The Tale of the Broken Clock: How LLMs Evaluate Requirement Quality and Where They Fail

Abstract

Requirements engineering (RE) plays an integral role in successful systems engineering (SE) as the vast majority of design rework and cost overruns are attributed to poor requirements. Assuring requirement quality remains a time-consuming and expertise-intensive activity. There is growing interest in leveraging artificial intelligence (AI) to support RE, especially with the broad availability of Large Language Models (LLMs) that work well with natural language, in which requirements have historically been defined. However, LLMs are stochastic tools, and the empirical evidence regarding the performance and reliability of LLMs in evaluating requirements quality is nascent. To address this gap, this study evaluates how well an off-the-shelf LLM evaluates requirement quality for a notional engineered system using INCOSE-derived criteria. We assess LLM performance by comparing it against a human-expert generated ground truth, characterizing its performance and variability of errors across independent runs. This study makes three important contributions. First, it provides the first quantitative evidence of the performance of LLM-based requirement quality evaluation, documenting substantially higher performance in avoiding false alarms compared to reliably detecting quality issues. Second, it documents pronounced variability in LLM behavior, suggesting both its performance and manifestation of AI errors vary drastically. Third, it reveals differences in detection performance for different requirement quality categories. More specifically, LLMs perform poorly in detecting issues that require SE judgement, such as correctness, necessity, and boundedness, and exhibit more reliable performance in detecting linguistically explicit issues such as ambiguity and Verifiability. Overall, these findings indicate that off-the-shelf LLMs are not yet suitable for autonomous requirement quality evaluation; yet could be more suitable to assist human experts or serve specialized roles in agentic AI frameworks. Overall, this work provides empirical grounding for AI-assisted RE quality assessment performance along with its errors and potential risks; highlighting the importance of trustworthy adoption of AI for SE.

Keywords

Artificial Intelligence for Systems Engineering (AI4SE), Requirements, Large Language Models (LLMs), Trustworthy AI

Introduction and Background

Requirements in Systems Engineering (SE) play an integral role as they define the technical problem to be solved in a solution agnostic manner (Bahill & Madni, 2017; INCOSE, 2023), thus translating the ill-defined problem into a well-defined one that is amenable to solving (Simon, 1973). Given that complex system development is a sequential process with highly coupled task interdependencies (Simon, 1995; Hazelrigg, 1998), the elicitation of requirements constrains the trajectory of downstream activities by locking in resources and shaping the development effort. Consequently, requirements engineering (RE) emerged as a subfield of SE to deal with the elicitation, analysis, and management of requirements through the system lifecycle. Numerous books have been published for the RE area (Pohl, 1996; Nuseibeh & Easterbrook, 2000; Macaulay, 2012), and several international standards have been formalized (IEEE, 2008, 2018).

Nevertheless, many of the SE challenges continue to originate from requirements-related problems (Aurum & Wohlin, 2005). For instance, incomplete or missing information in requirement sets is not discovered until much later in the lifecycle; and once discovered they are extremely costly to correct (Jayatilleke & Lai, 2018; Morkos et al., 2012). As such, requirement correction efforts lead to significant design changes and rework, and are documented to be a leading cause of program delays and cost overruns (Boehm & Papaccio, 1988; Peña & Valerdi, 2015).

AI for SE and RE

Given these challenges, advancing artificial intelligence (AI) capabilities and their use for SE (AI4SE) emerge as a promising avenue (McDermott et al., 2020; Allison et al., 2022). AI4SE research expanded rapidly in recent years, ranging from formulation of knowledge databases (Hirtz et al., 2002; Kitamura et al., 2004), virtual assistants (Bang et al., 2018; i Martin & Selva, 2019a; Islas-Cota et al., 2022), and design evaluators (Fu et al., 1997; Guerlain et al., 1999). Today, significant research is being conducted on both human-AI dyads (Singh & Szajnfarber, 2025) and teams of humans with AI advisors (Gyory et al., 2021; Song et al., 2022). Findings suggest that AI assistance can improve design performance, particularly if collaboration continues for extended periods of time (Viros I Martin & Selva, 2022). Others investigated AI-assistance during design decision-making and found that designer self-confidence plays a critical role in their trust in AI, error attribution, and eventual adoption of AI guidelines (Chong et al., 2022a). Although these are promising advances, effectiveness of AI in assisting SE tasks remains an open research avenue (Topcu et al., 2025; Zhang et al., 2021).

Despite this recent momentum, RE has been one of the earlier adopters of AI4SE research in part because requirements are traditionally expressed in natural human language (e.g., English), and the software engineering community - who are also deeply concerned with RE - led the development of natural language processing algorithms (NLPs). NLP aims to render natural human language palatable to computers through executing a combination of tasks such as information retrieval, text classification, and language generation (Weizenbaum, 1966; Jurafsky & Martin, 2008). While earlier NLP lacked contextual awareness (Hayes-Roth, 1985); their transition to learning-based methods enabled better prediction of word sequences (Baker, 1975; Jelinek et al., 1977). A key breakthrough in NLP performance was the use of deep learning methods (Elman, 1990) that enabled selective retaining information for longer periods of time (Hochreiter & Schmidhuber, 1997; Sutskever et al., 2014). Finally, the introduction of transformer models, such as large language models (LLMs) NLPs exhibited a significant performance leap (Ashish, 2017).

Combined with the wide-spread availability of text-based RE data, LLMs significantly accelerated the use of AI for RE. LLMs fundamentally differ from recurrent network-based NLP approaches as they rely on self-attention mechanisms, which allow them to process sequences of information in parallel, as opposed to the sequential approach of neural networks. Currently, AI4RE pursues a plethora of objectives, including defining specifications (Dalpiaz et al., 2018), classification and extraction of requirements types from other engineering documents (Hey et al., 2020; Sainani et al., 2020), establishing traceability (Guo et al., 2017; Lin et al., 2021), identification of user needs (Han & Moghaddam, 2021), and knowledge extraction (Berquand et al., 2021).

Recent review studies (Norheim et al., 2024; Sonbol et al., 2022; Zhao et al., 2022) summarize AI4RE initiatives into four categories: requirements generation, quality assessment, analysis, and translation. Generation focuses on extracting templates from common products (Reinhartz-Berger & Kemelman, 2020) or mining end-user reviews to identify a legacy system's ability to meet its user requirements (De Araújo & Marcacini, 2021). Others look into formulation of conversational AI assistants (Arora et al., 2023);

however, most of these studies prioritize appearance over quality (Habib et al., 2025).

Quality assessment research is unexpectedly nascent, yet it focuses on both individual requirements and requirement sets. In the case of individual requirements, quality assessment may focus on detecting ambiguities, adherence to requirement writing guidelines, or syntactic structures. In the case of the requirement set, quality assessment may focus on identifying missing requirements (i.e., incompleteness), redundancies, and conflicts. Existing work on sets focuses on semantic similarities between requirements to identify redundancies and conflicts between two or more requirements (Malik et al., 2024). Others have experimented with GPT-4 to identify ambiguities, conflicts, and inconsistencies in requirement sets, and have documented that its precision varies significantly – ranging between 41% to 61%; suggesting off-the-shelf approaches are promising but risky (Mahbub et al., 2024). Research on individual requirements uses unsupervised training mechanisms to assess completeness (Luitel et al., 2023).

There is a vast body of research in requirements analysis. These mostly focus on classification tasks that are useful, but arguably less valuable in preventing SE shortcomings (Berquand et al., 2021; Tikayat Ray et al., 2023; Zhao et al., 2022; Zhu et al., 2024). Another stream of analysis research is in traceability. Transformer-based work in this area suggests better performance compared to decision tree-based classifiers if large amounts of high-quality annotated training data are available (Deshpande et al., 2021). Others developed consistency checks, seeking if requirements contained causal statements (Fischbach et al., 2021). Rahimi et. al. developed a method for linking requirements to downstream design artifacts (i.e., design documentation and test cases) for improved traceability (Rahimi et al., 2014).

Finally, translation studies pursue to convert a given set of requirements into formal languages that are more suitable for machine processing (Cosler et al., 2023; Hahn et al., 2022). The core idea here is to transition from human language-based requirements to truly model-based ones (Salado & Wach, 2019), a pursuit that is also recognized by the current SysML v2 transition efforts (Bajaj et al., 2022), which could significantly improve SE practice by eliminating ambiguities and human errors.

The Gap Addressed in This Study

The existing literature on AI4RE has three major limitations. First, existing work is predominantly based on software systems, as opposed to engineered systems that have both hardware and software, as it's typically seen in most SE programs. This is a drastic difference as compared to software, engineered systems are (i) bounded by both the laws of physics and engineering, (ii) need to be manufactured, hence are costly to conduct rework; thus, (iii) are more complex (Hennig et al., 2021). Second, the current performance of AI4RE tools is less than perfect, with significant error rates. Recent research in this area benchmarked multi-modal LLMs for understanding reference engineering documentation, textual requirements, and CAD drawings (Doris et al., 2024); and documented that there are significant shortcomings in terms of reliability and performance. This is particularly true for more concerning false positives (Type I) and false negatives (Type II) errors. Third, although LLMs are stochastic tools, there is an increasing pressure in the SE community to explore agentic-AI approaches (Jiang et al., 2025; Massoudi & Fuge, 2025), with the notion that agents, some of which could be LLM-based, could serve specific roles that can be coordinated for better intelligent design performance. While this may be a fruitful endeavor, and fine-tuning and specialization may indeed help, LLMs are inherently stochastic tools (Huang et al., 2023; Ofsa & Topcu, 2025). Hence, their performance and insights are non-deterministic, which may exacerbate the reliability challenge of both specialized LLMs and agentic approaches.

To that end, this paper pursues the following research questions: how well does an off-the-shelf LLM perform in terms of evaluation of requirements quality, and how do its error rates vary for various INCOSE RE quality criteria? We pursue this

question by creating a purposefully poor-quality sample requirement set for a notional SE problem, creating human-expert generated ground-truth evaluation of this set, and then generating an automated workflow that asks ChatGPT 4.o to assess this requirement using INCOSE's criteria. We then analyze the manifestation of true positives, true negatives, false positives, and false negatives in AI-generated requirements evaluation; along with how these rates vary across independent AI runs.

Our findings are significant, as to the best of our knowledge, they provide the first quantitative evidence of the performance of an off-the-shelf LLM for requirements quality evaluation, along with the distribution of its error rates. The results demonstrate higher performance in avoiding false alarms than in correctly detecting existing quality issues. Specifically, the LLM exhibits relatively low Type I error rates, i.e., incorrectly flagging requirements as problematic, but substantially higher Type II error rates, i.e., missing existing quality issues. The results also reveal substantial variation across repeated LLM evaluation attempts. These findings shed light on where LLM-based tools can meaningfully support RE without further modifications, what their limitations are, and in which aspects they would need to be supplemented for more sophisticated SE use cases. Overall, the study contributes empirical grounding to ongoing discussions on the trustworthiness of AI, its role in SE, and the pressing need for a human-in-the-loop approach for RE.

Methodology

This study follows a multi-step methodology illustrated in Figure 1 to generate, evaluate, and analyze AI-based evaluation of requirement quality. Our approach begins with requirement set development for a notional SE problem. Next, a ground truth is generated through an expert review of the requirement set against specified evaluation criteria. After that, an automated workflow is established to collect data on AI evaluation of the requirement set in multiple independent runs. Finally, a quantitative analysis is conducted to assess the AI performance in evaluating requirement quality and the variability of its error rates by comparing its performance against the ground truth. We discuss each step in detail in the following subsections.

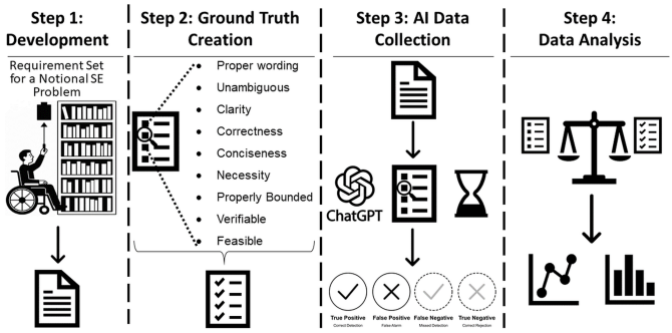

Figure 1. Methodology Overview

Step 1 Requirement Set Generation

To develop the requirement set, we used a notional SE problem and a set of requirements developed by researchers at the University of Texas at Dallas (UT Dallas). In this previous study, UT Dallas researchers gave a guest lecture on requirements to 1,208 pre-service engineers with no prior SE education and provided them with a notional SE problem on a book retrieval system for people in wheelchairs. The SE problem included information on the system's context, along with a brief Concept of Operations (ConOps). The participants were then instructed to work in various team sizes to generate as many requirements in a 20-minute time period as possible. The objective of the study understand the impact of team size on requirements variety (Nonso-Anyakwo & Summers, 2025b) and quantity (Nonso-Anyakwo & Summers, 2025a) .

UT Dallas study resulted in a large array of poor requirements, given the lack of SE training of the subjects and the limited problem formulation time. Nevertheless, it provides a large pool of human-generated requirements with a rich representation of human errors; that rendered it suitable for the purposes of this study. We then randomly selected 20 requirements from this set to facilitate easier ground truth generation, and categorized them following Salado's work on Max-Neef's existential categories that enable a minimum set of categories

to define a set of system requirements (Salado & Nilchiani, 2014). This helped organize the 20 requirements in the set into functional, performance, environmental, and resource requirements.

Step 2 Ground Truth Generation

In order to create a basis of comparison, the research team evaluated each of the requirements based on the evaluation criteria presented in Table 1. It is important to note that for this phase of the study, the research team only focused on evaluating individual requirements as a first step towards addressing our research questions; while recognizing that requirements are only meaningful as a set. We defer the analysis of the requirement set of future work.

Table 1. Requirement Quality Evaluation Criteria modified from INCOSE Handbook

Quality Criteria	Definition
Proper Format	Contains the word "Shall" follows by an imperative verb.
Unambiguous	Quantitative terms should be measurable, and qualitative terms should be clearly defined.
Clarity	Wording has to be clear and easy to understand.
Correctness	Statement needs to be justifiable and linked with needs, goals, objectives, or technological/program limitations; Its fulfillment satisfies the associated stakeholder(s).
Conciseness	The statement should, with sufficient detail, define only one thing.
Necessity	The statement should not exist if not needed and it should not be redundant with other existing requirements.
Proper Bounding	Each of the quantifiable words within the requirement statement should either be defined or captured through a reference to another requirements.
Verifiability	The requirement statement is worded such that its realization can be verified to the approving authority's satisfaction.
Feasibility	The requirement statement should make technical sense and be possible to achieve.

As listed in Table 1, a modified version of the INCOSE criteria was used for its ability to capture a broad range of quality aspects for a given requirement statement. The evaluation process involved having the research team read each requirement in from the lens of the problem context and ConOps, and identify which (if any) of the evaluation criteria had not been fulfilled, and document the shortcomings of each unmet criterion. In order to eliminate bias and intercoder error, the research team members each reviewed the requirements independently, then compared results, ultimately agreeing upon a single evaluation for each requirement. This set was used as the ground truth and serves as the basis for comparing AI-generated evaluations.

Step 3 AI Data Collection

To systematically evaluate how an off-the-shelf LLM assesses requirement quality and how this assessment varies, an automated workflow was developed. The intended objective of this workflow is to generate LLM-based evaluations of the requirement set produced in Step 1 by systematically assessing each requirement against the evaluation criteria defined in Step 2. However, in addition to the evaluation criteria listed in Table 1, two supplementary categories were included: No issues found with the requirement and Unsure of Category. The No Issues category captures cases where a requirement was assessed as satisfying all evaluation criteria. Unsure of Category accounts for the cases where an identified issue could not be confidently mapped to any predefined criterion. These two additional categories enable a more transparent requirement evaluation without forcing ambiguous cases into existing criteria. The workflow is configured in the n8n platform and illustrated in Figure 2. The following paragraphs

describe the workflow's inputs, model configuration, evaluation logic, and output artifact.

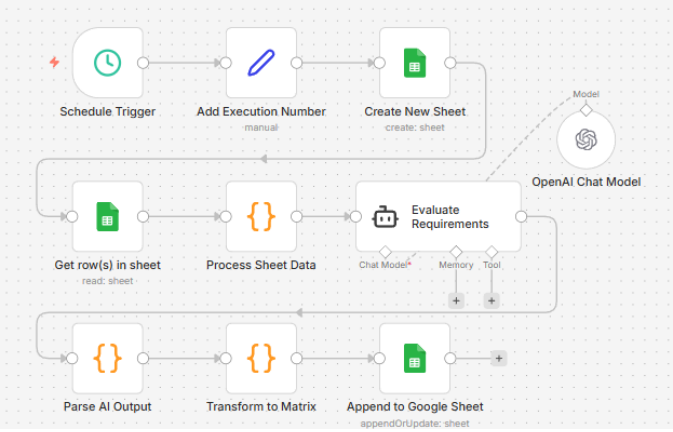

Figure 2. AI Experiment Workflow

The input that was entered into the automated workflow was a Google sheet containing the requirements, a description of the system to be evaluated, and a ConOps diagram. The Google sheet contained the requirements and project description in the form of structured row-formatted data. The structured data requirements are formatted through JSON code nodes in the automated workflow and then fed into the AI code node.

For the AI model, "ChatGPT 4.o" textual model was used. While newer versions of ChatGPT are available, ChatGPT 4.o was the newest version available on the n8n automation platform. Within the automated workflow, the AI was prompted to evaluate if a problem exists for each of the requirements by using its internal knowledge as well as the problem-relevant information (ConOps and Project Description text). AI-generated binary flags if an issue is detected with the requirement, which was captured in a formatted matrix. This matrix was exported in a Google Doc, where each requirement and issue criterion corresponds to a column and a row, respectively.

Since LLMs generate non-deterministic outputs, the workflow was configured to iterate 100 times using independent models to capture their variability. Additionally, the baseline temperature was set to 0.7 in order to capture the nominal distribution of AI outputs. AI sampling temperature is a feature of an LLM that determines the variability of response, with lower temperature resulting in a lower amount of variability in the response. This allowed us to investigate the variability of AI-generated quality assessment by comparing its performance across the independent runs.

Step 4 Analyzing the Findings

We employed the following procedure to analyze findings. First, for each run, AI-generated outcomes in Step 3 were compared against the ground truth that was generated in Step 2. Each comparison resulted in one of four possible outcomes: correct detection, correct non-detection, false detection, or missed detection. Next, building on this pairwise comparison, each AI decision was classified using standard confusion matrix logic (Foody, 2002; Heydarian et al., 2022) as follows:

- True Positive (TP): AI indicates a quality issue that is also captured in the ground truth.
- True Negative (TN): AI does not indicate an issue, and the issue also does not exist in the ground truth.
- False Positive (FP): AI indicates a quality issue when the ground truth indicates no issue. This is also known as a Type I error or a false alarm (Robinson et al., 1998).
- False Negative (FN): AI does not indicate a quality issue when the ground truth indicates an issue. This is also known as a Type II error or a miss (Robinson et al., 1998).

Finally, to enable fair comparison across quality issue categories, we normalized performance rates rather than reporting raw counts. Four normalized metrics were computed as follows (Luque et al., 2019):

True Positive Rate (TPR): TPR measures detection sensitivity by quantifying the proportion of issues correctly identified by the LLM. Higher TPR values indicate better quality issue detection performance.

$$TPR = \frac{TP}{Ground\ Truth\ Total\ Positve}$$

True Negative Rate (TNR): TNR measures avoidance capability by quantifying the proportion of non-issues correctly left unflagged. Higher TNR values indicate stronger resistance to false alarms.

$$TNR = \frac{TN}{Ground\ Truth\ Total\ Negative}$$

False Positive Rate (FPR): FPR quantifies the proportion of non-issues incorrectly flagged as issues. Lower FPR values indicate better false-alarm control or lower Type I error rate.

$$FPR = \frac{FP}{Ground\ Truth\ Total\ Negative}$$

False Negative Rate (FNR): FNR quantifies the proportion of issues that were missed by the LLM. Lower FNR values indicate stronger quality issue detection capability or lower Type II errors.

$$FNR = \frac{FN}{Ground\ Truth\ Total\ Positve}$$

These selected metrics correspond to standard confusion-matrix measures, where TPR is equivalent to recall and reflects correct issue detection, and TNR corresponds to specificity and reflects correct non-issue identification (Luque et al., 2019). FPR and FNR capture complementary error rates associated with incorrect issue flagging and missed issue detection, also known as Type I and Type II errors, respectively (Cuellar, 2025).

Once the TPR, TNR, FPR, and FNR were calculated for each of the AI decisions, overall metric distributions across all 100 AI quality evaluation executions were examined using boxplots to characterize central tendency, dispersion, and outliers. Then, the analysis was further decomposed at the issue-category level, enabling identification of issue types where the LLM performs consistently well, struggles more often, or exhibits highly variable behavior. It is important to note that variability across executions is an inherent characteristic of stochastic AI inference rather than experimental error, as our experimental data generation procedure was identical across the runs. Consequently, findings are interpreted based on the patterns observed across distributions instead of isolated best- or worst-case outcomes.

Findings

This section presents the findings from the quantitative analysis to examine the performance of AI in evaluating requirement issues. The findings are presented using boxplots to illustrate the distribution of performance metrics across multiple runs, both in the entire set (Figure 3) and later in each of the issue categories (Figures 4-7). The y-axis lists the normalization metrics and requirement issues in Figure 3 and Figures 4-7, respectively. In these figures, the x-axis represents the value of the corresponding normalized performance metric, bounded between 0.00 and 1.00, reflecting the full range from no occurrences to 100% occurrence. Each boxplot shows the distribution of a given metric, where the maroon median line indicates typical performance, the interquartile range captures variability across executions, whiskers show the spread of non-outlier values, and isolated points denote outliers.

AI's Performance in Evaluating Requirement Quality

Figure 3 summarizes the overall distribution of normalized metrics across executions for the requirements, revealing a clear asymmetry in performance in terms of issue detection and avoiding false alarms. The most pressing insight from Figure 3 is that none of the detection metrics exhibit perfect performance. Even in the best-performing cases, there are significant error rates, and some instances exhibit substantially poor performance.

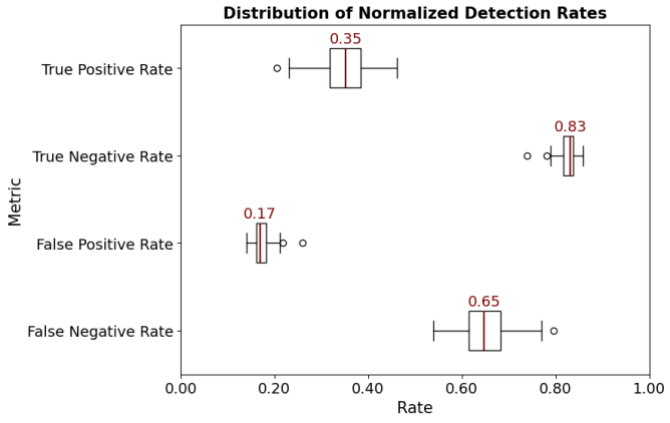

Figure 3. Distribution of Requirement Quality Issue Detection Rates

To elaborate, the median TPR is relatively low at 0.35, suggesting the LLM accurately catches only 35% of existing issues correctly. This is also reflected by the substantially higher FNR (Type II Errors) with a median of 0.65, indicating that a majority of known requirement quality issues are missed consistently, which is concerning. In terms of variance, both TPR and FNR exhibit comparatively wide distributions, with TPR ranging from 0.21 to 0.46 and FNR ranging from 0.54 to 0.79. This suggests that even under the most favorable execution conditions, the TPR is less than 0.50, indicating that fewer than half of the actual quality issues are detected at best. This spread is also reflected in their standard deviations of 0.05 for both metrics, indicating both poor accuracy and high execution-level variability in identifying actual quality issues. Additionally, the TPR distribution exhibits a mild left skew, suggesting that while detection performance is consistently low, a limited number of executions achieve relatively higher detection performance. The FNR distribution shows a slight right skew, with a longer upper tail, indicating a higher probability of missed detection.

In contrast, the model demonstrates more stable performance in avoiding false alarms. The median TNR is high at 0.83, indicating that the LLM correctly leaves most non-issues unflagged. In contrast, the median FPR (Type I Error) of 0.17 highlights that 17% non-issues are incorrectly flagged as issues. Although this error rate may seem relatively low, a 17% false alarm rate is significant, as unnecessary rework and mistrust in AI could potentially result from these false alarms. TNR values range from 0.74 to 0.86, and FPR values range from 0.14 to 0.26. Both metrics exhibit narrow interquartile ranges and significantly lower standard deviations of 0.02, indicating consistent behavior across executions. The boxplots show mild left skew for TNR and right skew for FPR. This indicates that most executions achieve consistently high true negative performance with occasional lower-performing runs, while false positive rates are generally low but occasionally increase under less favorable execution conditions.

While Figure 3 captures overall detection behavior, it does not reflect which quality issue categories drive this performance or where the LLM struggles most. Therefore, we proceed to a discussion of quality issue-level distribution for TPR, TNR, FPR, and FNR to enable a more granular examination of detection strengths and failure modes across issue types.

True Positive Rates per Issue

Before we present issue-level figures, it is important to note that some categories do not display boxplots for certain figures. This occurred when no such instances existed in the ground truth for that issue, meaning the corresponding rate is mathematically undefined. For example, if a "Clarity" issue did not exist in the ground truth, the denominator for calculating the TPR/FNR becomes zero, making these rates undefined – hence resulting in no boxes being plotted. In contrast, categories whose boxplots collapse to a single line at 0.00 or 1.00 indicate well-defined and near-invariant outcomes across executions. For example, “Conciseness” in TPR appears as a collapsed box at 0.00, meaning that in our data collection, AI never detected the issues of this type. In summary, these collapsed distributions indicate uniform behavior across executions, whereas the absence of a box indicates a lack of evaluable cases for that category.

Figure 4 presents the distribution of TPR for each issue category. Among the categories, Ambiguity and Verifiability exhibit the highest TPR rate.

Ambiguity has a median TPR of 0.69 and a mean of 0.73, with values ranging from 0.46 to 0.92 and a standard deviation of 0.09, indicating both strong overall performance and moderate variability across executions. This category displays an asymmetric boxplot with significantly extended upper whiskers, indicating occasional executions with substantially higher detection rates relative to the median. Verifiability has a median TPR of 0.64, a mean of 0.63, and a range of 0.36 to 0.82, accompanied by a larger standard deviation of 0.15, reflecting greater variability. This indicates a mild asymmetry, with a modest skew toward lower values.

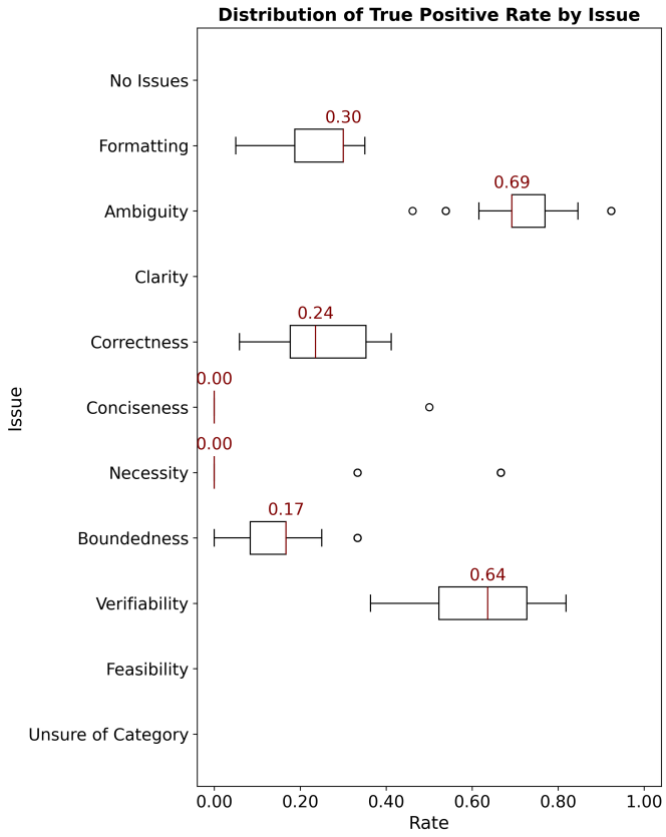

Figure 4. TPR Distribution by Issue

Formatting, Correctness, and Boundedness demonstrate more limited detection performance. Formatting has a median TPR of 0.30 and a mean of 0.25, with a range of 0.05 to 0.35 and a standard deviation of 0.10, suggesting moderate issue detection capability with execution-level variability. Correctness exhibits a median TPR of 0.24 and a mean of 0.23, spanning 0.06 to 0.41, and a standard deviation of 0.09. Boundedness exhibits a median TPR of 0.17 and a mean of 0.16, and a narrower range of 0.00 to 0.33. The relatively small standard deviation of 0.07 suggests more consistent, but consistently poor detection performance across executions. The boxplots of Formatting and Boundedness exhibit significant left skew, indicating a greater likelihood of substantially lower detection rates relative to the median. In contrast, the boxplot of Correctness shows mild right skew, with longer upper tails indicating that higher detection rates occur in a subset of executions.

Several issue categories demonstrate near-zero detection. Conciseness and Necessity both have median TPR values of 0.00, with mean values of 0.01 and 0.06, respectively. This occurs despite the presence of multiple presence of quality issues in ground truth for each category: two Conciseness issues and three Necessity issues. This minimal detection capability indicates that issues related to these categories are rarely identified. Finally, No Issues, Clarity, Feasibility, and Unsure of Category had no true positive occurrences in the ground truth, reflected by the absence of box distributions.

True Negative Rates per Issue

Figure 5 presents the distribution of TNR for each requirement issue category. Overall, Figure 5 shows substantially higher TNR values than TPR across most categories, indicating much stronger performance in issue avoidance than issue detection. Conciseness, Feasibility, and Unsure of Category show median TNR values of 1.00, with mean values above 0.97 and low standard deviations ($\leqslant 0.05$), indicating strong and stable avoidance of false positives. Necessity also demonstrates strong avoidance behavior, with a median TNR of 0.88, a mean of 0.86, and a relatively narrow range (0.71–1.00). This category shows relatively moderate variability with a standard deviation of 0.07. The boxplot displays left skew, indicating predominantly high true negative rates with a strong lower bound.

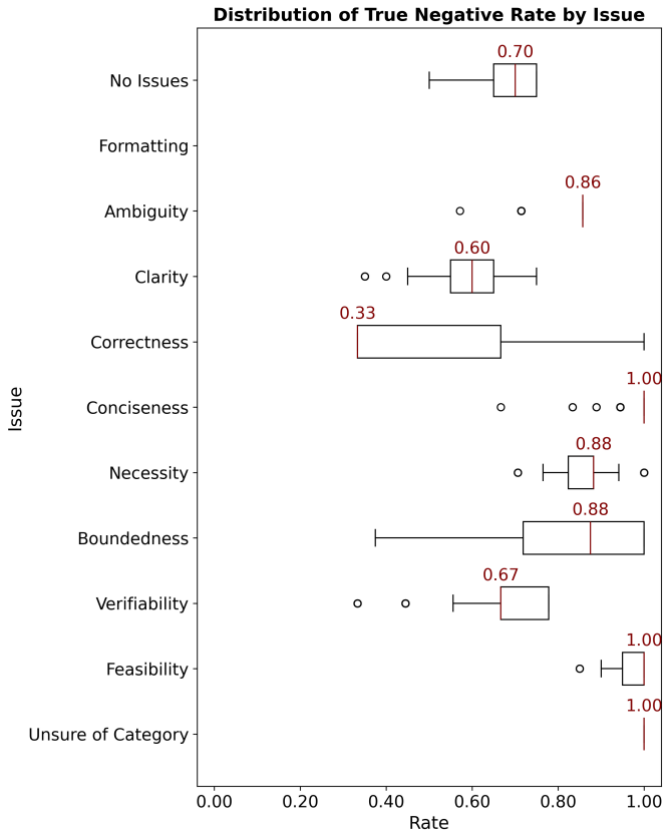

Figure 5. TNR Distribution by Issue

Ambiguity and Boundedness likewise show high TNR values, with median values of 0.86 and 0.88, respectively. Ambiguity exhibits a relatively tight distribution (0.57 – 0.86, standard deviation 0.06), and the boxplot is collapsed, indicating stability. Boundedness shows a wider range (0.38 –1.00) and a larger standard deviation (0.17), indicating greater variability across executions. The boxplot displays mild asymmetry, with extended lower tails reflecting occasional executions with reduced true negative performance.

Moderate TNR performance is observed for No Issues, Clarity, and Verifiability. No Issues has a median TNR of 0.70 and a mean of 0.69, while Clarity and Verifiability exhibit median values of 0.60 and 0.67, respectively. These categories show broader interquartile ranges and moderate standard deviations (0.09–0.13), indicating less consistent avoidance behavior relative to the higher-performing categories. The distributions No Issues and Clarity are approximately symmetric, while the distribution for Verifiability shows pronounced right skew.

Correctness exhibits the weakest true negative performance, with a median TNR of 0.33 and a mean of 0.44, alongside a wide range (0.33 – 1.00) and a comparatively large standard deviation of 0.16. The highly asymmetric boxplot, characterized by a long upper tail, indicates that while some executions achieve strong avoidance performance, many perform substantially worse. Finally, all requirements in the requirement set were intentionally written with formatting issues, and therefore, there were no true negative instances for Formatting in the ground truth dataset. As a result, Formatting shows no box distribution in Figure 5.

False Positive Rates per Issue

Figure 6 presents the distribution of FPR (i.e., Type I Errors) across executions for each requirement issue category; incorrectly flagging an issue when it does not exist. Overall, FPR values vary considerably by issue type, revealing notable differences in false alarm behavior across categories. Conciseness, Feasibility, and Unsure of Category all have median FPR values of 0.00, with mean values below 0.03 and low standard deviations ($\leqslant$ 0.05), indicating that these issues are rarely falsely identified across executions. The boxplots also show stable behavior. Necessity also demonstrates relatively low false positive behavior, with a median FPR of 0.12, a mean of 0.14, and a narrow range (0.00–0.29), suggesting generally stable avoidance of spurious detections. The boxplot shows a right skew, indicating predominantly low false positive rates with occasional high false detections.

Moderate false positive behavior is observed for Ambiguity and Boundedness. Ambiguity has a median FPR of 0.14 and a mean of 0.17, with values spanning 0.14–0.43 and a standard deviation of 0.06, indicating occasional increases in false alarms across executions. This issue has a collapsed box plot indicating stable outcomes. On the other hand, Boundedness exhibits a median FPR of 0.12 and a mean of 0.19, with a much broader range (0.00–

0.63) and a higher standard deviation (0.17), reflecting greater variability and a more asymmetric distribution, characterized by an extended upper tail.

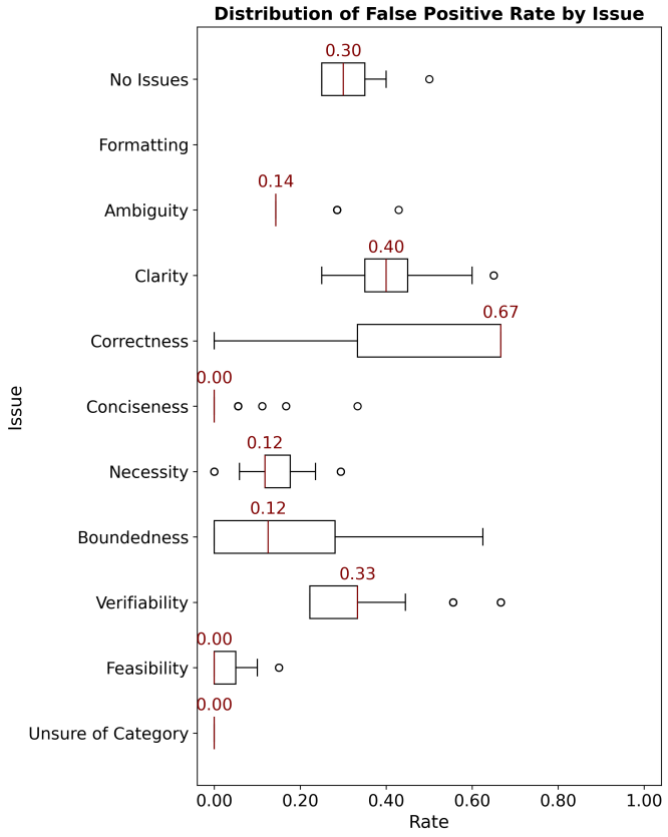

Figure 6. FPR Distribution by Issue

Higher false positive rates are observed for Clarity, Verifiability, and No Issues. Clarity shows a median FPR of 0.40 and a mean of 0.41, while Verifiability exhibits a median of 0.33 and a mean of 0.34, both indicating frequent false alarms for these categories. No Issues has a median FPR of 0.30 and a mean of 0.31, reflecting that the LLM incorrectly labeled requirements as issue-free in ~30% of evaluation attempts despite each requirement containing at least one issue in the ground truth. The boxplots for No Issues and Clarity are fairly symmetric, while Verifiability demonstrates noticeable asymmetry with a longer lower tail.

Finally, Correctness exhibits the highest false positive rates among all categories, with a median FPR of 0.67 and a mean of 0.56, alongside a wide range (0.00 – 0.67) and a large standard deviation (0.16). The strongly asymmetric distribution, characterized by a long lower tail, indicates that while some executions avoid false alarms, many exhibit frequent incorrect detections. Formatting shows no box distribution, reflecting the absence of applicable false positive instances for this category in the ground truth dataset.

False Negative Rates per Issue

Figure 7 presents the distribution of FNR (i.e., Type II Errors) across executions for each requirement issue category, illustrating how frequently true issues are missed by the model.

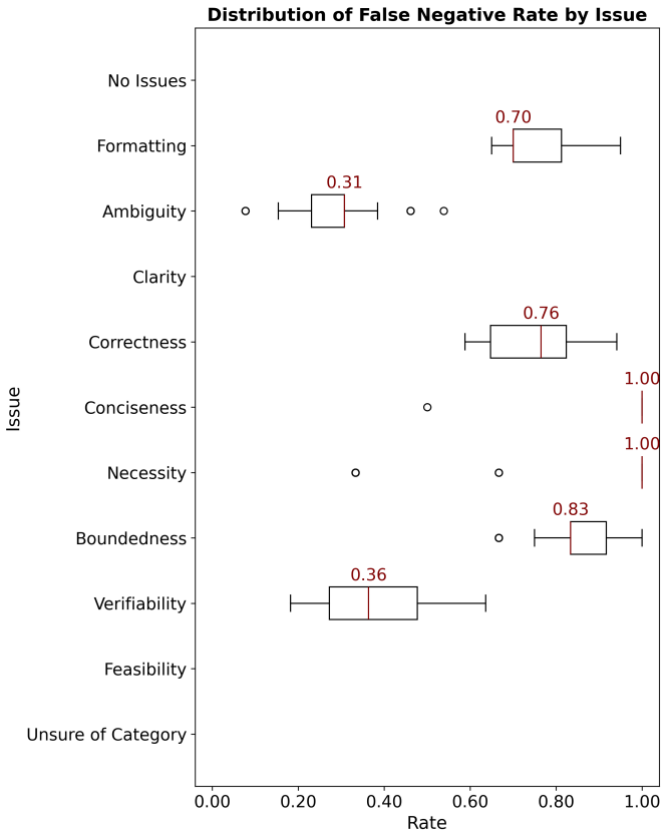

Figure 7. FNR Distribution by Issue

Several issue categories exhibit consistently high false negative behavior. Conciseness and Necessity both show median FNR values of 1.00 with collapsed boxplots, with mean values of 1.00 and 0.94, respectively, indicating that these issues are almost always missed when present. Boundedness also demonstrates high FNR, with a median FNR of 0.83, a mean of 0.84, and a range (0.67 – 1.00), suggesting consistently poor detection capability with

moderate variability. The boxplot shows right skew, indicating predominantly high false negative rates with occasional executions exhibiting near-complete missed detection.

Formatting and Correctness similarly exhibit elevated false negative rates. Formatting has a median FNR of 0.70 and a mean of 0.75, with values ranging from 0.65 to 0.95 and a standard deviation of 0.10. Correctness shows a median FNR of 0.76 and a mean of 0.77, spanning 0.59 – 0.94 with a standard deviation of 0.09. Relatively lower false negative rates are observed for Ambiguity and Verifiability. Ambiguity exhibits a median FNR of 0.31 and a mean of 0.27, with a range of 0.08 to 0.54 and a standard deviation of 0.09, indicating more frequent successful detection relative to other categories. Verifiability shows a slightly higher median FNR of 0.36, a mean of 0.37, and a broader range (0.18–0.64), accompanied by a larger standard deviation (0.15), reflecting greater variability. The boxplots for these four issue categories display pronounced asymmetry, further indicating uneven performance across iterations.

Finally, No Issues, Clarity, Feasibility, and Unsure of Category show no box distributions, reflecting the absence of applicable false negative instances for these categories in the ground truth.

Discussion

The objective of this study is to evaluate how well an off-the-shelf LLM assesses requirement quality for engineered systems, with particular emphasis on the error rates and their variability across INCOSE requirement quality criteria. Consequently, this study provides empirical insights on where the LLM performs relatively well and where limitations persist in requirements quality evaluation. The following discussion synthesizes results across aggregate and issue-level analyses to describe observed LLM performance, execution-level variability, and their implications for AI-assisted RE practice.

The findings collectively reveal a pronounced imbalance between detecting quality issues and avoiding false alarms in LLM-based requirements evaluation. At an aggregate level, the LLM shows substantially low TPR and high FNR, indicating concerningly poor performance in identifying true quality issues. In contrast, the LLM demonstrates higher TNR and lower FPR, which indicates significantly better performance in avoiding false alarms. This imbalance suggests that the AI behaves conservatively, with a pattern of avoiding incorrectly flagging issues at the cost of missing many actual issues that are present. Nevertheless, FPR rates are significantly high with ~17%, when combined with low TPR rates of ~35%, indicating that all LLM-provided quality assessments should be treated with caution.

Since different issues pose distinct challenges in SE practice, the issue-level analyses provide insight into where LLM is most effective and where additional review effort may be required in requirements evaluation. Our issue-level findings show that this imbalance is not uniform across requirement quality issue categories.

Categories such as Ambiguity and Verifiability exhibit consistently higher TPR and lower FNR, indicating that the model is comparatively effective at identifying quality issues related to unclear language or unverifiable statements. Arguably, these are easier to detect by a novice engineer as they are manifested through subjective language, missing measurable parameters, or the absence of verification constructs. which likely contributes to their comparatively stronger performance on these criteria.

In contrast, the findings suggest that quality issues related to categories such as Correctness, Conciseness, Boundedness, and Necessity often go undetected by the LLM. Assessing Correctness and Necessity requires judgment that depends on integrating information across the problem statement, ConOps, along with SE knowledge. The observed poor performance suggests that the LLM struggles substantially to establish these contextual and purpose-driven linkages; which are essential for

successful SE. This limitation is consistent with both a lack of domain-specific engineering judgment and the absence of broader situational understanding that human experts routinely apply when evaluating whether a requirement should exist at all. Conciseness assessment requires identifying whether a requirement contains unnecessary scope or overlaps with other requirements. Similarly, Boundedness often depends on references to external knowledge or cross-requirement constraints. The consistently poor detection performance in these categories suggests that such issues are not easily inferred from isolated textual analysis without a broader understanding. We find these to be expected yet really concerning, given that these kinds of issues could lead to the wrong problem being formulated, and consequently, poor engineering decisions being made with significant downstream correction activities. Reliance on AI guidance in these areas may consequently lead to the exacerbation of existing SE shortcomings that are manifested in schedule and cost overruns.

An interesting observation arises from the relatively low detection rate of Formatting-related issues, which is particularly surprising given that all requirements in the requirement set were intentionally written with poor formatting. Despite this, the LLM identified only about 30% of Formatting cases. This suggests that even seemingly straightforward requirement quality issues may not be consistently captured by the LLM.

The issue-level FPR and TNR analyses support earlier discussion on overall better performance in avoiding false alarms compared to issue detection. High TNR values and low FPR values across many categories indicate that the LLM is generally reliable when asserting the absence of issues. Particularly, Conciseness, Feasibility, and Unsure of Category demonstrate almost 100% true negative rate, meaning that false alarms almost never happen in these categories. We suspect this is also a function of the specific system of interest that was used in this study, as it is relatively simple compared to most complex SE programs. It is worth mentioning that the LLM never assigned "Unsure of Category" to any requirement across all evaluation attempts. This behavior is aligned with the ground truth, which contained no instances for this category. This indicates that the LLM reliably avoids incorrectly introducing uncertainty classifications in requirement evaluation when no uncertainty exists in the input data.

However, elevated FPR values for categories such as Correctness, Clarity, and Verifiability suggest that when the model produces false alarms, they are concentrated in categories that require understanding and establishing a connection to stakeholder intent, domain context, or justification beyond surface-level and isolated text features. This limitation is in line with the earlier discussion on the LLM's poor performance resulting from reliance on surface-level textual cues and a lack of understanding of the broader problem context. In addition to this, none of the metrics approaches perfect performance across executions, even in the most favorable cases. Beyond performance trends, the findings show notable variability in LLM behavior across executions, even when the underlying model, prompt structure, and evaluation criteria remain unchanged. This variability reflects the inherently probabilistic nature of LLM inference, where LLM-generated requirements evaluations are influenced by stochastic sampling processes even under identical inputs and evaluation settings.

Taken together, these findings indicate that off-the-shelf LLMs are currently far from supporting fully autonomous requirements quality evaluation in a trustworthy manner. This reinforces the necessity of a human-in-the-loop review process, consistent with prior work on human-AI collaboration in design and decision-making (Gyory et al., 2021; Song et al., 2022; Viros i Martin & Selva, 2022). Rather than replacing expert judgment, LLM-based tools should be positioned as decision-support aids that complement human expertise. This collaboration should help prioritize analyst attention, reduce review burden, and highlight likely problem areas while preserving human

authority over final decision (Chong et al., 2022b; i Martin & Selva, 2019b; Viros i Martin & Selva, 2022). Effective adoption of such AI-assisted workflows should consider prioritizing human trust in AI guidance, which prior studies show is shaped by factors such as accuracy, transparency, fairness, and robustness of the LLM system (Lotfalian Saremi et al., 2025).

The insights from this study have several important implications for future research in AI4RE. While off-the-shelf LLMs demonstrate promise for supporting early-stage evaluation, their limitations in detecting broader context-dependent quality issues highlight the need for continued investigation. This is also in line with previous findings (Husain et al., 2024; Topcu et al., 2025); however, it exemplifies more deeply how these arguments also hold true for RE. Future AI4RE research should explore approaches that integrate richer system context, structured domain knowledge, and multi-artifact reasoning to improve the detection of issues such as Necessity, Conciseness, and Boundedness. Additionally, given the observed variability in LLM evaluation. Further work is needed to pursue human-AI collaborative workflows that balance automation with expert oversight. Together, these directions point toward AI systems that support, rather than replace, systems engineers for enhanced transparency, robustness, and trust in requirements evaluation processes.

Conclusions

This study examined how an off-the-shelf LLM evaluates requirement quality using INCOSE-aligned criteria. By comparing LLM-generated evaluations against a human-expert generated ground truth for a notional SE problem, the study provides insights into detection performance, its variance, along with spread of its error rates. To our best knowledge, this represents the first empirical assessments of LLM-based requirements quality evaluation applied to an engineered systems context

The results show a clear imbalance in performance: the LLM demonstrates relatively good performance in avoiding false alarms but consistently struggles to identify many existing requirement quality issues. Nevertheless, there is a high proportion of Type I errors, and most of the missed true issues pertain to crucial quality criteria such as Correctness, Boundedness, and Necessity; suggesting AI-generated insights could lead to poor SE decisions. Furthermore, we document that detection performance varies substantially across issue categories, with higher effectiveness for linguistically explicit issues and poor performance for categories requiring contextual reasoning, cross-requirement integration, or engineering judgment. Pronounced variability across executions highlights the probabilistic nature of LLM inference and the limited reliability of single-run evaluations. These findings indicate the limitations of the off-the-shelf LLMs in fully autonomous requirements quality assurance. However, they show promise as decision-support tools within a carefully tailored human-in-the-loop workflow.

This study has several limitations that need to be acknowledged. First, the analysis is based on a single, purposefully poor-quality requirement set; for an arguably simple engineered system. This may limit broader generalizability to other domains, requirement maturity, and system complexity levels. Second, the evaluation relies on a single off-the-shelf LLM with baseline temperature settings, which may not reflect the capabilities of other LLM models, their variations, and potentially the best-case use of AI4RE configurations. It has been documented that various prompting techniques and specialization could lead to increased performance; thus, our findings should be taken as a documentation of the upper bound of these deviations. Third, the study focuses on binary issue detection at the individual requirement level. Future work investigates how this performance varies within the requirement set, ideally for requirement sets of varying quality. In addition, domain-grounded finetuning may be pursued to improve the LLM's ability to reason about contextual and system-level requirement quality issues. Finally, studies on human–AI collaboration may be conducted to determine how LLMs can most

effectively augment SE judgment in requirements quality assurance.

In conclusion, we hope this paper serves as a cautionary tale on trustworthy use of AI, with insights regarding to what extent we can rely on AI recommendations for RE, their variance, and how existing foundational LLM capabilities could be utilized to move forward for more effective AI4SE.

Acknowledgements

This research was supported by the National Nuclear Security Administration (NNSA) through Systems Engineering Research Center (SERC) Contract WRT-2416: Systems Engineering Beyond the Horizon.

References

Allison, J. T., Cardin, M.-A., McComb, C., Ren, M. Y., Selva, D., Tucker, C., Witherell, P., & Zhao, Y. F. (Eds.). (2022). Special Issue: Artificial Intelligence and Engineering Design. *Journal of Mechanical Design*, *144*(2), Article 2. https://doi.org/10.1115/1.4053111

Arora, C., Grundy, J., & Abdelrazek, M. (2023). *Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs* (No. arXiv:2310.13976). arXiv. https://doi.org/10.48550/arXiv.2310.13976

Ashish, V. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, *30*, I.

Aurum, A., & Wohlin, C. (2005). Requirements Engineering: Setting the Context. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 1–15). Springer. https://doi.org/10.1007/3-540-28244-0_1

Bahill, A. T., & Madni, A. M. (2017). Discovering System Requirements. In A. T. Bahill & A. M. Madni, *Tradeoff Decisions in System Design* (pp. 373–457). Springer International Publishing. https://doi.org/10.1007/978-3-319-43712-5_4

Bajaj, M., Friedenthal, S., & Seidewitz, E. (2022). Systems Modeling Language (SysML v2) Support for Digital Engineering. *INSIGHT*, *25*(1), 19–24. https://doi.org/10.1002/inst.12367

Baker, J. (1975). The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *23*(1), 24–29.

Bang, H., Virós Martin, A., Prat, A., & Selva, D. (2018). Daphne: An intelligent assistant for architecting earth observing satellite systems. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 1366).

Berquand, A., Darm, P., & Riccardi, A. (2021). Spacetransformers: Language modeling for space systems. *IEEE Access*, *9*, 133111–133122.

Boehm, B. W., & Papaccio, P. N. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*, *14*(10), Article 10.

Chong, L., Zhang, G., Goucher-Lambert, K., Kotovsky, K., & Cagan, J. (2022a). Human confidence in artificial intelligence and in themselves: The evolution and impact of confidence on adoption of AI advice. *Computers in Human Behavior*, *127*, 107018. https://doi.org/10.1016/j.chb.2021.107018

Chong, L., Zhang, G., Goucher-Lambert, K., Kotovsky, K., & Cagan, J. (2022b). Human confidence in artificial intelligence and in themselves: The evolution and impact of confidence on adoption of AI advice. *Computers in Human Behavior*, *127*, 107018.

Cosler, M., Hahn, C., Mendoza, D., Schmitt, F., & Trippel, C. (2023). *nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models* (No. arXiv:2303.04864). arXiv. https://doi.org/10.48550/arXiv.2303.04864

Cuellar, M. (2025). The problem with eliminations: Why forensic comparisons need false negative rates. *Forensic Science International: Synergy*, *11*, 100621. https://doi.org/10.1016/j.fsisyn.2025.100621

Dalpiaz, F., Ferrari, A., Franch, X., & Palomares, C. (2018). Natural language processing for requirements engineering: The best is yet to come. *IEEE Software*, *35*(5), 115–119.

Daniel Jurafsky & James H. Martin. (2008). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall. https://scholar.google.com/citations?view_op=view_citation&hl=en&user=uZg9l58AAAAJ&citation_for_view=uZg9l58AAAAJ:2osOgNQ5qMEC

De Araújo, A. F., & Marcacini, R. M. (2021). RE-BERT: Automatic extraction of software requirements from app reviews using BERT language model. *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 1321–1327. https://doi.org/10.1145/3412841.3442006

Deshpande, G., Sheikhi, B., Chakka, S., Zotegouon, D. L., Masahati, M. N., & Ruhe, G. (2021). Is bert the new silver bullet?-an empirical investigation of requirements dependency classification. *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 136–145. https://ieeexplore.ieee.org/abstract/document/9582338/?casa_token=j-6oc_eL0KcAAAAA:YPm2_y1dJJS8SVI_Dv0dimcjKq3d3RBMUOmkT6iyF59FX-9oMpch0p_Kimf0KQJO5zxDLICUWa0

Doris, A. C., Grandi, D., Tomich, R., Alam, M. F., Ataei, M., Cheong, H., & Ahmed, F. (2024). *DesignQA: A Multimodal Benchmark for Evaluating Large Language Models' Understanding of Engineering Documentation* (No. arXiv:2404.07917). arXiv. http://arxiv.org/abs/2404.07917

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Fischbach, J., Frattini, J., & Vogelsang, A. (2021). *CiRA: A Tool for the Automatic Detection of Causal Relationships in Requirements Artifacts* (No. arXiv:2103.06768). arXiv. https://doi.org/10.48550/arXiv.2103.06768

Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, *80*(1), 185–201.

Fu, M. C., Hayes, C. C., & East, E. W. (1997). SEDAR: Expert critiquing system for flat and low-slope roof design and review. *Journal of Computing in Civil Engineering*, *11*(1), 60–68.

Guerlain, S. A., Smith, P. J., Obradovich, J. H., Rudmann, S., Strohm, P., Smith, J. W., Svirbely, J., & Sachs, L. (1999). Interactive critiquing as a form of decision support: An empirical evaluation. *Human Factors*, *41*(1), 72–89.

Guo, J., Cheng, J., & Cleland-Huang, J. (2017). Semantically enhanced software traceability using deep learning techniques. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 3–14. https://ieeexplore.ieee.org/abstract/document/7985645/

Gyory, J. T., Soria Zurita, N. F., Martin, J., Balon, C., McComb, C., Kotovsky, K., & Cagan, J. (2021). Human Versus Artificial Intelligence: A Data-Driven Approach to Real-Time Process Management During Complex Engineering Design. *Journal of Mechanical Design*, *144*(2), Article 2. https://doi.org/10.1115/1.4052488

Habib, M. K., Graziotin, D., & Wagner, S. (2025). *ReqBrain: Task-Specific Instruction Tuning of LLMs for AI-Assisted Requirements Generation* (No. arXiv:2505.17632). arXiv. https://doi.org/10.48550/arXiv.2505.17632

Hahn, C., Schmitt, F., Tillman, J. J., Metzger, N., Siber, J., & Finkbeiner, B. (2022). *Formal Specifications from Natural Language* (No. arXiv:2206.01962). arXiv. https://doi.org/10.48550/arXiv.2206.01962

Han, Y., & Moghaddam, M. (2021). Eliciting attribute-level user needs from online reviews with deep language models and information extraction. *Journal of Mechanical Design*, *143*(6), 061403.

Hayes-Roth, F. (1985). Rule-based systems. *Communications of the ACM*, *28*(9), 921–932. https://doi.org/10.1145/4284.4286

Hazelrigg, G. (1998). A Framework for Decision-Based Engineering Design. *Journal of Mechanical Design*, *120*(4), Article 4. https://doi.org/10.1115/1.2829328

Hennig, A., Topcu, T. G., & Szajnfarber, Z. (2021). So You Think Your System Is Complex?: Why and How Existing Complexity Measures Rarely Agree. *Journal of Mechanical Design*, *144*(4), Article 4. https://doi.org/10.1115/1.4052701

Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2020). Norbert: Transfer learning for requirements classification. *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 169–179. https://ieeexplore.ieee.org/abstract/document/9218141/

Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-label confusion matrix. *Ieee Access*, *10*, 19083–19095.

Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., & Wood, K. L. (2002). A functional basis for engineering design: Reconciling and evolving previous efforts. *Research in Engineering Design*, *13*, 65–82.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Huang, Y., Song, J., Wang, Z., Zhao, S., Chen, H., Juefei-Xu, F., & Ma, L. (2023). Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv Preprint arXiv*:*2307*.*10236*.

Husain, M., Wach, P., & Topcu, T. G. (2024). Can Large Language Models Accelerate Digital Transformation by Generating Expert-Like Systems Engineering Artifacts? Insights from an Empirical Exploration. *Conference on Systems Engineering Research*, 371–385. https://link.springer.com/chapter/10.1007/978-3-031-62554-1_23

i Martin, A. V., & Selva, D. (2019a). Daphne: A virtual assistant for designing earth observation distributed spacecraft missions. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 30–48.

i Martin, A. V., & Selva, D. (2019b). Daphne: A virtual assistant for designing earth observation distributed spacecraft missions. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 30–48.

IEEE. (2008). *ISO/IEC/IEEE 15288:2008 International Standard—Systems and software engineering System life cycle processes*. IEEE. http://ieeexplore.ieee.org.ezproxy.lib.vt.edu/servlet/opac?punumber=6093921

IEEE. (2018). *IEEE/ISO/IEC 29148-2018 Systems and software engineering—Life cycle processes—Requirements engineering*. https://standards.ieee.org/ieee/29148/6937/

INCOSE. (2023). *INCOSE systems engineering handbook*. John Wiley & Sons. https://books.google.com/books?hl=en&lr=&id=HhjBEAAAQBAJ&oi=fnd&pg=PR9&dq=Guide+to+Writing+Requirements+incose&ots=hPDzRao1nR&sig=mG4PIX63Hcafz02u3TM6dGhe7EU

Islas-Cota, E., Gutierrez-Garcia, J. O., Acosta, C. O., & Rodríguez, L.-F. (2022). A systematic review of intelligent assistants. *Future Generation Computer Systems*, *128*, 45–62.

Jayatilleke, S., & Lai, R. (2018). A systematic review of requirements change management. *Information and Software Technology*, *93*, 163–185.

Jelinek, F., Mercer, R. L., Bahl, L. R., & Baker, J. K. (1977). Perplexity—A measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, *62*(S1), S63–S63.

Jiang, S., Xie, M., Chen, F. Y., Ma, J., & Luo, J. (2025). Intelligent Design 4.0: Paradigm Evolution Toward the Agentic Artificial Intelligence Era. *Journal of Computing and Information Science in Engineering*, *25*(12), 120808.

Kitamura, Y., Kashiwase, M., Fuse, M., & Mizoguchi, R. (2004). Deployment of an ontological framework of functional design knowledge. *Advanced Engineering Informatics*, *18*(2), 115–127.

Lin, J., Liu, Y., Zeng, Q., Jiang, M., & Cleland-Huang, J. (2021). Traceability transformed: Generating more accurate links with pre-trained bert models. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 324–335. https://ieeexplore.ieee.org/abstract/document/9402118/

Lotfalian Saremi, M., Ziv, I., Asan, O., & Bayrak, A. E. (2025). Trust, Workload, and Performance in Human–Artificial Intelligence Partnering: The Role of Artificial Intelligence Attributes in Solving Classification Problems. *Journal of Mechanical Design*, *147*(1), 011702.

Luitel, D., Hassani, S., & Sabetzadeh, M. (2023). Using Language Models for Enhancing the Completeness of Natural-Language Requirements. In A. Ferrari & B. Penzenstadler (Eds.), *Requirements Engineering: Foundation for Software Quality* (Vol. 13975, pp. 87–104). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-29786-1_7

Luque, A., Carrasco, A., Martín, A., & de Las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231.

Macaulay, L. A. (2012). *Requirements engineering*. Springer Science & Business Media. https://books.google.com/books?hl=en&lr=&id=RojbBwAAQBAJ&oi=fnd&pg=PR7&dq=requirements+engineering&ots=CitXBMQZJ1&sig=aWIlBrT_kdiEqRFLrFPPiKZV7aE

Mahbub, T., Dghaym, D., Shankarnarayanan, A., Syed, T., Shapsough, S., & Zualkernan, I. (2024). Can GPT-4 aid in detecting ambiguities, inconsistencies, and incompleteness in requirements analysis? A comprehensive case study. *IEEE Access*. https://ieeexplore.ieee.org/abstract/document/10684184/

Malik, G., Cevik, M., Parikh, D., & Basar, A. (2024). *Supervised Semantic Similarity-based Conflict Detection Algorithm: S3CDA* (No. arXiv:2206.13690). arXiv. https://doi.org/10.48550/arXiv.2206.13690

Massoudi, S., & Fuge, M. (2025). Agentic large language models for conceptual systems engineering and design. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, *89237*, V03BT03A045. https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-abstract/IDETC-CIE2025/89237/1226016

McDermott, T., DeLaurentis, D., Beling, P., Blackburn, M., & Bone, M. (2020). AI4SE and SE4AI: A Research Roadmap. *INSIGHT*, *23*(1), 8–14. https://doi.org/10.1002/inst.12278

Morkos, B., Shankar, P., & Summers, J. D. (2012). Predicting requirement change propagation, using higher order design structure matrices: An industry case study. *Journal of Engineering Design*, *23*(12), 905–926. https://doi.org/10.1080/09544828.2012.662273

Nonso-Anyakwo, K., & Summers, J. (2025a). Impact of team size on requirement quantity. *Proceedings of the Design Society*, *5*, 2053–2061. https://doi.org/10.1017/pds.2025.10219

Nonso-Anyakwo, K., & Summers, J. D. (2025b). Impact of Team Size on Requirements Variety. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, *89244*, V004T06A025. https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-abstract/IDETC-CIE2025/89244/1225985?casa_token=M3EgBDfDQ30AAAAA:rV5C2jWQ3WI3OGxkQ2BW69LeKeeKEsH1GhJ5xxuJ-Q-UTvZHL6-HDOehdDwk2TjoSBQo6D7OSg

Norheim, J. J., Rebentisch, E., Xiao, D., Draeger, L., Kerbrat, A., & Weck, O. L. de. (2024). Challenges in applying large language models to requirements engineering tasks. *Design Science*, *10*, e16. https://doi.org/10.1017/dsj.2024.8

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: A roadmap. *Proceedings of the Conference on The Future of Software Engineering*, 35–46. https://doi.org/10.1145/336512.336523

Ofsa, M., & Topcu, T. G. (2025). *An Empirical Exploration of ChatGPT's Ability to Support Problem Formulation Tasks for Mission Engineering and a Documentation of its Performance Variability* (No. arXiv:2502.03511). arXiv. https://doi.org/10.48550/arXiv.2502.03511

Peña, M., & Valerdi, R. (2015). Characterizing the Impact of Requirements Volatility on Systems Engineering Effort. *Systems Engineering*, *18*(1), 59–70. https://doi.org/10.1111/sys.21288

Pohl, K. (1996). *Requirements engineering: An overview*. RWTH, Fachgruppe Informatik Aachen. http://137.226.34.227/ftp/pub/packages/CREWS/CREWS-96-02.pdf

Rahimi, M., Mirakhorli, M., & Cleland-Huang, J. (2014). Automated extraction and visualization of quality concerns from requirements specifications. *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 253–262. https://doi.org/10.1109/RE.2014.6912267

Reinhartz-Berger, I., & Kemelman, M. (2020). Extracting core requirements for software product lines. *Requirements Engineering*, *25*(1), 47–65.

Robinson, J. A., Liang, V. W., Chambers, J. M., & MacKenzie, C. L. (1998). Computer user verification using login string keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics-Part a: Systems and Humans*, *28*(2), 236–241.

Sainani, A., Anish, P. R., Joshi, V., & Ghaisas, S. (2020). Extracting and classifying requirements from software engineering contracts. *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 147–157. https://ieeexplore.ieee.org/abstract/document/9218214/

Salado, A., & Nilchiani, R. (2014). A Categorization Model of Requirements Based on Max-Neef's Model of Human Needs. *Systems Engineering*, *17*(3), 348–360. https://doi.org/10.1002/sys.21274

Salado, A., & Wach, P. (2019). Constructing true model-based requirements in SysML. *Systems*, *7*(2), 19.

Simon, H. A. (1973). The structure of ill structured problems. *Artificial Intelligence*, *4*(3), Article 3. https://doi.org/10.1016/0004-3702(73)90011-8

Simon, H. A. (1995). Problem forming, problem finding and problem solving in design. *Design & Systems*, 245–257.

Singh, A., & Szajnfarber, Z. (2025). Architecting Human-AI Systems for Effective Collaboration and Oversight: Making Sense of Human/AI-in/on/Over/Under/Along-the-Loop. *Systems Engineering*, *n/a*(n/a), e70024. https://doi.org/10.1002/sys.70024

Sonbol, R., Rebdawi, G., & Ghneim, N. (2022). The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *Ieee Access*, *10*, 62811–62830.

Song, B., Gyory, J. T., Zhang, G., Zurita, N. F. S., Stump, G., Martin, J., Miller, S., Balon, C., Yukish, M., & McComb, C. (2022). Decoding the agility of artificial intelligence-assisted human design teams. *Design Studies*, *79*, 101094.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *27*. https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

Tikayat Ray, A., Cole, B. F., Pinon Fischer, O. J., White, R. T., & Mavris, D. N. (2023).

aerobert-classifier: Classification of aerospace requirements using bert. *Aerospace*, *10*(3), 279.

Topcu, T. G., Husain, M., Ofsa, M., & Wach, P. (2025). Trust at Your Own Peril: A Mixed Methods Exploration of the Ability of Large Language Models to Generate Expert-Like Systems Engineering Artifacts and a Characterization of Failure Modes. *Systems Engineering*, sys.21810. https://doi.org/10.1002/sys.21810

Viros I Martin, A., & Selva, D. (2022). Learning Comes from Experience: The Effects on Human Learning and Performance of a Virtual Assistant for Design Space Exploration. In J. S. Gero (Ed.), *Design Computing and Cognition'20* (pp. 655–665). Springer International Publishing. https://doi.org/10.1007/978-3-030-90625-2_39

Viros i Martin, A., & Selva, D. (2022). Learning Comes from Experience: The Effects on Human Learning and Performance of a Virtual Assistant for Design Space Exploration. In *Design Computing and Cognition'20* (pp. 655–665). Springer.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45. https://doi.org/10.1145/365153.365168

Zhang, G., Raina, A., Cagan, J., & McComb, C. (2021). A cautionary tale about the impact of AI on human design teams. *Design Studies*, *72*, 100990.

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V., & Batista-Navarro, R. T. (2022). Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Computing Surveys*, *54*(3), 1–41. https://doi.org/10.1145/3444689

Zhu, Y., Zhang, Y., Peng, X., Xue, C., Chen, B., & Cao, Y. (2024). SSMBERT: A Space Science Mission Requirement Classification Method Based on BERT. *Aerospace*, *11*(12), 1031.