Graz University of Technology, Austria
dgarber@ist.tugraz.at

5th Merfat El Mansi
Graz University of Technology, Austria
merfat.el-mansi@student.tugraz.at

6th Seda Polat Erdeniz
Graz University of Technology, Austria
sedapolat@gmail.com

7th Viet-Man Le
Graz University of Technology, Austria
vietman.le@ist.tugraz.at

2024 IEEE 32nd International Requirements Engineering Conference (RE) | 979-8-3503-9511-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/RE59067.2024.00046

***Abstract*—Successful software projects depend on the quality of software requirements. Creating high-quality requirements is a crucial step toward successful software development. Effective support in this area can significantly reduce development costs and enhance the software quality. In this paper, we introduce and assess the capabilities of a *Large Language Model (LLM)* to evaluate the quality characteristics of software requirements according to the *ISO 29148* standard. We aim to further improve the support of stakeholders engaged in *requirements engineering (RE)*. We show how an LLM can assess requirements, explain its decision-making process, and examine its capacity to propose improved versions of requirements. We conduct a study with software engineers to validate our approach. Our findings emphasize the potential of LLMs for improving the quality of software requirements.**

***Index Terms*—requirements engineering, software requirements, large language model, quality assurance, quality improvement, empirical study**

I. Introduction

The success of software projects depends on the quality of software requirements [1]. Formulating high-quality requirements constitutes an essential step towards achieving favorable outcomes [2]. However, reaching such quality requires significant collaborative efforts from stakeholders engaged in the project. To assist this process, various tools and techniques have demonstrated their efficiency in supporting multiple related tasks [3], including the classification [4], prioritization [5], [6], and quality assessment of requirements [7].

A recent related development in this direction has emerged with the introduction of powerful *Large Language Models (LLMs)* [8] such as *GPT-4* [9] and *Llama 2* [10]. These models show remarkable capabilities in generating natural language, producing a huge variety of accurately crafted responses to input prompts. Applying these models in requirements engineering has received increasing attention in past years. Among others, they have been explored for tasks such as inconsistency detection within sets of requirements [11], [12] and identification of incomplete requirements [13].

While the existing literature explores interesting possibilities for leveraging LLMs to assess and improve specific quality aspects of requirements, addressing the broader issue of *evaluating* the overall quality of software requirements, including multiple quality characteristics, remains unexplored. The study presented in this paper investigates the usage of an LLM to assess whether individual requirement statements adhere to the set of quality characteristics defined in *ISO 29148* [14]. We outline our approach to instruct the LLM in this assessment task and investigate its capacity to transparently explain decisions, aiming to enhance trust in the system. Furthermore, we examine the LLM's potential to propose improved versions of requirements to correct identified quality flaws. To assess the effectiveness of this approach, we present the findings of an empirical user study conducted with participants having a background in software engineering, offering a proof-of-concept evaluation of the approach's benefits.

With this work, we provide a novel approach to using an LLM to transparently evaluate the quality of software requirements by explaining the decisions and presenting suggestions for improvement. This includes three primary contributions. *Firstly*, we show how an LLM can be instructed to evaluate the quality of software requirement statements, comparing its accuracy in flaw detection with that of study participants. *Secondly*, we analyze the LLM's capacity to consistently explain its assessments, thereby enhancing transparency and fostering trust in the evaluation. *Thirdly*, we investigate the LLM's capability to offer constructive suggestions for improved requirements addressing identified flaws.

The remainder of the paper is organized as follows. Section II covers essential aspects required to follow the paper. This includes requirement quality characteristics as defined in *ISO 29148* and fundamental knowledge related to applying LLMs in evaluating software requirements. Section III gives

The presented work has been developed within the research project StreamDiver, which is funded by the Austrian Research Promotion Agency (FFG) under the project number 886205.

2024 IEEE 32nd International Requirements Engineering Conference (RE) | 979-8-3503-9511-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/RE59067.2024.00046

2332-6441/24/$31.00 ©2024 IEEE
DOI 10.1109/RE59067.2024.00046

an overview of existing work on the application of LLM-enhanced techniques in the context of software requirements engineering. In Section IV, we present the research questions, outline the experimental setup of the user study, and provide a comprehensive discussion of its results. Section V explores the implications of the results, taking into account the limitations of this study and suggesting open issues for future work. Finally, Section VI summarizes and concludes the paper.

II. Background

A. Quality Characteristics of Software Requirements

Requirements engineering (RE) is an important phase in the software development process, and the creation of high-quality requirements is crucial for successful software development [2]. The *ISO 29148* [14] describes the international standard for systems and software engineering, including RE. This standard defines *nine* characteristics (dimensions) for individual software requirements, describing the capabilities, characteristics, constraints, and quality factors of a software system. Table I summarizes these quality characteristics.

Characteristic	Description
Appropriate	The level of abstraction is adequate, excludes unnecessary constraints, and avoids implementation details.
Complete	All information needed to understand the requirement is included in the description.
Conforming	The representation of the requirement follows an approved standard template.
Correct	The need is accurately represented in the requirement.
Feasible	The requirement is realizable within the given system constraints considering an acceptable risk.
Necessary	The requirement defines an essential aspect of the system and is irremovable without causing a deficiency.
Singular	The requirement defines only one aspect of the system.
Unambiguous	The requirement is clearly stated, understandable, and allows only one interpretation.
Verifiable	The requirement is formulated in a way that its fulfillment can be proven or, in the best case, measured.

TABLE I: Characteristics of high-quality software requirements as defined in *ISO 29148* [14].

To determine if project requirements fulfill these quality characteristics, a review by project stakeholders is required. These stakeholders need to understand the project's scope and have experience in software projects. Their task is to decide whether the software requirements meet the mentioned quality characteristics. This evaluation process can involve multiple stakeholders reviewing the requirements and reaching a consensus through a majority decision.

B. Instructing LLMs to Evaluate Software Requirements

A *Large Language Model (LLM)* is an advanced *Natural Language Processing (NLP)* model, demonstrating capabilities in comprehending and generating human-like text [8]. These models find application across diverse domains, such as content generation [15], language translation [16], and code completion [17], enabling the creation of relevant information based on contextual input. To execute various tasks, LLMs are provided with specific input queries known as *prompts*, guiding them to generate desired outputs or responses in natural language [18], [19].

When assessing individual software requirements, the LLM must understand the project scope. To achieve this, the project scope description must be incorporated into the prompt. The LLM then has to be instructed to evaluate a given requirement, considering both, the project description and the definition of a specified quality characteristic (refer to Table I), as additional contextual information. The primary task of the LLM is then to classify whether the requirement satisfies the specified quality dimension and to explain its decision. If a quality issue is identified, the LLM is further instructed to propose an improved version of the requirement that addresses the identified flaw. The related LLM prompt used in our study is depicted in Figure 1, including the quality characteristic to be evaluated, its definition, and the project scope description. Additionally, the model's output space is limited to two options, indicating whether the requirement meets the quality criteria or not.

Model Input

Your task is to evaluate the quality of a software requirement.
Evaluate whether the following requirement is {quality_characteristic}.
{quality_characteristic} means: {quality_characteristic_explanation}
The evaluation result must be: 'yes' or 'no'.

Request: Based on the following description of the project:
{project_description}
Evaluate the quality of the following requirement: {requirement}.
Explain your decision and suggest an improved version.

Fig. 1: LLM prompt template used to evaluate a software requirement for a specified quality characteristic and project. Variables are written in curly brackets "{...}".

III. Related Work

The integration of AI-enhanced tools for requirements engineering has received increased attention in recent years. The application of *machine learning (ML)* and NLP forms a crucial focus of research in this domain. Cheeliger et al. [20] conducted a comprehensive literature review on machine learning in requirements engineering, categorizing publications into four tasks: preparation, collection, validation, and negotiation. The review's findings suggest that while the potential of ML in this field is promising, its seamless integration into existing engineering workflows remains challenging. Our work seeks to contribute additional value, specifically in the validation task, by facilitating the quality assessment process.

Leveraging LLMs in software requirement engineering is not novel. Arora et al. [21] conducted a preliminary evaluation of using an LLM to elicitate software requirements for a real-world application. The study results emphasized the feasibility of LLMs for this task. Fantechi et al. [11] explored the use of *ChatGPT* to identify conflicting requirements. The results of the study suggest that the model cannot replace human judgment but can complement manual analysis and speed up the process. In a similar direction, Bertram et al. [12] incorporated an LLM into a suggested toolchain to detect inconsistencies in large requirement specifications within the

automotive industry. The LLM translated the natural language requirements into structured English, enabling formal processing for consistency checking. Luitel et al. [13] investigated the potential of LLMs in enhancing the completeness of requirements expressed in natural language. The evaluation indicates that LLMs can assist in identifying incompleteness within requirements.

Our objective is to enhance the understanding of LLMs in identifying quality issues within software requirements with regard to various dimensions. In exploring the potential of LLMs, we investigate the applicability of the model to provide explainable insights into its assessments, enhance the transparency of decisions, and increase trust in the model. Additionally, we investigate the model's capability to offer constructive suggestions for improving flawed requirements.

IV. Quality Assurance of Software Requirements with LLMs

A. Methodology

Considering the rich capabilities of LLMs in comprehending contexts [22] and reasoning over textual descriptions [23], we expect their potential to support requirements engineering tasks. We expect these models to support the identification of potential weaknesses in requirements artifacts and, ideally, locate quality issues precisely, offering explanations and suggestions for improvement. In doing so, these models might serve as valuable assistants, potentially reducing the need for extensive review cycles.

To evaluate our assumptions, we formulate the following hypotheses:

H1: LLMs achieve comparable performance to software engineers in identifying quality issues in software requirements concerning various quality characteristics.
H2: LLMs can reasonably explain identified quality issues.
H3: LLMs can suggest improved requirement statements, enhancing their quality.

Building upon these hypotheses, we define the following research questions:

R1: How accurately can LLMs identify quality issues in software requirements across different quality characteristics?
R2: Can LLMs provide meaningful explanations for their quality assessments of software requirements?
R3: Do LLMs have the ability to suggest improved versions of software requirements, addressing identified flaws?

B. Study Participants

Our empirical study involved individuals with educational and professional backgrounds in software engineering. While participants were not required to be experienced experts in RE, they needed practical experience working on software projects. This background of participants warranted that participants had fundamental knowledge in working with software requirements. The study instructions included a definition and explanation of the quality characteristics to ensure a clear understanding of those among all participants.

Model Input

Your task is to generate software requirements for a given project description.
A software requirement is a statement that expresses a need and its associated constraints and conditions in the form of a natural language.
The statement should include a subject and a verb, together with other elements necessary to adequately express the content of the requirement.
The requirements should serve as a basis for a software developer to implement the project.

Request: Based on the following description of the project:
{project_description}
Suggest 7 functional and 3 non-functional software requirements.

Fig. 2: LLM prompt template used to generate software requirements for an example project to implement a *Stopwatch app for Android smartphones*. Variables are written in curly brackets "{...}".

C. Dataset and Preprocessing

For our empirical study, we used two example projects described in the following:

1) Stopwatch Project: We initially considered a hypothetical small project. As most software projects are typically large and complex, understanding their context and evaluating their requirements requires high effort. For this reason, we decided to first consider a relatively simple project comprising ten requirements, seven functional and three non-functional, to evaluate our proof-of-concept. The software to be described with those requirements is a *Stopwatch app designed for Android smartphones*. We used an LLM[1] to generate this set of requirements by using the prompt shown in Figure 2, along with the following project scope description: "*Develop an intuitive and user-friendly Stopwatch app for Android smartphones that allows users to easily measure time intervals with precision. The app should have a simple and clean interface, allowing users to start, pause, and reset the stopwatch easily. Additionally, the app should display the total time elapsed and provide options for split and lap times. The goal is to create an efficient and reliable tool that can be used in various contexts such as sports, cooking, or any other activity where accurate time measurement is important.*"

The LLM was not explicitly instructed to fulfill specific quality characteristics during the generation of requirements, which means that realistic and potentially flawed samples were generated. To ensure their coherence and suitability as an explanatory example for our study, we carefully reviewed the generated requirements[2]. The generated requirements are shown in Table II

2) DigitalHome Project: As a second example, we focused on the software requirements of a real-world project included in the *PURE dataset* [24]. This dataset collects requirement documents from public projects. From those samples, we selected *DigitalHome*, a project detailing the requirements regarding a smart home prototype, as a representative case for our study. These requirements need to be evaluated by a group

[1] *Llama 2* [10] with 70 billion parameters as described in Section IV-D.

[2] We deliberately refrained from correcting the requirements concerning the quality characteristics outlined in Table I, as we wanted to consider a realistic example that could contain quality issues.

Id	Requirement
$\mathbf{r_1}$	The app shall allow users to start the stopwatch by tapping a prominent 'Start' button.
$\mathbf{r_2}$	The app shall allow users to pause the stopwatch by tapping a prominent 'Pause' button.
$\mathbf{r_3}$	The app shall allow users to reset the stopwatch to zero by tapping a prominent 'Reset' button.
$\mathbf{r_4}$	The app shall display the total time elapsed since the last reset.
$\mathbf{r_5}$	The app shall provide an option for split times, allowing users to manually enter a split time and display it alongside the total time elapsed.
$\mathbf{r_6}$	The app shall provide an option for lap times, allowing users to manually enter a lap time and display it alongside the total time elapsed.
$\mathbf{r_7}$	The app shall allow users to view their previous splits and laps, including the time taken for each split and lap.
$\mathbf{r_8}$	The app shall be designed with a simple and clean interface, ensuring ease of use for users of all ages and skill levels.
$\mathbf{r_9}$	The app shall be optimized for performance, ensuring that it operates efficiently and without significant lag or errors.
$\mathbf{r_{10}}$	The app shall be compatible with Android smartphones running version 10 or higher, ensuring that it can be installed and used on a wide range of devices.

TABLE II: Generated requirements for the development of a *Stopwatch app for Android smartphones*.

of study participants who understand the project scope, and we assumed that a basic understanding of smart homes can be presumed. The project includes *63* requirements, comprising *42 functional* and *21 non-functional* requirements.

D. LLM-based Requirement Evaluation

We conducted an LLM-based evaluation of requirements utilizing the *Llama 2* language model [10] with 70 billion parameters, fine-tuned to complete chat responses. We decided to perform our experiments with this LLM, as it achieved competitive benchmark performance[3] at the time the study was conducted and was published as open-source[4]. For practicality, we accessed the hosted model on *Replicate*[5] with parameters $\text{temperature} = 0.01, \text{max\_new\_tokens} = 2000$. Given that real-time responses were not a criterion for our evaluation, we used the most accurate model despite its slower response generation speed (≈*15-30 seconds*). Details on the LLM instructions for this task are provided in Section II-B.

E. Experimental Setup

To assess the validity of our hypotheses regarding the capabilities of LLMs in evaluating the quality of software requirements and to address the research questions outlined in Section IV-A, we implemented the following procedure for our experiments:

1) Instruct study participants to evaluate the requirements of both example projects (see Section IV-C) based on the *ISO 29148* quality characteristics (see Section II-A) and their project scope description.
2) Use the requirements of both example projects as input for LLM requests, instructing it to assess their quality (see Section IV-D) and record the model's responses. The project scope descriptions and quality characteristic definitions are part of the LLM prompt.
3) Instruct study participants to evaluate the LLM responses, considering:
   - The *agreement* with the LLM's decision (**R1**)
   - The *plausibility* of the LLM's explanation (**R2**)
   - The *quality* of the improved requirement suggested by the LLM (**R3**)
4) Compare the assessment of quality flaws by the LLM and study participants, using the study participants' responses as the baseline.

Each requirement and its corresponding LLM response has been evaluated by at least *four* individuals with software engineering backgrounds. They used the project description and an explanation of quality characteristics as a guideline for their assessments. We used their evaluation to establish the ground truth for our project examples. A majority vote was applied to label the samples, determining if a requirement meets the observed quality characteristic. If a majority consensus was not reached or if the majority expressed uncertainty, we labeled the requirement as not fulfilling the observed quality characteristic. Our assumption was that when a group of reviewers cannot reach a unanimous decision, the requirement is flawed, and additional review is needed.

Using the described process, we had two phases of study participants evaluating the quality characteristics of requirements. In the first evaluation phase, which we name *independent assessment*, the study participant classified requirements without knowledge about the decision of the LLM. In the second phase, the study participants knew the decision and explanation of the LLM and decided whether they agreed. We refer to the second phase as *bound assessment*. Study participants might have classified requirements differently in the second phase, indicating that the LLM has convinced them. During this phase, study participants evaluated the explanation provided by the LLM by categorizing it as either plausible, implausible, or neutral. Similarly, the quality of suggested requirement improvements by the LLM was evaluated.

F. Results

1) Agreement between Study Participants and LLM (R1): As a first step, we evaluate the agreement between the LLM and study participants when assessing the quality characteristics of requirements in our investigated *Stopwatch* and *DigitalHome* projects. In line with our initial hypothesis, we anticipate a high level of agreement between the two. To measure this agreement, we employ the *Cohen's Kappa* metric [25], a measure of inter-rater reliability that quantifies the extent to which two raters agree on the categorical labeling of data samples, accounting for the potential of agreement by chance. We considered an agreement between the study participants and LLM when their assessment of a requirement and quality characteristic was identical.

In Table III, the *Cohen's Kappa* values are shown. For the independent assessment, a *weak* agreement is observed

[3]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
[4]https://github.com/meta-llama/llama
[5]Model version: *meta/llama-2-70b-chat: 02e509c789964a7ea8736978a43525956ef40397be9033abf9fd2badfe68c9e3*

Project	Independent Assessment	Bound Assessment
Stopwatch	0.4028	0.7545
DigitalHome	0.0486	0.2223

TABLE III: Cohen's Kappa value indicating the inter-rater agreement between the study participants and LLM in classifying requirements as flawed. Independent assessment means that both raters answered without knowledge about how the other rater answered, while bound assessment describes the situation where the study participants classified the requirements being aware of the decision and explanation of the LLM.

between the LLM and study participants' evaluations for the *Stopwatch* project ($0.4 <= \kappa < 0.6$), according to the interpretation outlined by McHugh in [25]. In the bound assessment, we could measure a *substantial* agreement ($0.6 <= \kappa < 0.8$) between the LLM and study participants for the *Stopwatch* example, which indicates that the study participants agreed with the LLM assessment of requirements in many cases. For the *DigitalHome* project, we could not establish an agreement in the independent assessment, while a *fair* agreement ($0.2 <= \kappa < 0.4$) was identified in the bound assessment scenario. The metric values show that we cannot assert that the LLM and study participants yield identical evaluations.

While *Cohen's Kappa* metric measures agreement between raters, it lacks an assessment of the validity of responses, i.e., whether two raters have correctly classified samples. To understand the values better, we provide an additional perspective to the evaluations. The Tables IVa and IVb show the assessments of quality characteristics by study participants and the LLM for the requirements of the *Stopwatch* project (refer to Table II). The first table indicates the majority decisions (see Section IV-E) of study participants if a requirement fulfills a quality characteristic or not. The second table shows the LLM decisions for the same requirements. The sum values $\mathbf{\Sigma_{req}}$ in the tables reflect the number of fulfilled quality characteristics per requirement. The sum in column $\mathbf{\Sigma_{qc}}$ shows the number of requirements fulfilling a specific quality characteristic.

The evaluation summary reveals relevant findings. *Firstly*, the total count of requirements fulfilling a quality characteristic is higher for study participants, suggesting that the LLM tends to evaluate more requirements negatively. The observation can be confirmed for the *DigitalHome* project (see Table V), where the study participants decided on more than twice as many assessments (*514*) that a requirement fulfilled the characteristic, compared to the LLM (*225*). *Secondly*, substantial differences exist in the positive evaluations of various quality dimensions, with the LLM notably assessing fewer requirements as *appropriate*, *correct*, *singular*, *unambiguous*, and *verifiable*. Again, the same observation was made for the *DigitalHome* project. Table V shows the sum of requirements fulfilling a quality characteristic given the study participants' and LLM's decisions.

To assess the validity of the LLM evaluation, we establish the study participants' assessment as the ground truth. This information is used to calculate the *precision* and *recall* of the LLM's capacity to identify quality flaws in requirements

(a) Study participants' assessment.

	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$	$\mathbf{r_7}$	$\mathbf{r_8}$	$\mathbf{r_9}$	$\mathbf{r_{10}}$	$\mathbf{\Sigma_{qc}}$
Appropriate	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	**7**
Complete	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	**2**
Conforming	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓	**6**
Correct	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	**9**
Feasible	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Necessary	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Singular	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	**7**
Unambiguous	✓	✓	✓	✗	✗	✗	✓	✗	✗	✓	**5**
Verifiable	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	**6**
$\mathbf{\Sigma_{req}}$	**7**	**6**	**8**	**6**	**6**	**4**	**7**	**5**	**5**	**8**	**62**

(b) LLM assessment.

	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$	$\mathbf{r_7}$	$\mathbf{r_8}$	$\mathbf{r_9}$	$\mathbf{r_{10}}$	$\mathbf{\Sigma_{qc}}$
Appropriate	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	**3**
Complete	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	**2**
Conforming	✓	✗	✗	✓	✗	✗	✓	✓	✗	✓	**5**
Correct	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	**6**
Feasible	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Necessary	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Singular	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	**0**
Unambiguous	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	**2**
Verifiable	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	**2**
$\mathbf{\Sigma_{req}}$	**3**	**2**	**2**	**8**	**3**	**2**	**4**	**5**	**4**	**8**	**40**

TABLE IV: Requirements evaluation of the *Stopwatch* project. "✓" indicates a requirement that fulfills the quality characteristic. $\mathbf{\Sigma_{req}}$ shows the number of fulfilled quality characteristics per requirement. $\mathbf{\Sigma_{qc}}$ shows the number of requirements fulfilling a quality characteristic.

	Study Participants	LLM
appropriate	60	12
complete	53	15
conforming	61	42
correct	63	27
feasible	61	56
necessary	59	48
singular	57	5
unambiguous	49	18
verifiable	51	2
$\mathbf{\Sigma_{reqs}}$	**514**	**225**

TABLE V: The number of requirements in the *DigitalHome* project fulfilling a quality characteristic based on the assessment of the study participants and LLM evaluation. $\mathbf{\Sigma_{reqs}}$ is the sum of fulfilled characteristics of requirements. The total number of evaluated requirements in this project is *63*.

accurately, using the equations 1 and 2.

$$Precision = \frac{\#Flawed\ reqs.\ identified\ by\ LLM}{\#Reqs.\ classified\ as\ flawed\ by\ LLM} \quad (1)$$

$$Recall = \frac{\#Flawed\ reqs.\ identified\ by\ LLM}{\#All\ \ flawed\ reqs.} \quad (2)$$

In this context, the *precision* describes the proportion of identified requirements with quality issues among all requirements the LLM has classified as flawed. The *recall* values represent how many requirements with quality issues in the ground truth have been identified correctly by the LLM. The values for the independent and bound assessments are reported in Table VI.

Project	*Independent Assessment* Precision	Recall	*Bound Assessment* Precision	Recall
Stopwatch	0.50	0.8929	0.8837	1.0
DigitalHome	0.1257	0.7414	0.3214	1.0

TABLE VI: Precision and recall of LLM recognized requirement flaws. Independent assessment means that both raters answered without knowledge about how the other rater answered, while bound assessment describes the situation where the study participants classified the requirements being aware of the decision and explanation of the LLM.

For the independent assessment, the precision is relatively low in both investigated projects, which implies a higher likelihood of false positives. Quality issues seem to be incorrectly recognized by the LLM using the study participants' evaluation as ground truth. However, the precision value is higher when reviewing the bound assessment. This can be explained by the fact that the study participants' decisions have changed regarding the LLM assessment and explanation. We assume that the study participants received a new perspective on the requirement and, therefore, reconsidered their decision. This means manual verification is still required, even if the number of misclassified requirements is lower. For example, in the *Stopwatch* project, the LLM incorrectly identified *25* out of *90* items as flawed during the independent assessment. This number decreased to *5* in the bound assessment. Considering the precision values for the *DigitalHome* project, a trend to higher precision in the bound assessment can be observed. Yet, the values are much lower, as the number of wrongly identified flaws by the LLM is quite high. We assume that this is related to the project scope, which is more complex than the *Stopwatch* example.

For both projects and assessment phases, high recall values were measured. Those affirm that a large number of quality flaws are indeed identified by the LLM. This outcome suggests that the LLM can be effectively utilized to highlight quality issues in software requirements and guide stakeholders toward requirements that may need improvement.

Based on our analysis, we can answer the first research question, which focuses on the accuracy of LLMs in correctly identifying quality issues in software requirements. For independent assessment, the *Cohen's Kappa* metric indicates weak or even non-existent agreement between study participants and the LLM when determining whether a software requirement fulfills a quality characteristic (refer to Table III). However, the agreement increases for the bound assessment phase, which indicates that study participants changed their opinion toward the LLM given its explanation. We assume that the evaluation of the LLM helped correct the participants' assessment. This might help identify potential quality issues with requirements, especially when reviewers and stakeholders lack experience in RE. The LLM might partly compensate for this by suggesting additional perspectives.

The recall values demonstrate that the LLM effectively identified the majority of flawed software requirements. This suggests it can accurately predict the presence of quality issues in software requirements (see Table VI). Nevertheless, manual verification is still required, given the low precision values, as false positives may occur. For the first research question, we conclude that LLMs can accurately identify quality flaws in many cases and show the potential to guide stakeholders toward quality issues they might have overlooked otherwise.

	Stopwatch	DigitalHome
appropriate	100%	70%
complete	100%	100%
conforming	100%	100%
correct	80%	100%
feasible	100%	100%
necessary	100%	100%
singular	100%	80%
unambiguous	100%	100%
verifiable	100%	70%

TABLE VII: Percentages of LLM-generated quality-related explanations considered plausible by study participants, categorized by project and quality characteristics.

2) Meaningful Explanations of LLM-Assessment (R2): The second research question investigates the capability of LLMs to provide reasonable explanations for their quality assessments of software requirements. To answer this question, we engaged study participants to review the decision and explanation offered by the LLM for each requirement and quality dimension. As shown in Table VII, the majority of study participants agreed that the LLM could plausibly explain its assessment of software requirement quality issues in both investigated example projects. Therefore, we can affirmatively answer the second research question. We conclude that the LLM can provide reasonable explanations for its decisions. The explanations were considered meaningful, regardless of whether the LLM classified the requirement as flawless or having a quality issue. This result is promising for the successful integration of this approach in real-life settings, as the LLM's decision is traceable for reviewers and can be accurately validated. Nonetheless, we recommend replicating the experiment with more complex requirements to observe if the LLM-generated explanations can also help generate reliable explanations for those.

3) LLM-improved Requirements (R3): The third research question is whether an LLM can suggest improved versions of flawed software requirements. To answer this question, we instructed study participants to evaluate the LLM-proposed improved versions of original requirements in the *Stopwatch* and *DigitalHome* projects. The study participants determined whether the suggested version improved the original requirement. As shown in Table VIII, the suggested version was considered an improvement in most cases. As the number of requirements with potential issues with respect to feasibility and necessity was small (see Tables IVb and V), there were only a few examples where an improvement could be evaluated. Therefore, more examples should be reviewed in the future to validate this aspect further.

Overall, the outcome emphasizes the LLM's capacity to assist reviewers in correcting flaws in software requirements, thereby contributing to an overall improvement in the quality of requirements. Based on the collected data, we can affirma-

	Stopwatch	DigitalHome
appropriate	100%	66%
complete	100%	100%
conforming	100%	100%
correct	100%	100%
feasible	-	50%
necessary	-	50%
singular	90%	66%
unambiguous	100%	100%
verifiable	100%	90%

TABLE VIII: Percentages of LLM-generated requirement improvements considered enhancements compared to the original requirements by study participants. Values are categorized by project and quality characteristics. For *feasibility* and *necessity* in the *Stopwatch* project, values are not available as this quality issue was not identified in the requirements and hence could not be improved.

tively answer the third research question, indicating that the LLM can suggest versions of requirements that address flaws within a given quality dimension. This shows that LLMs may offer immediate benefits in software requirements engineering.

V. Discussion

A. Study Results

The results of this empirical study provide valuable insights into the potential and capabilities of leveraging an LLM to support the development of high-quality software requirements. As demonstrated by the results, the LLM successfully identified a majority of the incorporated quality issues, showing its utility in the requirements review process. This ability can significantly reduce review time by directing reviewers to requirements that may contain quality flaws. However, it is important to note that the intervention of the reviewers remains necessary, as the LLM did not identify all quality issues in the examined samples. The level of collaboration between humans and LLMs is a topic of recent research. Faggioli et al. [26] outline various integration stages from full human judgment to complete automation. We assume that LLMs are currently most beneficial in assisting humans by suggesting and explaining potential flaws for the quality assurance of software requirements. Those need to be subsequently verified and corrected with the help of the LLM. The current capabilities of LLMs are not sufficient to fully automate this task.

The LLM consistently offered meaningful explanations, regardless of whether participants agreed with the LLM's quality assessment. This suggests that the LLM is transparent in articulating its reasoning, potentially helping reviewers understand why a requirement was identified as having a quality issue. Additionally, these explanations may encourage reviewers to revise their assessments by considering an alternative perspective. We believe that explaining decisions is crucial for establishing LLM-assisted review processes for software requirements. Such explanations could facilitate the validation of assessments and increase trust in the system.

Moreover, the recommendation of improved requirements appears beneficial and reliable based on the evaluation of our examples. The study participants agreed that most of the LLM-suggested requirements improved their quality, regardless of whether the study participants agreed with the LLM's quality assessment. This additional perspective might help reviewers by providing an alternative viewpoint and potentially refining the formulation of requirements where quality issues are not particularly severe. In this study, the LLM was explicitly instructed to correct the identified flaw concerning the assessed quality characteristic, ignoring the possibility that a requirement might require improvement across multiple dimensions. Addressing the combination of different dimensions in a single prompt instruction to achieve an overall enhanced requirement remains an important consideration for future research.

Overall, we believe that LLMs have the potential to enhance the quality of software requirements, particularly by reducing the manual review effort. It is crucial to interpret the findings of our study as a proof-of-concept, emphasizing the need to develop and evaluate ideas regarding this application further.

B. Threats to Validity

While the study presented in this paper provides valuable insights into the potential of assuring and improving the quality of software requirements with LLMs, we acknowledge certain limitations. *Firstly*, the evaluation was conducted using a hypothetical example project and only one real-life scenario, impacting the generalizability of the presented results. To obtain better evidence of the general applicability, we plan to examine the presented approach within upcoming real-life projects, using the LLM as a supportive tool during the review phase of requirements. With this, we hope to learn more about its usefulness in practical settings.

Secondly, the investigated projects were relatively simple, meaning that the reported results might differ for larger projects with more complex requirements. Additionally, we did not address the token limit aspect in the prompts for more extensive project descriptions. In such cases, a *Retrieval Augmented Generation (RAG)* [27], [28] approach could be employed to overcome this challenge by loading relevant project data into the prompt context as needed to reduce the context length.

Thirdly, our study did not explore and compare different options to instruct LLMs. The focus was on one specific model (Llama 2) and a single prompt template, making it impossible to estimate the general quality of LLMs in evaluating software requirements. The usage of different LLMs or prompts might lead to other results. Consequently, we consider future studies addressing this aspect by comparing various prompting strategies and models. Given the extensive research on LLMs and the ongoing release of improved model versions, it is important to continuously evaluate new approaches for interaction and application in experimental studies.

C. Future Work

In addition to assessing the quality of individual requirements, we see the potential for applying our approach to evaluate the quality characteristics of requirement sets. Similar

to the quality dimensions for individual statements, the *ISO 29148* [14] specifies characteristics for sets or requirements, encapsulating the quality aspects of a consistent solution that meets stakeholder expectations while adhering to project constraints. We believe that extending our approach to cover these aspects could yield additional benefits.

Additionally, we consider the potential of using an LLM to identify hidden dependencies between requirements as an open issue. Existing approaches for detecting these dependencies [29], [30] use natural language techniques to extract features as a basis for classification of dependencies between requirements. In our future research, we aim to develop a framework for instructing an LLM to handle this task. We believe that the extensive general knowledge of LLMs, combined with their ability to understand specific project contexts, could significantly enhance the accuracy of identifying these dependencies.

Furthermore, we intend to analyze additional projects to provide a more general understanding of the capabilities of LLMs in improving software requirements. For this, we consider the application of *Retrieval Augmented Generation* [27], [28] for complex and huge projects as a promising approach for future research. In this context, exploring strategies to identify and access the relevant information is essential to let the LLM understand the project scope.

VI. Conclusions

In this paper, we analyzed the possibilities of using an LLM to evaluate the quality of software requirements in accordance with the quality characteristics defined in the *ISO 29148* standard. We discuss an approach for instructing the LLM to assess requirements, explain its decision-making process, and suggest enhanced versions when quality issues are detected. To evaluate the LLM's capabilities in fulfilling these tasks, we conducted an empirical study involving participants with backgrounds in software engineering. The findings indicate that the LLM accurately identifies the majority of requirements with quality flaws, demonstrating its potential to assist in the requirement engineering process. Moreover, the study reveals that the LLM provides reliable explanations for its decisions, which can improve trust in the system. The evaluation of LLM-suggested improvements to requirements with quality issues indicates their effectiveness in resolving quality concerns. To summarize, this paper highlights the utility of LLMs in ensuring the quality of software requirements, suggesting promising potentials for stakeholder support throughout the software requirements engineering process, and motivating future related research.

References

[1] M. I. Kamata and T. Tamai, "How does requirements quality relate to project success or failure?" in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007, pp. 69–78.

[2] A. Hussain, E. O. Mkpojiogu, and F. M. Kamal, "The role of requirements in the success or failure of software projects," *International Review of Management and Marketing*, vol. 6, no. 7, p. 306–311, 2016.

[3] A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej, *An Overview of Recommender Systems in Requirements Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 315–332. [Online]. Available: https://doi.org/10.1007/978-3-642-34419-0_14

[4] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 490–495.

[5] A. Felfernig, M. Stettinger, M. Atas, R. Samer, J. Nerlich, S. Scholz, J. Tiihonen, and M. Raatikainen, "Towards utility-based prioritization of requirements in open source environments," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, 2018, pp. 406–411.

[6] S. Vijayakumar and N. P. S., "Use of natural language processing in software requirements prioritization – a systematic literature review," *International Journal of Applied Engineering and Management Letters (IJAEML)*, vol. 5, no. 2, p. 152–174, Nov. 2021. [Online]. Available: https://www.supublication.com/index.php/ijaeml/article/view/399

[7] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," *Information and Software Technology*, vol. 67, pp. 180–195, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584915001299

[8] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Comput. Surv.*, vol. 56, no. 2, sep 2023. [Online]. Available: https://doi.org/10.1145/3605943

[9] OpenAI, "Gpt-4 technical report," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.09288

[10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.09288

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

[12] V. Bertram, H. Kausch, E. Kusmenko, H. Nqiri, B. Rumpe, and C. Venhoff, "Leveraging natural language processing for a consistency checking toolchain of automotive requirements," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*, 2023, pp. 212–222.

[13] D. Luitel, S. Hassani, and M. Sabetzadeh, "Using language models for enhancing the completeness of natural-language requirements," in *Requirements Engineering: Foundation for Software Quality*, A. Ferrari and B. Penzenstadler, Eds. Cham: Springer Nature Switzerland, 2023, pp. 87–104.

[14] International Organization for Standardization, "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[15] S. Moore, R. Tong, A. Singh, Z. Liu, X. Hu, Y. Lu, J. Liang, C. Cao, H. Khosravi, P. Denny, C. Brooks, and J. Stamper, "Empowering education with llms - the next-gen interface and content generation," in *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, N. Wang, G. Rebolledo-Mendez, V. Dimitrova, N. Matsuda, and O. C. Santos, Eds. Cham: Springer Nature Switzerland, 2023, pp. 32–37.

[16] H. Huang, S. Wu, X. Liang, B. Wang, Y. Shi, P. Wu, M. Yang, and T. Zhao, "Towards making the most of llm for translation quality estimation," in *Natural Language Processing and Chinese Computing*, F. Liu, N. Duan, Q. Xu, and Y. Hong, Eds. Cham: Springer Nature Switzerland, 2023, pp. 375–386.

[17] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.10620

[18] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in*

Computing Systems, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411763.3451760

[19] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Prompting is programming: A query language for large language models," *Proc. ACM Program. Lang.*, vol. 7, no. PLDI, jun 2023. [Online]. Available: https://doi.org/10.1145/3591300

[20] C. Cheligeer, J. Huang, G. Wu, N. Bhuiyan, Y. Xu, and Y. Zeng, "Machine learning in requirements elicitation: a literature review," *AI EDAM*, vol. 36, p. e32, 2022.

[21] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative ai: Assessing the role of llms," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.13976

[22] W. Zhou, S. Zhang, H. Poon, and M. Chen, "Context-faithful prompting for large language models," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.11315

[23] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2212.10403

[24] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: a dataset of public requirements documents," Sep. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1414117

[25] M. McHugh, "Interrater reliability: The kappa statistic," *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, vol. 22, pp. 276–82, 10 2012.

[26] G. Faggioli, L. Dietz, C. L. A. Clarke, G. Demartini, M. Hagen, C. Hauff, N. Kando, E. Kanoulas, M. Potthast, B. Stein, and H. Wachsmuth, "Perspectives on large language models for relevance judgment," in *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, ser. ICTIR '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 39–50. [Online]. Available: https://doi.org/10.1145/3578337.3605136

[27] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[28] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2312.10997

[29] M. Atas, R. Samer, and A. Felfernig, "Automated identification of type-specific dependencies between requirements," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018, pp. 688–695.

[30] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe, and G. Deshpande, "New approaches to the identification of dependencies between requirements," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1265–1270.