LLM-based Coverage Analysis of Tests and Requirements

Francesco Mariotti, Lorenzo Sarti, Stefano Bacherini, Iacopo Trotta*
Alstom Ferroviaria S.p.A.
*Corresponding author - e-mail: iacopo.trotta@alstomgroup.com

Abstract

The correct management of traceability and coverage between tests and requirements can stem the late discovery of issues, which could have significant impact on project delivery. LLM capabilities can be used to perform a semantic comparison between the text of requirements and tests, to classify if a test is covering or not a specific requirement. This can be used to detect mismatches at the early stage. In this work, an LLM-based approach for the analysis of coverage between tests and requirements is shown and evaluated through classification metrics.

Keywords: railway signaling; RAMS.

1 Introduction

Railway control and protection systems are safety-critical, so they are regulated by strict CENELEC standards such as EN 50716, EN 50126, and EN 50129. These standards emphasize rigorous verification and validation (V&V), systematic requirements engineering, and the construction of a defensible safety case. A key expectation is full traceability of hazards, requirements, design and implementation and verification evidence, along with adequate test coverage for the intended Safety Integrity Level (SIL).
Requirements-to-test traceability is essential because of the following benefits:

1. It strengthens the safety case by proving all safety requirements, especially those derived from hazard analysis, are verified by specific tests.
2. It supports impact analysis and configuration management, which is crucial for long-lifecycle railway assets and multi-supplier environments.
3. It enables coverage assessment, ensuring not only code coverage but also coverage of requirements, interfaces, and operational scenarios aligned with CENELEC expectations.

Proper management of traceability of tests and requirements can avoid the late finding of problems with significant cost and time reduction [1]. However, such management is often done in peer-review through a Requirement Validation Table (RVT), which can lead to systematic errors.
Large Language Models (LLMs) offer powerful capabilities for semantic comparison, contextual interpretation, and similarity analysis of natural-language artifacts [2][3]. In this work, we leverage these capabilities to automatically identify potential correspondences between test descriptions and requirements. By analysing the meaning and intent of the texts LLMs can surface latent or non-obvious relationships that traditional traceability tools may overlook [4]. This automated semantic matching provides analysts with timely and actionable feedback on the

completeness and correctness of traceability links, supporting both the validation of existing mappings and the discovery of missing or inconsistent ones.
To assess the effectiveness of the proposed approach, we evaluate its performance using classification metrics, measuring the extent to which the model-generated matches align with expert-validated ground truth. This quantitative evaluation allows us to determine the reliability of LLM-based semantic traceability and to identify areas where additional refinements may be necessary.

2 Methodology

The proposed idea has been implemented through a Python pipeline and can be schematized as follows:

1. RVT is provided as input by the user. The information present in the document consists of the test IDs and their descriptions, the requirements IDs, their descriptions and the associated tests IDs.
2. The prompts to be provided to the LLM are created. One single prompt is built for each requirement/test couple. By providing the description of both test and requirement, we ask in the prompt to understand if the given test is totally, partially or not covering the given requirement.
3. The prompts are launched through API calls to the LLM.
4. The results are gathered and shown to the user.

2.1 Evaluation metrics

To evaluate the approach, we can consider as the ground truth the RVTs already validated in past projects. As the approach addresses a classification problem, i.e., if a given test covers a given requirement, the classic confusion matrix is considered [5].

Table 1: Confusion Matrix

		Actual Values	
		Positive	**Negative**
Predicted Values	**Positive**	**True Positive (TP):** test indicated by the LLM as covering the requirement and present in the RVT	**False Positive (FP):** test indicated by the LLM as covering the requirement but not present in the RVT
	Negative	**False Negative (FN):** test not indicated by the LLM as covering the requirement but present in the RVT	**True Negative (TN):** test not indicated by the LLM as covering the requirement and not present in the RVT

The composition of the confusion matrix is shown in Table 1. To evaluate the approach, we can consider as the ground truth the RVTs already validated in past projects. As the approach

addresses a classification problem, i.e., if a given test covers a given requirement, the classic confusion matrix is considered [5]. The composition of the confusion matrix is shown in Table 1. Classic evaluation metrics such as Precision and Recall are not relevant in this case, due to the high occurrence of TNs. In fact, given a test it is natural that it will cover only a subset of requirements. Instead, the most relevant metric is the percentage of FNs: an FN means a missing test/requirement traceability, possibly leading to late discovery of issues.

2.2 Evaluation campaigns

The following evaluation campaigns were conducted to assess the effectiveness of the approach:

- **EC1** - Is the LLM analysis consistent if repeated? Launch an experiment with the same setting multiple times (trade-off between statistical relevance and cost). Analyze the metrics outcome and check the standard deviation.
- **EC2** - How does the prompt affect the results? Using a fixed project, change the prompt to possibly improve the outcome.
- **EC3** - What are the performances of the pipeline on different projects? Run the pipeline on different projects to evaluate how the system performs with different requirements and tests.

3 Results and discussion

EC1 was conducted on *Project1*, having 66 requirements and 34 tests, hence resulting in 2244 prompts. The results are shown in Table 2. First, it is possible to notice that the percentage of FNs is low with respect to the population. Moreover, the standard deviation is low, meaning that the results obtained from the LLM are stable with respect to repetitions.
For EC2 we still used *Project1*, using three different prompts with an increasing degree of details: *Prompt1* containing just the instruction and the description of requirement and test; *Prompt2* providing also some definitions, e.g., test coverage; *Prompt3* containing also some examples of test/requirement coverage. The results are shown in
Table 3. Differences between Prompt1 and Prompt2 are negligible, while for Prompt3 it is possible to notice a slight decrease in the percentage of FNs, but with a higher execution time. This means that the details contained in the prompt affect the results, but a time and cost increase shall be considered.

Table 2: Results for EC1

Experiment ID	TP	TN	FP	FN	Accuracy	FN %
EC1_01	65	1974	92	113	0,909	5,036
EC1_02	68	1971	95	110	0,909	4,902
EC1_03	66	1976	90	112	0,91	4,991
EC1_04	66	1982	84	112	0,913	4,991
EC1_05	65	1973	93	113	0,908	5,036
EC1_06	63	1977	89	115	0,909	5,125
EC1_07	63	1972	94	115	0,907	5,125
EC1_08	73	1979	87	105	0,914	4,679
EC1_09	67	1981	85	111	0,913	4,947
EC1_10	67	1967	99	111	0,906	4,947
Standard Deviation	2,722	4,467	4,467	2,722	0,003	0,017

Finally, in EC3 we experimented with different projects coming from main line and urban domains. Results in Table 4 show that the percentage of FNs across the different projects are quite comparable, meaning that the approach can be applied to different sectors.

Table 3: Results for EC2

Experiment ID	Prompt ID	Execution Time	TP	TN	FP	FN	FN %
EC2_01	Prompt1	08m31s	73	1979	87	105	4,679
EC2_02	Prompt1	08m37s	67	1981	85	111	4,947
EC2_03	Prompt1	07m45s	67	1967	99	111	4,947
EC2_04	Prompt2	08m02s	66	1984	82	112	4,991
EC2_05	Prompt2	08m15s	68	1976	90	110	4,902
EC2_06	Prompt2	08m29s	70	1974	92	108	4,813
EC2_07	Prompt3	33m53s	103	1658	408	75	3,342
EC2_08	Prompt3	38m35s	108	1666	400	70	3,119
EC2_09	Prompt3	35m25s	109	1677	389	69	3,075

Table 4: Results for EC3

Experiment ID	TP	TN	FP	FN	Accuracy	FN%
EC3_PJ1	66	1984	82	112	0,914	4,991
EC3_PJ2	25	248	63	16	0,776	4,545
EC3_PJ3	73	873	219	55	0,775	4,508
EC3_PJ4	32	775	8	10	0,978	1,212
EC3_PJ5	23	781	16	16	0,962	1,914

4 Conclusions

In this work we presented an LLM-based approach for the identification of tests/requirements coverage, which can be used as a support tool for the analyst to avoid the late detection of issues. The evaluation campaigns showed that the approach is consistent and applicable to different domains.

References

[1] M. Ruiz, J.Y. Hu, F. Dalpiaz (2023). Why don't we trace? A study on the barriers to software traceability in practice. *Requirements Eng 28*.

[2] Y. Zhu, J.R. Antony Moniz, S. Bhargava, J. Lu, D. Piraviperumal, S. Li, Y. Zhang, H. Yu, and B. Tseng (2024). Can Large Language Models Understand Context? In *Findings of the Association for Computational Linguistics: EACL 2024*, St. Julian's, Malta. Association for Computational Linguistics.

[3] E.E. Erkan, M.A. Kömürcü, T. Çelikten, A.E. Ergün, A. Onan (2025). Understanding Large Language Model Performance in Question Answering: A Comparative Analysis of Semantic and Lexical Metrics. In: Kahraman, C., *et al*. Intelligent and Fuzzy Systems. INFUS 2025. Lecture Notes in Networks and Systems, vol 1529. Springer, Cham.

[4] S.J. Ali, V. Naganathan, D. Bork (2025). Establishing Traceability Between Natural Language Requirements and Software Artifacts by Combining RAG and LLMs. In: Maass, W., Han, H., Yasar, H., Multari, N. (eds) Conceptual Modeling. ER 2024. Lecture Notes in Computer Science, vol 15238. Springer, Cham.

[5] S.V. Stehman (1997), Selecting and interpreting measures of thematic classification accuracy, Remote Sensing of Environment, Volume 62, Issue 1.