On Converting Natural Language Requirements into Semi-Formal Templates Using LLMs

Julian Roßkothen
Karlsruhe Institute of Technology (KIT)
and Vector Informatik GmbH
Karlsruhe, Germany
rosskothen@kit.edu

Dominik Fuchß
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
dominik.fuchss@kit.edu

Florian Erdösi
Vector Informatik GmbH
Karlsruhe, Germany
florian.erdoesi@vector.com

Maria Floruß
Vector Informatik GmbH
Karlsruhe, Germany
maria.floruss@vector.com

Jan Keim
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
jan.keim@kit.edu

Tobias Hey
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
hey@kit.edu

***Abstract*—[Context and Motivation] Semi-formal syntax templates for natural language requirements positively impact various requirements metrics, such as singularity. Using templates such as MASTeR or EARS also improves understandability. Requirements that conform to templates are easier for humans to understand than unrestricted requirements. [Question/Problem] However, converting requirements into templates is time-consuming and requires substantial prior training and in-depth domain knowledge. Thus, most requirements are still written in unrestricted natural language. [Principal Ideas and Results] Our approach is to use large language models (LLMs) to convert unrestricted natural language requirements into templates. Our evaluation demonstrates the proficiency of LLM-based systems. LLM-converted requirements are rated similarly to human rephrasings, especially for shorter requirements. Thus, they can be used to accelerate the requirements rephrasing process. [Contribution] In this paper, we present an approach for automatically parsing free-text requirements into templates with LLMs. We also provide an overview of metrics for automatically validating such systems and use them to compare our rephrased requirements with a ground truth. In a practitioner survey comparing LLM- and human-converted requirements, we assess the validity of these metrics.**

***Index Terms*—Requirements Syntax Templates, Requirements Engineering, Large Language Models, LLM, MASTeR Template**

I. Introduction

Natural-language (NL) requirements remain the primary vehicle for capturing stakeholder needs in industrial projects. Yet, unrestricted NL introduces ambiguity, inconsistent phrasing, and variable granularity, hindering automation, traceability, and scalable quality assurance. Semi-formal templates such as MASTeR provide human-readable patterns that reduce ambiguity and improve singularity, clarity, and compliance with guidelines [1], [2]. For these reasons, those templates are often used in safety-critical industries, such as automotive and avionics, where compliance and verification are mandatory [3], [4]. However, migrating large, legacy specifications to a template scheme is costly: it requires both domain and template expertise, often involves splitting a single free-text requirement into multiple template-atomic requirements, and must preserve links to other engineering artifacts at scale.

Recent advances in large language models (LLMs) suggest an opportunity to automate parts of this transition. LLMs have demonstrated strong performance on translation-like tasks [5] and can draft high-quality requirements artifacts [6], but industrial adoption demands more than raw generation quality: organizations need project-scale throughput, predictable behavior, and measurable quality signals that align with existing tool chains and review practices.

This paper presents a practically oriented approach for converting free-text requirements into the MASTeR template using LLMs. The MASTeR template syntax consists of several distinct patterns to capture the type of requirements in order to improve requirements quality [7]. The workflow, co-designed with an industry partner, a leading automotive software and tools supplier, targets batch processing of thousands of requirements and integrates guardrails to keep LLMs in control: a reflection loop [8] iteratively improves rephrasings based on validator feedback; a grammar-based MASTeR checker and an LLM-as-a-judge complement each other to balance precision and coverage; and an INCOSE-Guide-aligned quality rater provides interpretable signals to guide acceptance and review. The INCOSE Guide for Writing Requirements [9], [10] contains characteristics of well-formed requirements.

We evaluate the approach on a publicly available dataset of 249 source requirements and 410 human-rephrased MASTeR targets [11], [12], and we compare multiple LLM configurations using both text-similarity and template-accuracy metrics. To ground the results in practitioner needs, we conduct a survey with requirements specialists from our industry partner that rates the usefulness of the LLM-generated rephrasings.

Our results indicate that LLM-generated rephrasings draw near human quality for shorter, less complex requirements and thus might substantially accelerate template migration when used as drafting assistance with validation and review in the loop. We observe that BERT Score and template-accuracy

signals (MASTeR main template score) are the most informative automatic proxies for practitioner preferences among the tested metrics. However, in isolation they might not suffice for scalable quality monitoring in industrial pipelines, but should be combined and complemented with human supervision.

This work makes the following contributions:

(i) an industry-motivated, project-scale LLM workflow for converting unrestricted NL requirements to the MASTeR template, combining a reflection loop with complementary validators;
(ii) an evaluation of different metrics on a public dataset, including analyses by requirements complexity and comparisons across LLM configurations;
(iii) an analysis of which automatic metrics best align with expert judgment, by conducting a practitioner survey that quantifies the perceived usefulness of LLM rephrasings.

II. Background and Related Work

Numerous controlled-language templates (such as EARS, ADV-EARS, Boilerplates, SPIDER, and MASTeR) have been proposed to structure natural-language requirements. These semi-formal templates impose a fixed syntactic blueprint on each requirement statement, thereby reducing ambiguity and enforcing atomic requirements. Prior studies indicate that templates improve requirements quality versus unconstrained text: Großer et al. [2] found that using template systems can have positive effects on measures like clarity, singularity, and guideline compliance compared to free-text requirements. However, they report that most participants of a user experiment with 43 participants, mostly students, required learning and expertise to understand templated requirements [2].

A. MASTeR Template system

The MASTeR Template system [1] is a semi-formal requirements template system that structures requirements. Each requirement conforming to the MASTeR template system consists of a main template and an optional condition template. Each main template and condition template has specific keyphrases or placeholders, which can be used to identify the template used in a requirement [7].

The main template describes the core content of the requirement, while the condition template describes the conditions under which the requirement applies. The main template can be a functional or non-functional requirement. Functional requirements can be further classified as user interaction, independent system activity, or interface requirements. Non-functional requirements can be further classified into property, environmental, or process requirements.

A MASTeR requirement may also include a condition template that describes the conditions under which the requirement applies. The three condition templates are time, event, and logic conditions.

Arora et al. [14] automatically checked the conformance of requirements to template systems, such as MASTeR. They found that their approach provides a robust and accurate basis for checking conformance to templates.

B. Automated Requirements Rephrasing

Research has addressed the automated rephrasing of natural-language requirements into structured forms to improve quality and enable automated processing [15]. Early works investigated rule-based transformations that rewrite requirements according to handcrafted linguistic patterns and semantic mappings. Ambriola and Gervasi [16] used pattern matching to map requirements to a predefined ontology structure with temporal semantics. Zichler and Helke [17] applied rule-based NLP techniques to automatically transform heterogeneous requirements specifications into standardized boilerplate structures. Their approach focuses on parsing and restructuring unstructured or semi-structured requirements texts into a uniform, tool-processable format. Tiwari et al. [18] used NLP-based preprocessing and pattern analysis to parse free-text requirements and map them to predefined requirements template structures (such as EARS-style templates). The tool applies linguistic preprocessing and quality metrics to segment, normalize, and structurally align unstructured NL requirements with standard template slots, enabling semi-automated conversion into formalized requirements templates.

Only recently, the improved semantic and generative abilities of LLMs are opening up new opportunities for automatically rephrasing requirements. Norheim and Rebentisch [19] also aim to parse natural-language requirements into the EARS requirements template syntax. However, they propose to make use of LLMs to extract key phrases from natural language requirements and place them in the slots of the template syntax, whereas we generate complete requirements. Their proposed approach uses one-shot prompting for slot-filling classification. However, they only presented preliminary example results without a systematic and substantial evaluation.

Okamoto and Kusumoto [20] make use of LLMs to reorganize entire Software Requirements Specifications (SRS) into standardized structures. Their results show that while being able to convert the SRS to the intended structure, 20% of the manually checked requirements were not retained.

C. Large Language Models for Related Requirements Generation Tasks

Large language models have recently been explored for a range of RE tasks. Zadenoori et al. [21] found that studies using LLMs focus especially on requirements elicitation, validation, and analysis. Krishna et al. [6] investigate whether LLMs are capable of producing drafts of SRS. Their results indicate that LLMs can match the output quality of entry-level software engineers in drafting complete and consistent SRS.

Another direction of research aims to use LLMs to convert requirements into formal specifications [22]. For example, the Req2Spec [23] framework automatically converts requirements into formal specifications usable by HANFOR [24]. Their approach was able to successfully translate 71% of automotive requirements in a dataset by BOSCH into a formal model. Other approaches leverage LLMs to convert natural-language requirements into computation tree logic (CTL) [25], [26]. While these approaches also convert requirements, they

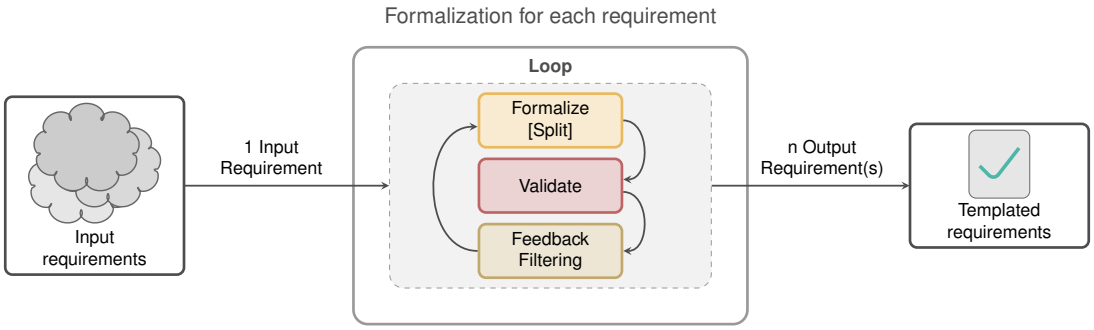

Fig. 1. Architecture overview of our approach. A simple implementation can be found in our supplementary material [13].

differ from the rephrasings analyzed in this paper by mapping requirement statements to formally defined logical semantics instead of constraining their linguistic structure within natural language.

III. Approach

The approach was developed in collaboration with Vector Informatik GmbH, a leading automotive software and tools supplier. The overall approach is designed for batch processing of entire projects in PREEvision, a model-driven engineering tool that treats requirements as model elements. Projects in this tool typically comprise thousands of hierarchically structured requirements, linked to other model elements via trace links and related mechanisms. For these linking mechanisms, it is beneficial to have atomic requirements. The MASTeR template in particular enforces strict atomification through its well-defined syntax. Additionally, the MASTeR template is commonly used by customers of our industry partner. The requirements volume and tight integration make manual rephrasing near infeasible, and the requirements splitting needed to conform to the MASTeR semi-formal template further complicates manual application.

The approach centers on an inner reflection loop, shown in Figure 1, that generates rephrased versions of individual requirements; this rephrasing component is the primary focus of this paper, as it is most interesting for the broader requirements engineering community. Additional processing steps rephrase multi-requirement results, transfer formatting and other embedded information, and generate confidence scores for user interaction. These additional steps are not evaluated in this work.

Prior work has shown that, when converting a natural language requirement into semi-formal templates, often more than one template requirement must be generated [11] (see Table I). Therefore, we allow the LLM to freely split a natural language requirement into one or more template-syntax requirements. Because the overall approach is intended for batch processing, the initial rephrasing of requirements is independent of other requirements. This is also beneficial for performance because the initial rephrasing into the template syntax can be done concurrently for each requirement. In a batch-processing setup, splitting requirements into atomic ones makes it easier to identify duplicates, which can then be merged in a subsequent step. The MASTeR template system has a high atomization level [2], which further motivates splitting the requirements into multiple template-syntax requirements.

The overall LLM workflow architecture is as a Reflection Loop [8]. Rather than attempting to fill the template slots as attempted in prior work (see Section II), we allow LLMs to generate the entire requirement in the template form. Subsequently, the rephrased requirements are validated for correct template syntax and other requirements quality criteria with score-generating validators. If a validator detects an error, the LLM generates a new requirement based on the previous version and the collected feedback. This loop can be repeated several times until the validators are satisfied with the rephrased requirement.

In the following, we describe the prompts used at each step of the loop. Constrained Decoding was used for all LLM generations with task-specific output formats. Due to space limitations, we could only include schematic versions of the prompts and output formats. All prompts and output formats used in the reflection loop and for our evaluation as well as a schematic implementation of the loop are included in our supplementary material [13].

A. System Prompt

For the prompts, we first include a system prompt that provides a detailed explanation of the MASTeR format. A schematic overview of the system prompt is shown in Figure 2, the full prompt is included in our supplementary material [13]. In the prompt, we first state the properties of good requirements. These properties are aligned to the INCOSE norm for requirement quality [9], [10]. We then explain the MASTeR template system in detail, including the various templates and their syntax. Before detailing the templates, we describe the general structure of MASTeR requirements, which is the same across all templates. For each template, we also include a partial example requirement. Lastly, in the MASTeR prompt, we also include a few complete examples of natural-language requirements and their rephrased versions in the MASTeR template syntax. In total, the system prompt is around 1400 tokens long (according to the qwen3-235b tokenizer).

TABLE I
REQUIREMENTS SET COMPARISON (MEAN ± STANDARD DEVIATION VALUES ARE REPORTED. THE DATA IS BASED ON THE REQUIREMENT SETS USED AND REPHRASED BY GROSSER ET AL. [12])

	All	Flex	ECSS	CSE	TSS	EVS
# source reqs.	249	18	33	25	63	110
# rephrased reqs.	410	41	63	58	105	143
avg split ratio	1.65	2.28	1.91	2.32	1.67	1.30
avg source char count	141.06 ± 71.41	146.61 ± 45.45	140.67 ± 42.98	256.04 ± 91.51	132.22 ± 69.45	119.21 ± 50.85
avg source word count	23.12 ± 11.57	21.83 ± 7.57	22.91 ± 9.41	40.20 ± 15.20	21.84 ± 11.31	20.24 ± 8.32

```
You are an expert in requirements engineering.
Your role is to [...] formalize them [...]
A good requirement adheres to the following
quality criteria:
  1. Necessary: [short description]
  [...]
  9. Conforming: [short description]
In the following sections different patterns will
be explained.
  [Notation conventions: [] optional, {} alternatives, <> placeholder]
  [Modal verbs: shall, should, will with definitions]
FunctionMASTeR:
  1. [...] the system <system> {shall|should|will}
     <process verb> <object> [...].
  2. [...] provide <actor> with the ability to
     <process verb> <object> [...].
  3. [...] be able to <process verb> <object> [...].
  [Examples for each flavor]
[PropertyMASTeR, EnvironmentMASTeR, ProcessMASTeR patterns with
examples]
LogicMASTeR: If <compared object> is [...], ...
EventMASTeR: As soon as the event [...], ...
TimeMASTeR: As long as [...], ...
  [Sub-patterns, examples, and complex condition rules]
[Overall NL → MASTeR input/output example pairs]
```

Fig. 2. Schematic overview of the system prompt describing the MASTeR template system. The complete prompt is in our supplementary material [13].

Following the MASTeR explanation, we include a description of the current task in the prompt. This may involve formalizing the requirement or grading it.

B. Rephrasing Prompt

For the rephrasing and splitting task, the prompt includes the concrete task for the LLM (see Figure 3). We also include detailed examples of several requirements and their rephrased versions in the MASTeR template syntax. Lastly, we give a detailed description of the structured output format.

Enforced by the structured output format, the LLM must first determine if the requirement needs an update at all, a rephrasing without splitting, or a rephrasing with splitting. The LLM has the option to create an additional heading for the rephrased requirements, allowing for creation of hierarchical structures. We do not consider such headings and treat that option just like the simple split without a heading.

The structured output format imposes no constraints on the requirement text. We force the LLM to generate a name for the requirement before generating the requirement text. The

```
[MASTeR format description — see Figure 2]
Your concrete task: [...] convert this text into a
list of properly structured requirements [...].
[Context usage, placeholder handling, naming conventions]
For the formalization, you can select one of four
possible actions:
  "no_update" | "reformulate" | "simple_split" |
"heading" [with descriptions]
```

```
{"action": str,
"requirements": [{"name": str, "content": str}],
"heading": {"heading_name": str,
"heading_text_content": str} | null}
```

Fig. 3. Schematic overview of the rephrasing prompt (appended to Figure 2). Below the rule: structured output schema.

name shall guide the LLM as a free-form abbreviation of the requirement. Summarizing the answer before generating the actual output is a common prompting technique to improve the output quality of LLMs as used by Ragas metrics [27].

C. Validators

Two validators are used in the reflective loop: a Template Validator and an INCOSE Validator. These validators are used to improve the rephrased requirements by providing feedback for specific issues.

The collected feedback from the validators is filtered before being returned to the LLM (see Figure 6). This is done to ensure that no non-fulfillable feedback is provided to the LLM. For example, if an INCOSE validator states that a specific value is required for the requirement to be unambiguous, the feedback is not provided to the rephrasing LLM to avoid forcing the LLM to hallucinate a specific value.

1) Template Validator: The Template Validator verifies the correct use of the MASTeR templates. It is based on an NLP-Tree-parsing context-free grammar similar to Arora et al.'s [14] and attempts to parse the requirement into one of the MASTeR template structures. In addition to the grammar validator, the template validator also uses an LLM-as-a-judge validator for the MASTeR syntax (see Figure 4). This LLM-based validator finally determines conformity with the MASTeR templates and generates a reason if a violation is found.

We aim to improve validation precision by combining the grammar-based and the LLM-as-a-judge-based validator, while also providing more specific feedback to the LLM. In our tests, the grammar validator has almost no false-negative

[MASTeR format description — see Figure 2]
`Determine whether a given requirement adheres to the [...] templates.`
[Attention points: modal verbs, <system> and <object> placeholders]

`{"reason": str | null, "valid": bool}`

Fig. 4. Schematic overview of the template validator prompt. Below the rule: structured output schema.

`Evaluate the following aspects:`
`- necessary:` *[desc.]* `... - conforming:` *[desc.]*
`Return a feedback comment and a score` *[...]* `between 0 and 1.`
[JSON examples, injected requirement and type]

`{"<category>": {"comment": str | null, "score": float}}`
where `<category>` $\in$ {*necessary, ..., conforming*}

Fig. 5. Schematic overview of the INCOSE validator prompt. Below the rule: structured output schema.

MASTeR template findings. This means that a grammar-fitting requirement is almost always in correct MASTeR template syntax. In contrast, the LLM-as-a-judge validator yields fewer false-positive MASTeR template findings. Additionally, the latency of the grammar validator is negligible in comparison to the LLM-as-a-judge validator.

In general, we aim to reduce false negatives because we would rather return incorrect MASTeR requirements than block the formalization process with false feedback. We use the LLM-as-a-judge validator in the hybrid setup: when the grammar validator identifies an issue, we confirm the finding and provide more specific feedback to the rephrasing LLM with the LLM-based validator.

2) INCOSE Validator: The INCOSE Validator uses LLM-as-a-judge techniques to rate the rephrased requirements based on the INCOSE norm for requirements quality [9], [10] (see Figure 5). For each of the nine individual requirement characteristics [9], [10, C1-C9], we include a short description of the meaning of the characteristic. The LLM-as-a-judge then rates the requirement on a 0-1 scale for each characteristic. Before the LLM can rate the requirement, we generate a short rating text, similar to Ragas metrics [27].

IV. Research Design

In this work, we aim to show the capabilities of LLMs in converting natural language requirements into semi-formal syntax templates as well as providing insights on how automated rephrasing approaches can be judged by quantitative metrics. Therefore, we pose the following research questions:

RQ1: How good are LLMs for rephrasing requirements into a semi-formal syntax template?

RQ2: Which metrics are suitable to evaluate the quality of a rephrased requirement against humans?

RQ3: How do requirements length and split factor influence the performance of rephrasing systems and metrics?

`Assess whether each feedback item is actionable, given requirement R and source requirements S.`
[Definitions of actionable/not actionable, example, injected R, S, and feedback]

`[{"name": str, "comment": str | null, "is_actionable": bool}]`

Fig. 6. Schematic overview of the feedback filter prompt. Below the rule: structured output schema.

TABLE II
Split factor by requirement set

Split	1	2	3	4	5	6	7	8	11
FLEX	6	5	4	2	1	0	0	0	0
ECSS	16	10	2	4	1	0	0	0	0
CSE	14	4	1	4	0	1	0	0	1
TSS	40	12	6	4	0	0	1	0	0
EVS	92	12	3	0	2	0	0	1	0
Total	168	43	16	14	4	1	1	1	1

A. Dataset

We evaluate our approach by comparing the automatically rephrased requirements with those rephrased by humans. For this paper, we focus on the MASTeR template syntax because our industry partner's rephrasing approach targets only this template.

Großer et al. rephrased 249 natural language requirements from five projects into 410 MASTeR requirements by students and researchers in a reviewed process [11].

The different requirements sets vary in complexity and structure (see Table I).

Table II shows the resulting split factors of the rephrasings done by Großer et al. Across all datasets, most requirements were rephrased as a single MASTeR requirement. The requirements of an electronic voting system (EVS) were already of high quality and had a low average word count; therefore, the rephrasing resulted in a low split ratio. In contrast, the requirements of the Certification Specifications for Engines (CSE) were less structured and, on average, Großer et al. identified one additional MASTeR requirement per source requirement compared with the EVS set.

Figure 7 shows the distribution of the source requirement length by word count. Most requirements are between 10 and 35 words long; the maximum is 70 words. Notably, the CSE requirements are predominantly longer than 35 words, whereas the EVS requirements are often shorter than 30 words.

Since a split factor of 1 is present in most cases in our data, we used requirement length as a second measure of requirement complexity in our evaluation. Word count is a commonly used metric in requirements engineering [2], [28]. Figure 8 shows the distribution of the requirement length by split factor.

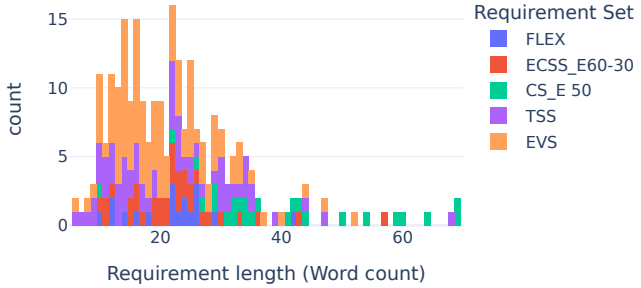

Fig. 7. Overview of the different requirement lengths by word count for the different requirements sets.

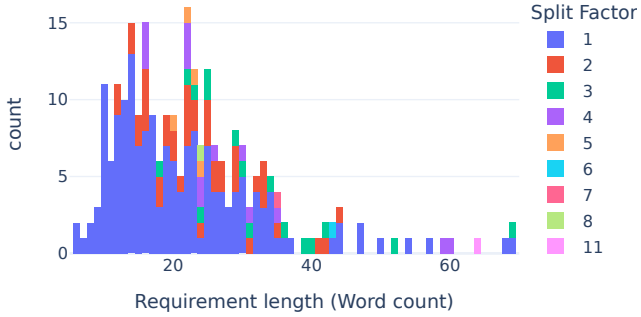

Fig. 8. Overview of the different requirement lengths by split factor.

B. Evaluated Configurations

We tested four approach configurations with three different LLMs. Additionally, if possible, we included the human-rephrased version (Human) in our tests.

- **Judge-Qwen3**: Qwen3-235B-A22B-Instruct-2507-AWQ rephrasing with GPT-OSS-120B low-thinking LLM-as-a-judge validators
- **Qwen3**: Qwen3-235B-A22B-Instruct-2507-AWQ rephrasing without validators
- **Qwen3-thinking**: Qwen3-Next-80B-A3B-Thinking rephrasing without validators
- **Judge-GPT-5**: GPT-5 rephrasing with GPT-5 LLM-as-a-judge validators
- **Human**: human rephrased requirements by Großer et al. [2]

Qwen3-235B-A22B-Instruct-2507-AWQ was the primary LLM used to develop the formalization workflow. We first evaluated the framework with all features enabled (Judge-Qwen3). For the LLM-as-a-judge validators, we used GPT-OSS-120B [29], as it was also used during development and benefited from low-thinking in the validator implementations.

To assess the LLM's ability to one-shot MASTeR requirements, we additionally evaluated the framework with all LLM-as-a-judge validators disabled (Qwen3). We further included Qwen3-Next-80B-A3B-Thinking to investigate thinking models (Qwen3-thinking), again without LLM-as-a-judge validators: Its thinking process is expected to pre-validate results, and enabling additional validators would increase latency to an impractical level for production. In this configuration, the Template Validator never returned feedback, so it effectively corresponds to a simple few-shot rephrasing setup.

Lastly, we evaluated a configuration with GPT-5 for both formalization and validation to assess closed-source LLMs (Judge-GPT-5).

C. Survey

In addition to the calculated metrics, we conducted a survey to allow human evaluators to assess the added value of LLM-based requirements rephrasing. Most of the calculated metrics we use compare LLM-generated requirements with human-rewritten requirements. While we measure properties relative to the human baseline, we cannot determine whether the LLM-generated requirements are better or worse than those rephrased by humans.

In the survey, we asked software requirements specialists from our industry partner to categorize several rephrased versions of a requirement based on their usefulness in a real-world context. For each of the 249 requirements collected and rephrased by Großer et al. [2], we presented the original requirement and five different rephrasings in the MASTeR semi-formal requirements template syntax to the survey participants. In total, the rephrasings of 103 randomly selected requirements were rated by five requirements specialists from our industry partner. The task of the participants was to categorize each of the five options by their usefulness for creating the correct requirement rephrasing:

- **No update needed (C1)**: You would accept the option as the MASTeR version of the requirement
- **Only small changes necessary (C2)**: Within a few seconds, you could fix the option to be the correct formalization
- **Useful draft for formalization (C3)**: You would need to change quite a few things, but it is better to start from this option than from the original requirement
- **No help for formalization (C4)**: The option does not help you, and you would be better off formalizing the requirement completely by yourself

We presented the 249 original requirements in a randomized order to participants, thereby ensuring broad requirements diversity while allowing participants to decide how much time to contribute.

For the rephrasing, we selected five different options: The different configurations of our approach and the human-rephrased option (see Section IV-B). By including the human-rephrased requirements from Großer et al. [2], we tried to measure the perceived usefulness of rephrasing propositions in comparison to LLM-generated rephrasing. However, we did not state in the survey that there is a human-rephrased requirement among the options, so they would not attempt to guess which option was created by humans.

D. Metrics

All metrics are calculated independently for each rephrasing of every natural language requirement. With this, we can measure the range of fluctuations not only between different

requirement sets but also within each requirement set. We evaluated several metrics to assess the quality of rephrasing relative to the ground truth of Großer et al. [11].

We normalize all metric scores to $[0-1]$. All scores are 1 if there is a perfect match between candidates and references.

1) Sentence-level metrics: In machine translation, it is common practice to compare a candidate (or hypothesis) sentence to one or multiple reference sentences. We use the LLM-generated requirements as candidate sentences and the human-rephrased requirements as references. Due to the splitting of natural-language requirements, both candidates and references can be one or multiple sentences. To get a single score, we average the scores for each candidate sentence, similar to corpus tests. We tested several sentence-based metrics:

a) BLEU score: The BLEU score is a well-established metric for evaluating machine translation [5], [30], [31]. However, the BLEU score has its limitations, such as ignoring synonyms, and researchers have shown that BLEU does not perform well outside of machine translation [32]. While still used widely, BLEU should not be the primary evaluation technique for NLP tasks [32]. Despite these limitations, we still include it to be comparable to prior requirements rephrasing research [19].

b) BERT score: BERT score uses a pretrained BERT model to measure the similarity between two sentences [33]. Comparing two sentences with BERT Score calculates precision, recall, and F_1-scores. We use the DeBERTa-XLarge-MNLI model [34], as suggested by the BERT score authors [33].

Additionally to the sentence-based evaluation, we tested BERT score by also concatenating all candidate and reference sentences to a single string respectively. Thus, we only have to compare one candidate and one reference string. We further refer to this as *source-level BERT*.

2) Embedding distance metrics: In information retrieval, there are many metrics used to compute distances between query sentences and reference knowledge. We tested three embedding-based distance metrics (cosine similarity, dot product, L2 distance), because they are commonly used [35].

We then compare scores similar to the sentence-level metrics (see Section IV-D1). We embed each LLM-rephrased requirement and each human-rephrased ground truth requirement using the Qwen3-Embedding-8B [36] embedding model and then use each distance metric to find the best-fitting reference.

In addition, we tested the embedding-based distance metrics on a source-requirement level by first concatenating all reference and candidate sentences to a single string, respectively, and measuring the distance in the embedding space.

3) Embedding Mover's Distance (EMD): Kusner et al. [37] proposed the Word Mover's Distance (WMD), which is a distance metric for texts. It computes the embeddings of the non-stop words of both texts and then computes the Earth Mover's Distance in the embedding space by defining the distance between two texts by the cumulative euclidean embedding distance between each word of text A to its nearest word in text B [37].

TABLE III
CONFUSION MATRIX OF THE MASTER MAIN AND CONDITION TEMPLATE CHECKER (THE COLUMNS ARE THE GROUND TRUTH FROM GROSSER ET AL. [11], THE ROWS ARE THE CHECKER LABELS)

Main template	EnvM	FM	PRCM	PRPM
EnvironmentMASTeR (EnvM)	8	1		
FunctionMASTeR (FM)				
- Independent System Activity		207	2	
- Interface requirement		27		
- User Interaction		87		
ProcessMASTeR (PRCM)		10	40	1
PropertyMASTeR (PRPM)		10		17
Condition template	**EM**	**LM**	**TM**	**NC**
EventMASTeR (EM)	62			10
LogicMASTeR (LM)		24	1	4
TimeMASTeR (TM)			28	17
No Condition (NC)			6	258

We use the WMD concept to develop a novel metric for comparing requirements rephrasings. In contrast to WMD, we do not embed non-stop words in documents; rather, we embed each rephrased natural-text requirement. We then calculate the WMD between the set of LLM-rephrased candidates of one source requirement and the set of human-rephrased requirements from the dataset. In short, we use the WMD by treating requirements as non-stop words. The benefit of this metric is that configurations that tend to create many requirements do not benefit from the number of requirements. To compute the embedded vectors, we again use the Qwen3-Embedding-8B [36] embedding model. The implementation of the EMD is included in our supplementary material [13].

4) Template checker: The MASTeR format comprises multiple sub-templates designed to represent different types of requirements. It distinguishes, for instance, between functional and environmental requirements and additionally provides dedicated templates for specifying requirements' conditions. These templates explicitly structure the intended target and context of a requirement, making them well-suited for evaluating whether an LLM has captured the correct requirement.

We compute the F_1-score between the rephrased requirements and the ground truth. Note that the evaluated requirements and the ground truth may consist of different numbers of requirements. The true positives are the overlapping template classifications. We calculate three distinct F_1-scores: condition template only, main template only, and exact matches of both condition and main template.

For checking the MASTeR template syntax, we implemented a template checker. Given a requirement, the checker returns the main MASTeR template and, if a condition template is used, that condition template. We do not check whether the requirement is a valid MASTeR requirement because there is some interpretive latitude in the correct use of MASTeR templates, which we acknowledge and do not want to evaluate further.

TABLE IV
SURVEY RESULTS ACROSS THE DIFFERENT REQUIREMENTS SETS AND CONFIGURATIONS

	All				FLEX				EVS				TSS				ECSSE				CSE			
	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4	C1	C2	C3	C4
Judge-Qwen3	57	11	19	16	3	0	2	1	23	4	4	5	20	6	10	5	6	0	2	3	5	1	1	2
Qwen3	49	21	22	11	3	0	2	1	18	10	5	3	20	6	12	3	4	3	1	3	4	2	2	1
Qwen3-thinking	40	24	28	11	0	1	2	3	20	8	6	2	15	9	14	3	2	4	4	1	3	2	2	2
Judge-GPT-5	41	20	19	23	2	0	2	2	14	9	4	9	20	8	7	6	1	3	4	3	4	0	2	3
Human	53	18	17	15	2	2	2	0	23	6	2	5	24	6	7	4	3	3	2	3	1	1	4	3

The template checker consists of two parts: A natural language processing-based pre-check and an LLM-as-a-judge-based post-check. The pre-check attempts to identify the template used based on keywords and key phrases in the MASTeR templates. For example, when the requirements include both the phrase ”be designed in a way” and the phrase ”can be operated”, we assume that it is an EnvironmentMASTeR requirement. Similarly, if a requirement starts with the phrase ”As long as”, we assume that it contains a TimeMASTeR condition. The post-check is used when at least one of the main or condition template cannot be identified by syntax matching only. The implementations of the Template Checker Metrics are included in our supplementary material [13].

We tested our template checker on the human rephrasings in the dataset by Großer et al. [11]. In the reproduction package of Großer et al. [12], all human-rephrased requirements are labeled with the used main template. Also, the condition template is labeled if one was used. We use those labels as ground truth for our template checker. The confusion matrices of the main template and condition template checkers are shown in Table III. Note that Großer et al. did not distinguish among the types of FunctionMASTeR requirements, so the main template checker's ground truth contains a single label for all FunctionMASTeR requirements. The condition template checker displays a ”No Condition” label when no condition template is present in the requirement.

V. RESULTS

To be able to answer our research questions, we present our findings from the survey (Section V-A) first and analyze whether any of the metrics correlate with the experts' judgments in our survey in Section V-B. Then we contextualize these findings based on the metrics (Section V-C) and analyze whether the metrics and the survey ratings are being influenced by requirements complexity.

A. RQ1: Survey Analysis

Five requirements specialists from our industry partner rated the rephrased versions of 103 randomly sampled source requirements produced by the four configurations and humans from the dataset. Some requirements were rated multiple times, so that in total 90 unique source requirements were rated.

Table IV shows the total ratings for each configuration and requirements set. Notably, for each LLM configuration, more than half of the requirements were rated C1 or C2. The LLMs were rated best for the EVS and TSS datasets, which are also the ones with the lowest average split factor (see Table I). Also, 32% of Human-rephrased requirements were rated C3 or C4.

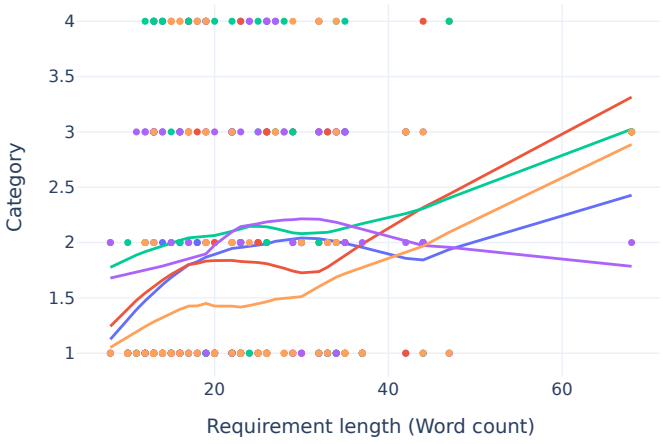

Fig. 9. Survey rank by requirement length and configuration (regression done by lowess). The configurations are Judge-Qwen3, Qwen3, Qwen3-thinking, Judge-GPT-5, and Human.

TABLE V
AVERAGE RATING CATEGORY BY SPLIT FACTOR OF THE HUMAN-REPHRASED VERSION (SF) (THE SAMPLE COUNT (SC) IS THE NUMBER OF RATINGS IN THE SURVEY. BEST AND NOT SIGNIFICANTLY WORSE CONFIGURATIONS ARE BOLD ($\alpha = .05$, P-VALUE IN BRACKETS))

SF	SC	Judge-Qwen3	Qwen3	Qwen3-thinking	Judge-GPT-5	Human
all	90	**1.93 (.97)**	**1.94 (.94)**	**2.10 (.20)**	2.26 (.03)	**1.93**
1	53	**1.80 (.86)**	**1.79**	**1.83 (.80)**	**2.12 (.07)**	**1.93 (.51)**
2	20	2.20 (.03)	**2.20 (.06)**	2.58 (.01)	2.58 (.02)	**1.65**
3+	17	**2.03**	**2.12 (.67)**	**2.38 (.29)**	**2.29 (.50)**	**2.24 (.61)**

Table V shows the average survey rank for each configuration and split factor. The configuration with the best average rating is highlighted in bold. Configurations not significantly worse than the best configuration are highlighted, too. We tested significance with the Wilcoxon signed-rank test ($\alpha = .05$) based on the hypothesis that the LLM-rephrased requirements by one configuration or the human are rated worse than the configuration rated best on average. The overall results highlight that the requirements by our two best-performing

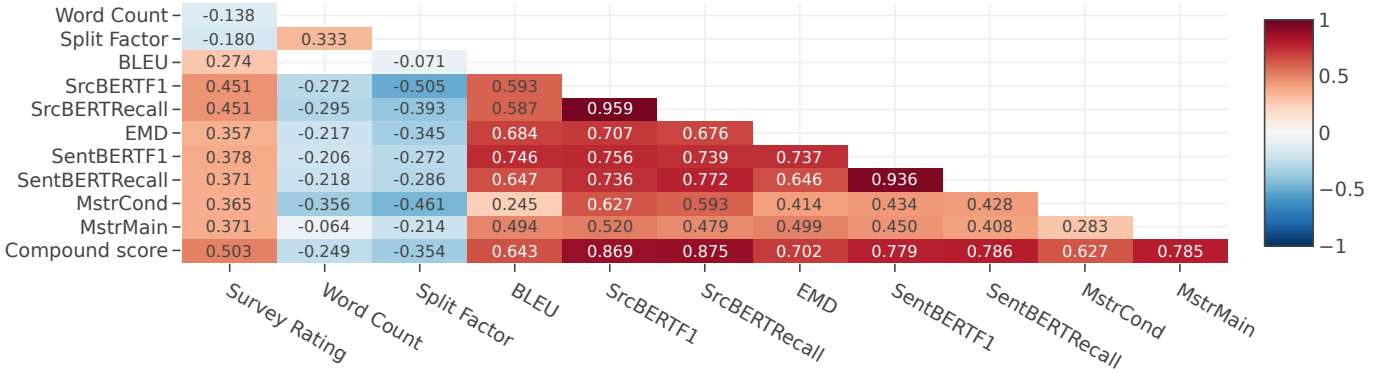

Fig. 10. Spearman correlation heatmap between different metrics. Only shows significant correlations ($\alpha = .05$).

configurations, Qwen3 and Judge-Qwen3, were rated similarly to those rephrased by humans. Only for a split factor of two, the human-rephrased version (Human) is significantly better than most LLM-generated rephrasings. Judge-GPT-5's rephrasings are the only LLM-rephrasings that overall were rated significantly worse than the human rephrasings.

In Figure 9, we show the average survey rank by word count of the source requirement. We see that requirements rephrased by humans are generally ranked better (lower category) than the LLM-generated rephrasings. Human is not always ranked C1, especially for longer requirements, which shows that either it is hard to verify correct rephrasings or that complex rephrasings are not unambiguous. We also see that, for shorter requirements, the Qwen3-235B-Instruct-based configurations (Judge-Qwen3 and Qwen3) are frequently ranked better than the other LLM-based configurations.

Answer to RQ1: We found that LLMs are capable of rephrasing requirements into template systems. In our survey, requirements rephrased by our best configuration (Qwen3) were rated C1 or C2 in 68% of cases, with only 10.7% being rated unhelpful (C4), achieving similar ratings as the human-rephrased requirements.

B. RQ2: Correlation Analysis

To be able to identify metrics that can be used as an indicator for good rephrasings to semi-formal templates, we analyze the correlation of the different metrics and the survey results. We want to show which quality metrics correlate with good survey rankings. We only selectively include those metrics in this analysis that showed interesting or good performance. The results for all tested metrics can be found in our supplementary material [13].

We use Spearman correlation [38] for our tests. For most metric correlations, we can calculate the correlation over all 249 source requirements for each of the four configurations (n=996). For the survey ranking, we calculate the correlation with the average ranking per requirement and configuration (n=360).

In Figure 10, we show the correlation between the survey rankings, the complexity metrics (source requirement length and split factor) and the calculated requirements metrics, as well as the best compound metric.

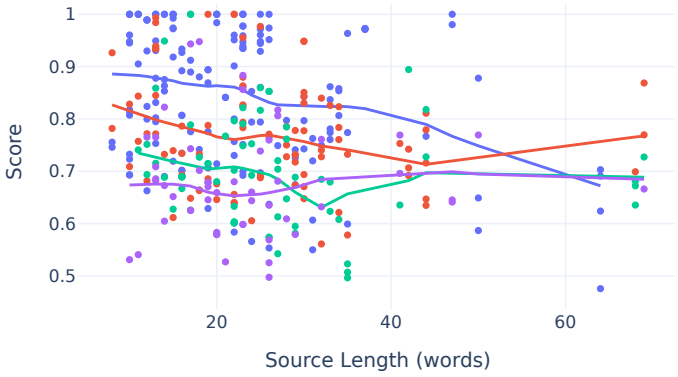

Fig. 11. Best score combination by source requirement length and survey rating (The regression is done with lowess. The categories are C1, C2, C3 and C4. The score is a combination of sentence-level BERT recall, source-level BERT recall, MASTeR Main and MASTeR Condition scores)

The correlations between metrics and the survey ratings are all positive. For the correlation tests, the survey categories were encoded by the negative number of their category (C1 $\rightarrow$ -1). Therefore, a positive correlation indicates that higher metric scores are associated with better survey rankings (i.e., lower category numbers).

Good survey ratings also correlate with low complexity. The split factor correlates slightly higher with the survey ratings than with the length.

Of all our tested metrics, the source-level BERT recall and F_1 have the highest correlation with the survey rankings (.451), followed by the MASTeR main template score (.371) as the first non-BERT metric. The embedding-based metric with the highest correlation with survey rankings is the EMD (.357).

The correlation between the MASTeR metrics and BERT scores is relatively low. This indicates that BERT and MASTeR metrics evaluate different properties of rephrased requirements, which both achieve higher survey rankings.

By testing all weighted average combinations of all scores up to weights of 10, we calculated the best compound score, which has a correlation with survey ratings of .503. We

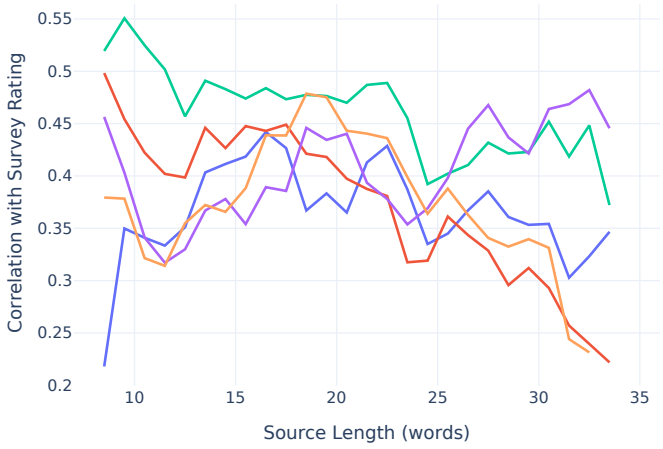

Fig. 12. Score correlation with survey ratings by source requirement length. Moving average of size 15 with at least 1/4 of samples. The scores are sentence-level BERT Recall, source-level BERT Recall, MASTeR Condition, MASTeR Main Template and EMD. All points are significant with $\alpha = .05$.

calculated the compound score as a weighted average of 7 parts source-level BERT recall, 5 parts sentence-level BERT recall, 2 parts MASTeR Main and 1 part MASTeR Condition scores. Using BERT F_1 scores instead of recall achieved similar results. While this method is prone to overfitting, it shows that combining scores of NLP-based and template-based scores achieves better overall correlation with survey rankings.

Figure 11 shows the best score combination by source requirement length and survey rating category. In addition, the figure shows the lowess smoothing trend line for each category. A higher survey ranking in general has a higher score, which supports the correlation from Figure 10. Up to source requirement lengths of 30 words, the trend line shows an ordinal separation of the average survey ranking scores, while for longer requirements, especially C1 ratings drop in score with respect to the others. However, the individual data points also show a high score variation for each survey category.

Answer to RQ2: BERT-based and template-based metrics are best suited for automatically testing LLM-based requirements rephrasing systems. However, all correlations of single scores are below .5, so the combination of NLP-based and template-based metrics should be used and especially for longer requirements complemented by human judgment.

C. RQ3: Influence of Complexity on Metrics

Comparing the different requirement sets (see Table I), we see that there are differences between the sets in terms of average split factor and requirement length. The split factor seems to correlate with the requirement length (see Figure 10).

Figure 12 shows the correlation of different scores with the survey ratings by source requirement length. BERT scores and EMD show a higher correlation with survey ratings for shorter requirements, with source-level BERT recall having the strongest correlation among all metrics for requirements under 25 words. Only the MASTeR Main Template score has a higher correlation with survey ratings for longer requirements. The MASTeR Condition score shows a low correlation for very short requirements, which is probably due to short requirements not having conditions. This might also be the reason for the higher correlation of the MASTeR Main Template score for very short requirements.

TABLE VI
CORRELATION OF DIFFERENT SCORES WITH SURVEY RATINGS BY SPLIT FACTOR (SIGNIFICANT CORRELATIONS ARE IN BOLD ($\alpha = .05$))

Split Factor	1	2	3+
Sample Count	212	80	68
Source-level BERT F_1	**0.465**	**0.463**	0.156
Source-level BERT Recall	**0.462**	**0.454**	0.141
Sentence-level BERT F_1	**0.371**	**0.338**	0.214
Sentence-level BERT Recall	**0.346**	**0.327**	**0.277**
MASTeR Condition	**0.338**	**0.410**	0.155
MASTeR Template	**0.376**	**0.390**	0.189
Embedding Mover's Distance	**0.388**	0.209	0.130

Table VI shows the correlation of different scores with survey ratings by split factor. Due to low sample sizes, we aggregate all split factors of 3 and above in one category.

The split factor heavily impacts the correlation of different metrics with the survey ratings. The only score with a significant correlation for split factor 3 and above is the sentence-level BERT Recall score. For split factor 1, only the MASTeR Condition score deviates with a lower correlation from the overall results (compare Figure 10 and Table VI).

Answer to RQ3: Requirements complexity has a moderate impact on LLM performance and a large impact on the tested metrics. LLM performance is higher and the tested metrics work better for less complex requirements.

VI. DISCUSSION

In this section, we discuss the implications of our findings for practitioners and researchers as well as address potential threats to the validity of our study.

A. Implications

For practitioners, our results imply that LLMs are well suited for rephrasing requirements to a semi-formal template, even for more complex requirements. However, the choice of the underlying model and configuration can significantly impact the quality of the rephrased requirements. Also, more complex requirements should be checked by humans in autonomous workflows.

Automated metrics should be used with caution, as their correlation with human judgment varies depending on the complexity of the requirements. For simpler requirements, automated metrics can serve as an indicator for quality assessment, but for more complex ones, human review remains essential.

For researchers, the correlation of the different metrics with human survey ratings is impacted by the complexity of the requirements. It also indicates that the metrics are

better suited for less complex requirements. Future research should investigate how to improve the metrics for more complex requirements, e.g., by using more advanced LLM-based evaluation approaches.

Finally, the observed differences in model performance across complexity indicators highlight the need for benchmark datasets that explicitly categorize requirements by complexity, enabling more fine-grained assessment of LLM-based requirements rephrasing tools. Also, common machine translation benchmark techniques should be applied to requirements rephrasing, such as using multiple reference rephrasings and more semantics-aware metrics.

B. Threats to validity

Following the guidelines of Runeson and Höst [39], we outline and address potential threats to the validity of our research and experiments.

a) Construct validity: All of our chosen metrics compare the content of LLM-rephrased requirements against human-rephrased requirements as a single ground truth. However, in the survey, the human-rephrased requirements did not achieve near-perfect scores. Especially for longer or more split requirements, the average ranking of the human-rephrased requirements lowered. Comparing rephrased requirements to a single ground truth might therefore lead to noise in the data, especially for more complex requirements. The data suggests that our metrics evaluate different properties of requirements, and there may be additional scores or metrics not considered in this study that could better capture quality. Additionally, we operationalized requirements complexity by using requirement length and split factor. However, complexity is a multi-dimensional construct that also includes semantic and logical aspects not captured by these heuristics, which may limit the extent to which our measures fully represent requirements complexity.

b) Internal validity: We evaluated only variations of a single rephrasing approach. Since the configurations shared prompts and, in some cases, architectural components, their effects cannot be considered fully independent. These shared elements may have confounded the results and contributed to the non-significant differences between the configurations. Future work should investigate different rephrasing architectures to better isolate the effects of individual configuration choices. Additionally, more complex models than correlation might be used to measure metric effectiveness.

c) External validity: We evaluated requirements rephrasings only to the MASTeR template. Other semi-formal template systems may differ in structure or constraints and could be more or less suitable for automated LLM-based rephrasing, limiting the generalizability of our findings beyond this template. Furthermore, the dataset is limited in size and diversity, which may restrict the applicability of our results to other domains, requirement types, or industrial contexts.

d) Reliability: The survey-based evaluation relies on subjective human judgment. As we did not collect specific reasons for low ratings and also could not properly calculate inter-rater agreement due to a low overlapping sample size, it is difficult to assess whether different evaluators would produce consistent results. Moreover, for longer or more complex requirements, the human-rephrased ground truth itself showed lower rankings, suggesting that the rephrasing task involves inherent subjectivity that may affect the reproducibility of the results.

VII. Conclusion

In this paper, we presented an approach for automatically converting free-text requirements into semi-formal templates with LLMs. On a dataset of 249 free-text requirements rephrased by humans to 410 MASTeR template requirements, we measured different metrics for automatically assessing such systems. In a user study comparing the LLM- and human-converted requirements, we analyzed the suitability of these metrics for automatic assessment. Our evaluation demonstrates the proficiency of LLM-based systems. LLM-converted requirements are rated similar to human-converted requirements and thus can be used to accelerate the requirements-conversion process.

BERT Score and MASTeR Template conformance are the most promising metrics for automatically evaluating the proficiency of LLM-based requirements rephrasing systems. Both metrics show a moderate correlation (.451 and .371, respectively) with human ratings of the rephrased requirements. The combination of both types of scores achieved the highest correlation to human judgment (.503).

In general, all tested metrics showed a stronger correlation with human judgment for shorter, less complex requirements. This indicates that the proficiency of LLM-based requirements rephrasing systems is higher for shorter, less complex requirements. More research is needed to improve the proficiency of LLM-based requirements rephrasing systems for more complex requirements. Finding or combining better-suited metrics for evaluating the proficiency of LLM-based requirements rephrasing systems may increase the ability to test systems automatically and thus facilitate their usage in practice.

Data Availability Statement

All used prompts, constrained decoding output formats, some metric implementations, a simple implementation of the approach and all requirements with all metric scores and survey ratings can be found in our supplementary material [13].

The prompts contain some minor changes compared to the ones used in the implementation, which only extends to specific MASTeR template examples.

Due to confidentiality agreements with our industry partners, we cannot share the code of the approach.

Acknowledgment

This work was funded by the German Research Foundation (DFG) - SFB 1608 - 501798263 and Core Informatics at KIT (KiKIT) of the Helmholtz Assoc. (HGF) and supported by KASTEL Security Research Labs.

References

[1] T. R. Company, "Sophist master patterns knowledge library," https://www.reusecompany.com/wp-content/uploads/2020/08/SOPHIST-Master-Patterns-Library.pdf, 2020.

[2] K. Großer, A. S. Ahmadian, M. Rukavitsyna, Q. Ramadan, and J. Jürjens, "Benchmarking requirement template systems: Comparing appropriateness, usability, and expressiveness," *Requirements Engineering*, vol. 29, no. 4, pp. 481–522, Dec. 2024.

[3] ISO, "Road vehicles – functional safety," ISO 26262:2011, 2011.

[4] RTCA, Inc. and EUROCAE, "DO-178C: Software considerations in Airborne Systems and Equipment Certification," Standard, Washington, DC, 2011.

[5] W. Zhu, H. Liu, Q. Dong, J. Xu, S. Huang, L. Kong, J. Chen, and L. Li, "Multilingual machine translation with large language models: Empirical results and analysis," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 2765–2781. [Online]. Available: https://aclanthology.org/2024.findings-naacl.176/

[6] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using LLMs in Software Requirements Specifications: An Empirical Evaluation," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, Jun. 2024, pp. 475–483.

[7] C. Rupp and R. Joppich, "Anforderungsschablonen," in *Requirements Engineering und-Management, 6th ed.* Carl Hanser Verlag München, 2014, pp. 215–246.

[8] Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung, "Towards Mitigating LLM Hallucination via Self Reflection," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1827–1843.

[9] I. R. W. Group *et al.*, "Incose guide to writing requirements v3. 1–summary sheet," in *INCOSE*, 2022.

[10] INCOSE, *INCOSE Guide to Writing Requirements*. INCOSE Publications Office, 2023.

[11] K. Großer, M. Rukavitsyna, and J. Jürjens, "A comparative evaluation of requirement template systems," in *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE, 2023, pp. 41–52. [Online]. Available: https://ieeexplore.ieee.org/document/10260862/

[12] K. Großer and M. Rukavitsyna, "K-grosser/Evaluation-of-templates-for-requirements-documentation: Replication Package for Article in Requirements Engineering Journal," Zenodo, Aug. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13343495

[13] J. Roßkothen, D. Fuchß, F. Erdösi, M. Floruß, J. Keim, and T. Hey, "Supplementary material: On converting natural language requirements into semi-formal templates using llms," Jun. 2026. [Online]. Available: https://doi.org/10.5281/zenodo.20698022

[14] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[15] I. Darif, G. El Boussaidi, S. Kpodjedo, and C. Politowski, "Controlled natural language for requirements specification: A systematic literature review," *ACM Comput. Surv.*, vol. 58, no. 7, Jan. 2026. [Online]. Available: https://doi.org/10.1145/3778169

[16] V. Ambriola and V. Gervasi, "On the Systematic Analysis of Natural Language Requirements with CIRCE," *Autom Software Eng*, vol. 13, no. 1, pp. 107–167, Jan. 2006.

[17] K. Zichler and S. Helke, "R2BC: Tool-Based Requirements Preparation for Delta Analyses by Conversion into Boilerplates," in *Software Engineering*, 2019.

[18] S. Tiwari, P. Shah, and M. Khare, "NL2RT: A Tool to Translate Natural Language Text into Requirements Templates (RTs)," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*, Aug. 2022, pp. 262–263.

[19] J. J. Norheim and E. Rebentisch, "Structuring natural language requirements with large language models," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, 2024, pp. 68–71.

[20] R. Okamoto and S. Kusumoto, "Towards the automatic restructuring of software requirements specifications to conform to standards using large language models," in *2025 IEEE 33rd International Requirements Engineering Conference (RE)*, 2025, pp. 467–475.

[21] M. A. Zadenoori, J. Dabrowski, W. Alhoshan, L. Zhao, and A. Ferrari, "Large language models (llms) for requirements engineering (re): A systematic literature review," 2025. [Online]. Available: https://arxiv.org/abs/2509.11446

[22] A. Beg, D. O'Donoghue, and R. Monahan, "A short survey on formalising software requirements using large language models," 2025. [Online]. Available: https://arxiv.org/abs/2506.11874

[23] A. Nayak, H. P. Timmapathini, V. Murali, K. Ponnalagu, V. G. Venkoparao, and A. Post, "Req2spec: Transforming software requirements into formal specifications using natural language processing," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2022, pp. 87–95.

[24] S. Becker, D. Dietsch, N. Hauff, E. Henkel, V. Langenfeld, A. Podelski, and B. Westphal, "Hanfor: Semantic Requirements Review at Scale," in *Joint Proceedings of REFSQ-2021 Workshops, OpenRE, Posters and Tools Track, and Doctoral Symposium*, Essen, Germany, Apr. 2021.

[25] M. Zhao, R. Tao, Y. Huang, J. Shi, S. Qin, and Y. Yang, "NL2CTL: Automatic Generation of Formal Requirements Specifications via Large Language Models," in *Formal Methods and Software Engineering*, K. Ogata, D. Mery, M. Sun, and S. Liu, Eds. Singapore: Springer Nature, 2024, pp. 1–17.

[26] R. Zrelli, H. A. Misson, M. Ben Attia, F. G. de Magalhaes, A. Shabah, and G. Nicolescu, "Advancing Formal Verification: Fine-Tuning LLMs for Translating Natural Language Requirements to CTL Specifications," in *2024 International Workshop on Rapid System Prototyping (RSP)*, Oct. 2024, pp. 21–27.

[27] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, N. Aletras and O. De Clercq, Eds. St. Julians, Malta: Association for Computational Linguistics, Mar. 2024, pp. 150–158. [Online]. Available: https://aclanthology.org/2024.eacl-demo.16/

[28] J. Frattini, L. Montgomery, J. Fischbach, M. Unterkalmsteiner, D. Mendez, and D. Fucci, "A live extensible ontology of quality factors for textual requirements," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 2022, pp. 274–280.

[29] OpenAI, "gpt-oss-120b & gpt-oss-20b model card," 2025. [Online]. Available: https://arxiv.org/abs/2508.10925

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[31] I. Rivera-Trigueros, "Machine translation systems and quality assessment: a systematic review," *Language Resources and Evaluation*, vol. 56, no. 2, pp. 593–619, 2022.

[32] E. Reiter, "A structured review of the validity of bleu," *Computational Linguistics*, vol. 44, no. 3, pp. 393–401, 2018.

[33] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020. [Online]. Available: https://arxiv.org/abs/1904.09675

[34] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," 2021. [Online]. Available: https://arxiv.org/abs/2006.03654

[35] H. Elkiran and J. Rasheed, "Evarag: Evaluating advanced rag techniques with indexing and distance metrics," *IEEE Access*, vol. 13, pp. 215 724–215 747, 2025.

[36] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin, F. Huang, and J. Zhou, "Qwen3 embedding: Advancing text embedding and reranking through foundation models," *arXiv preprint arXiv:2506.05176*, 2025.

[37] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 957–966. [Online]. Available: https://proceedings.mlr.press/v37/kusnerb15.html

[38] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904. [Online]. Available: http://www.jstor.org/stable/1412159

[39] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, 2008.