

Evaluating the Capabilities of LLMs in Traceability Maintenance for Automotive System and Software Requirements

Vibhashree Hippargi^{1,2}, Erik Kamsties¹ and Jürgen Naumann²

¹Dortmund University of Applied Sciences and Arts, Germany

²Area 21 Software GmbH, Dortmund, Germany

Abstract

Various researchers have explored the potential of Large Language Models (LLMs) for several software engineering tasks, including design solution generation, coding, and test case creation. This paper presents five empirical studies performed on *OpenAI's ChatGPT-4o* to analyze its performance to support different requirements engineering tasks related to requirements *traceability*. Using a dataset from an ongoing automotive project and industry experts' assessments as ground truth, we evaluate ChatGPT-4o's ability to assess trace link *quality* between system requirements, software requirements and test cases, to *predict* the trace links and also to analyze the quality of the *requirements* itself. We also tested ChatGPT-4o with an existing project ticket. Our findings in these studies indicate that ChatGPT-4o demonstrated strong performance, as evidenced by the metrics. These results suggest that ChatGPT-4o can be effectively integrated into daily industry practices as a *support* tool. The dataset is available on GitHub [3].

Keywords

traceability, link quality, automotive, large language model, trace link prediction

1. Motivation

Requirements traceability is a crucial aspect of automotive software engineering, ensuring that all requirements maintain clear and meaningful connections throughout the software lifecycle. Within the framework of Automotive SPICE (ASPICE 4.0) standard Base Practices (BP4: Ensure consistency and establish bidirectional traceability), not only must requirements be traceable, but the trace links must also exhibit consistency, meaning that each link must be relevant and logically meaningful. Achieving this level of consistency requires significant effort from human analysts to verify the quality of trace links. As project complexity increases, so do the requirements and their subsequent connections, leading to the search for tools to support.

Maintaining accurate trace links between system requirements, software requirements, and test cases is resource-intensive, especially when human analysts must verify each link. Furthermore, inconsistencies in requirements documentation can lead to missed or weak links, negatively affecting quality and compliance. The problem this short paper addresses is whether LLMs can effectively support requirements traceability by assessing the link quality and predicting trace links, potentially reducing the effort needed by human analysts.

2. Research Design and Evaluation

This section discusses the empirical studies performed on ChatGPT-4o along with their results. The studies' design and methodology were inspired by foundational AI-assisted requirements engineering

In: A. Hess, A. Susi, E. C. Groen, M. Ruiz, M. Abbas, F. B. Aydemir, M. Daneva, R. Guizzardi, J. Gulden, A. Herrmann, J. Horkoff, S. Kopczyńska, P. Mennig, M. Oriol Hilari, E. Paja, A. Perini, A. Rachmann, K. Schneider, L. Semini, P. Spoletini, A. Vogelsang. *Joint Proceedings of REFSQ-2025 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2025. Barcelona, Spain, April 7, 2025.*

✉ vibhashree.hippargi001@stud.fh-dortmund.de (V. Hippargi); erik.kamsties@fh-dortmund.de (E. Kamsties); juergen.naumann@area21-software.com (J. Naumann)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

research [7], [2]. The studies are supported by and conducted in Area 21 Software GmbH, a software member of Elmos Semiconductors. 21 system and 24 software requirements (low complexity, no IP) were selected from an ongoing project on firmware for an Ultrasonic Park Assist IC. Parameter values were obfuscated, and the dataset was reviewed and approved by experts for integrity. We used Elmos ChatGPT-4o Workspace equipped with ChatGPT-4o. To simulate real-world challenges, minor inconsistencies and ambiguities were introduced in the requirements to test the robustness.

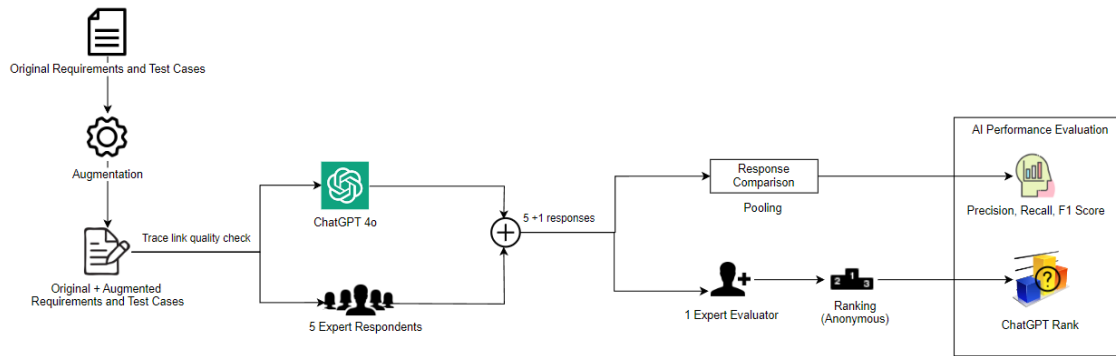


Figure 1: Approach for Study 1

Study 1: Trace Link Quality check between System and Software Requirements. For this study, ChatGPT-4o was prompted to assess and comment on the quality of the trace link with regards to two aspects, *meaningfulness* (logically relevant) and *completeness* (all aspects are reflected). This is based on how well the software requirement details its linked system requirement. To account for variability in ChatGPT’s responses across different sessions, the task was performed in five separate sessions. The responses across these sessions were largely consistent. However, to ensure reliability, findings that appeared in multiple sessions were identified as ChatGPT’s final response.

The evaluation was done using two independent approaches: Pooling and Ranking. In the pooling-based evaluation, five experts at Area 21 with an average industry experience of 10 years (who will be referred to as ‘expert respondents’ going forward) were requested to perform the same task as ChatGPT-4o. Their responses were pooled into a consolidated ground truth, against which ChatGPT-4o’s assessment was compared to evaluate its alignment with human judgment and reliability as a support tool. Its performance was then evaluated for Precision, Recall, and F1 score using TP (ChatGPT-4o matches respondents’ findings), FP (ChatGPT-4o flagged insignificant issues that respondents did not), FN (ChatGPT-4o missed valid issues that respondents identified), and TN (ChatGPT-4o found valid issues that respondents overlooked, extending TN’s traditional definition to reflect AI’s ability to uncover additional insights).

In the ranking-based evaluation, an expert evaluator, i.e., the Head of Embedded Software Development at Area 21, was presented with six anonymous responses (five from expert respondents and one from ChatGPT-4o) and requested to rank them based on response quality and validity, with Rank 1 being the highest. Below is an example of linked requirements given, which ChatGPT-4o analyzed to assess the trace link quality, evaluating the meaningfulness and completeness of SWRS004 in relation to its linked SYSRS004.

SYSRS004: IF performing Erase Backup AND IF Addressed with the Unicast Command THEN the Sensor IC shall respond with CRM_RSP 0x0: Report Acknowledgement.
SWRS004: IF performing Erase Backup AND IF Addressed with the Unicast Command THEN the Software shall prepare to respond with CRM_RSP 0x0: Report Acknowledgment.

The issue with this linkage is that SWRS 004 states, "the software shall *prepare* to respond", while

SYRSR 004 explicitly mandates sending the response. This ambiguity was flagged during the completeness check, as it introduces uncertainty about execution. Both expert respondents and ChatGPT-4o identified this issue, which adds one count to the TP.

The pooling based evaluation result is, $P=0.96$, $R=0.71$, $F1\text{-Score}=0.82$. That is, the precision is high, meaning ChatGPT-4o correctly identified most flagged issues. The recall is moderate, as ChatGPT-4o missed some valid issues that *expert respondents* identified leveraging their experience and expertise. Performance can be improved by refining the prompt to include additional evaluation criteria. The F1-Score is moderate, balancing precision and recall. Time Efficiency: *Expert respondents* took 120 minutes, while ChatGPT-4o took 20 minutes, *with requirements fed manually sequentially* (~ 1 min per requirement). Future automation could further enhance efficiency.

The ranking-based evaluation resulted in ChatGPT-4o responses having an *average rank between 1 and 2* indicating that they were marked as valid and useful by the expert evaluator. There were only two outliers that occurred because all responses for those entries addressed similar issues, leading to no significant differentiation (charts omitted for reasons of space).

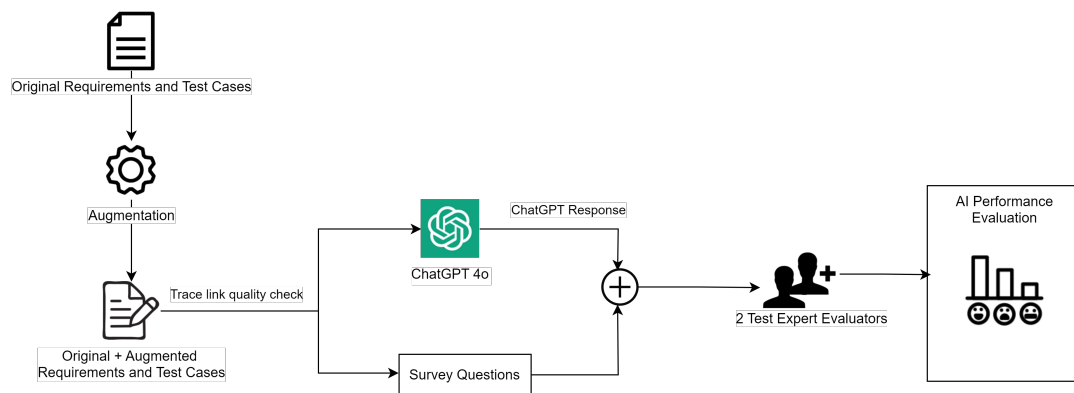


Figure 2: Approach for Study 2

Study 2: Trace Link Quality check between Software Requirements and Test Cases. In this study, ChatGPT-4o was initially provided with possible combinations of parameter values and their interpretations from the software requirements to store in its *memory*. It was then prompted to assess whether or not the test cases entirely covered its linked software requirement and also suggest any missing cases (referring to its *memory*) to attain complete coverage. During its analysis, ChatGPT-4o not only suggested missing test cases but also identified incorrect parameter values being checked in existing test cases. ChatGPT-4o's analysis was reviewed by two test experts at Area 21, who rated its validity and usefulness based on the survey questions: Q1: Is the Meaningfulness and Completeness Analysis Valid? Q2: Do the suggested missing test cases improve coverage? Q3: Are there any other critical aspects or test cases missed? Q4: Please comment on the performance of GenAI. The study was designed as a quick survey to account for the limited availability of the test experts. This approach helped minimize time demands while still gathering their insights into whether ChatGPT-4o could be recommended for similar traceability tasks.

The expert responses were analyzed using the priority order: "Partial > Yes > No." For Q1 and Q2, the ChatGPT-4o achieved a combined "Yes" or "Partial" rating of 100%, with no "No" ratings recorded. This highlights its consistent ability to address the criteria effectively. Furthermore, for Q3, 100% of the response was "No", meaning, the evaluators confirmed that ChatGPT-4o *did not* omit any critical aspects in its test cases, further validating its thoroughness.

The "Partial" ratings for Q1 and Q2 were attributed primarily (70%) to deficiencies in the quality of the software requirements rather than the GenAI's output, indicating that software requirement's lack of detail also contributes to the ambiguity in writing test cases. The remaining 30% of the issues were related

to missing preconditions and negative test scenarios or redundant tests for some of the SWRS - Test cases pair. However, this can be tackled by refining the prompts. Finally, the evaluation of ChatGPT-4o's responses for SWRS test cases showed strong performance. Here is the comment from one of the test experts (also agreed by other) regarding the overall impression of GenAI: *"AI is already highly advanced for evaluating test cases and even requirements. However, there is still room for improvement. It is on a promising path toward being fully utilized for such purposes. In the future, GenAI might take over such tasks entirely, leaving us as reviewers to ensure that it has correctly understood and implemented our expectations."*

Study 3: Trace Link Prediction between System and Software requirements. This study evaluates ChatGPT-4o's performance in predicting trace links between system requirements (SYSRS) and software requirements (SWRS) in the so-called Standard Firmware project. Existing links, established by engineers at Area 21, serve as the ground truth. The process involves providing a prompt and presenting SYSRS and SWRS in randomized order.

The analysis is conducted seven times, each using the same SYSRS list but a different SWRS list. In each iteration, some SWRS are randomly removed (making it different from the previous list), leaving their originally linked SYSRS *unlinked*. ChatGPT-4o's predicted links are then compared to the ground truth to evaluate precision, recall, and F1 score using TP (correctly linked), FP (falsely linked), TN (correctly unlinked), FN (falsely unlinked). The final performance is averaged across all runs as P=1, R=0.97, F1-Score=0.98. Both precision and recall are high, with a slight drop in recall due to a FN. ChatGPT-4o correctly flagged an incomplete lower-level requirement but was marked false negative as the ground truth retained the link.

Study 4: Anomaly detection to identify gaps or ambiguities in requirements. In this study, ChatGPT-4o was instructed to flag any linguistic or content-based ambiguities that may exist within the requirements. Linguistic ambiguities include lexical, syntactic, semantic, syntax, and pragmatic ambiguity [4], [6]. Content based ambiguities include Noise (irrelevant information), Silence (missing important details), Over-specification (including unnecessary solution details), Contradiction (conflicting definitions), Ambiguity (multiple interpretations of a requirement), Forward reference (referring to undefined concepts), and Wishful thinking (unrealistic or unverifiable requirements) [5]. The outcome of this experiment aims to determine whether GenAI can support quality monitoring and improvement by identifying potential ambiguities in requirement documentation. Below is the requirement and ChatGPT-4o's response in the outlined box.

SYSRS 016: AFTER Reception of the Valid CRM_FCC_CMD, THEN the Sensor IC shall set the PDCM_RSP Parameter KAC = 0x0.

For the system requirement **SYSRS 016** shown above, the term *valid* leaves room for interpretation. Such terms like *right*, *valid*, *graciously*, *fast*, *slow* cause pragmatic ambiguity and hence should be avoided in the requirements, especially without a proper definition of what constitutes them. ChatGPT-4o correctly identified this: "The context around what makes CRM_FCC_CMD **valid** could lead to misunderstandings. Is the validity based on the correct syntax, values, or another condition? This needs to be explicitly defined to avoid confusion". For the other requirements, it also identified the presence of noise and contradiction.

Study 5: Inconsistencies between system and software requirements - Real Project Ticket. In this study, ChatGPT-4o was tested on an existing project ticket at Area 21 to determine its ability to identify gaps in system-software requirement alignment.

Description: The requirement SYSRS 021 is not sufficiently detailed by the related downstream requirements SWRS 023 and SWRS 024.

SYSRS 021: The firmware shall allow configurability to select the retry for open and short faults independently.

SWRS 023: IF the software detects an open or short error AND retry is configured as enabled in the DEFAULT_OPEN_SHORT_RETRY value in the COMPILE TIME CONFIGURATION data, the affected channel shall be re-enabled for one TIMESLICE with the following setup at least every 200 milliseconds. **SWRS 024:** The COMPILE TIME CONFIGURATION data shall contain values for DEFAULT_OPEN_SHORT_RETRY.

The missing piece is "independent configuration": SWRS has only one parameter "requiring open and short retries to be configured simultaneously." If an application needs separate timing parameters, a single parameter DEFAULT_OPEN_SHORT_RETRY is insufficient.

It was tested with ChatGPT-4o, to see if it was able to spot the issue. **Prompt:** "I will provide you a system and its linked software requirement. Can you tell me if the latter sufficiently details the former? if not, please explain the issue and suggest revisions." **Response:** "SYSRS 021 emphasizes independent configurability for open and short faults. However, SWRS 023 doesn't clarify how open and short fault retries are configured separately. The statement -if retry is enabled in DEFAULT_OPEN_SHORT_RETRY - implies a single retry configuration for both fault types, *lacking the necessary independent configurability*. There is no mention of whether the retry for open and short faults can be configured separately, which is essential according to SYSRS 021. Suggested revision-Parameters- DEFAULT_OPEN_RETRY and DEFAULT_SHORT_RETRY for open and short fault retries, respectively." The response shows that ChatGPT-4o successfully identified the misalignment, demonstrating its ability to detect inconsistencies between system and software requirements and suggest revisions.

3. Threats to Validity

This section outlines the potential threats to the validity of these empirical studies. *Complexity and Quantity of Requirements:* The dataset was small and consisted of low-complexity requirements, as more complex ones were restricted due to IP and confidentiality constraints. *GenAI Tool Used:* The study utilized ChatGPT-4o, but outcomes may vary with other tools like Gemini or even the newer versions of ChatGPT. *Prompt Engineering:* The prompts were designed with specific criteria checks for the task. However, further refinement to include additional criteria could result in more diverse and comprehensive outputs. *Human Responses:* The quality of responses from expert respondents was influenced by their understanding, motivation, and approach to the evaluation task.

4. Conclusion and Future Work

The results indicate that ChatGPT-4o demonstrates strong performance during the mentioned use cases (refer to P, R, F1-score), positioning it as a valuable tool for preliminary analysis. As highlighted in the GPT-4 Technical Documentation [1] and by other researchers, human verification is crucial for accuracy and reliability. ChatGPT-4o shows promise as an *efficient assistant* for a draft. Readers can refer to GitHub [3] for a detailed view of the prompts and responses, as well as to replicate the case studies. Future work can explore ChatGPT-4o for broader Requirements Engineering tasks, including custom GPT models tailored for collaboration and efficiency. OpenAI's API could enable batch processing, allowing large-scale requirement analysis overnight for review the next day, significantly improving the time required for the evaluation.

Acknowledgments

Sincere appreciation and gratitude are extended to Area 21 Software GmbH for supporting the empirical studies and the participants for providing valuable insights. Furthermore, we acknowledge Jianwei Shi for his valuable feedback about the study on trace link prediction.

References

- [1] *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: <https://arxiv.org/abs/2303.08774>.
- [2] Jin L. C. Guo et al. “Natural Language Processing for Requirements Traceability”. In: To appear in *Handbook on Natural Language Processing for Requirements Engineering*, Springer, 2025. 2024. arXiv: 2405.10845 [cs.SE]. URL: <https://arxiv.org/abs/2405.10845>.
- [3] Vibhashree Hippargi. *Case Study on ChatGPT for RE*. 2025. URL: <https://github.com/VibhaHippargi/CaseStudyonChatGPTforRE>.
- [4] Erik Kamsties. “Understanding ambiguity in requirements engineering”. In: *Engineering and Managing Software Requirements* (2005). Publisher: Springer, pp. 245–266.
- [5] B. Meyer. “On Formalism in Specifications”. In: *IEEE Software* (1985). DOI: 10.1109/MS.1985.229776.
- [6] Fatin Muhamad et al. “Fault-Prone Software Requirements Specification Detection Using Ensemble Learning for Edge/Cloud Applications”. In: *Applied Sciences* 13.14 (2023), p. 8368. DOI: 10.3390/app13148368.
- [7] Krishna Ronanki, Christian Berger, and Jennifer Horkoff. “Investigating ChatGPT’s Potential to Assist in Requirements Elicitation Processes”. In: *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2023, pp. 354–361. DOI: 10.1109/SEAA60479.2023.00061.