Towards a Large Language Model Based Approach for the Management of Requirements Technical Debt.

Fernando Guilhermino[1], Maria Lencastre[2]

[1]Universidade de Pernambuco (UPE)
Caixa Postal 50720 – 001 – Recife – PE – Brazil

[2]Universidade de Pernambuco (UPE)

{jfgs,mlpm}@ecomp.poli.br

Abstract. *This study addresses the challenge of mitigating Requirements Technical Debt (RTD) by proposing an RTD management approach supported by Large Language Models (LLMs). The approach structures automated mechanisms for the identification, characterization, impact analysis, and evolutionary monitoring of debt throughout the software lifecycle. It articulates linguistic, technical, and decision-making aspects, allowing RTD to be treated as a continuous phenomenon that influences product quality, risk, and sustainability. The approach is expected to provide systematic support for RTD detection, prevention, and mitigation activities, based on the analysis of its potential impact and evolution over time.*

1. Introduction

Technical Debt (TD) refers to short-term decisions made during software development that, while potentially accelerating the initial delivery, result in additional maintenance and evolution costs throughout the system's life cycle [Ramac et al. 2022]. Among the various types of TD, Requirements Technical Debt (RTD) stands out, defined as the gap between the optimal solution to a requirements problem and the solution implemented within a given decision context [Ernst 2012]. RTD is particularly critical because requirements form the foundation for all other activities in the software development process, directly influencing stages such as design, implementation, testing, and maintenance [Barbosa et al. 2022].

[Lenarduzzi and Fucci 2019] classify RTD into three main types: Type 0, related to incomplete stakeholder needs; Type 1, associated with requirement smells that violate quality criteria (ISO/IEC/IEEE 29148); and Type 2, which refers to inconsistencies between specified requirements and system implementation. TD management is crucial for project success and focuses on minimizing its harmful effects through prevention and control. According to [Li, Avgeriou, and Liang 2015] the key activities include: detecting technical debt; quantifying costs and benefits of the debt; ranking debts based on criteria to decide which to fix first; preventing new TD formation; monitoring changes in costs and benefits of unresolved debt over time; resolving or mitigating TD; uniformly recording technical debts; and communicating debt information with stakeholders for proper management.

In this context, requirements management plays a fundamental role, particularly in the early stages of software development. Despite its importance, existing tools often

lack a specific focus on requirements management and the different types of RTD [Silva et al., 2024]. Moreover, current studies do not adequately address the evolution of RTD throughout the software lifecycle, especially its integration with machine learning and artificial intelligence-based development approaches. Traditional techniques for requirements management, such as rule-based natural language processing and conventional machine learning, also present limitations, including rigidity, strong dependence on manually crafted rules, and the need for large labeled datasets [Fantechi, Gnesi, and Semini, 2023].

Given their ability to process and relate large volumes of information with contextual awareness, Large Language Models (LLMs) have emerged [Fantechi, Gnesi, and Semini, 2023]. This could be a promising approach to support the management of RTD. This article proposes an LLM based approach to support the mitigation and management of RTD. The approach identifies missing, implicit, or insufficiently specified stakeholder needs by analyzing requirements and highlighting potential gaps. It also supports the detection and mitigation of specification issues, such as ambiguity, inconsistency, and lack of clarity, which violate established quality standards (ISO/IEC/IEEE 29148). Overall, the approach contributes to core RTD management activities particularly detection, prevention, and mitigation and supports RTD documentation and communication by structuring identified issues in a comprehensible manner. Its development is guided by the Design Science Research (DSR) methodology [Wieringa, 2014], which is well suited for designing artifacts that address practical problems.

2. Related Works

The following sections present several of the tools identified in the literature. There are tools designed to support Requirements Engineering (RE) that align with the prevention phase of RTD management. However, most of them focus on specific aspects of the process, lacking an integrated perspective that considers different RTD types. The following sections present some of the tools identified in the literature.

PASKA is an automated tool for requirements quality analysis that operates on natural language SRS documents. It combines linguistic processing techniques with a Controlled Natural Language (Rimay) to identify requirement smells. By improving attributes such as clarity, atomicity, and correctness, the tool follows a rule-based approach grounded in quality standards and primarily contributes to mitigating Type 1 RTD related to semantic and syntactic deficiencies [Veizaga, Shin, and Briand, 2024].

ReqGo is a semi-automatic tool for requirements management that analyzes natural language requirements in early stages. It applies NLP techniques, such as tokenization and lemmatization, to detect ambiguity and support classification and prioritization. By identifying textual and structural defects, it focuses on requirement smells, contributing to the mitigation of Type 1 RTD related to specification quality issues [Koh and Chua, 2023].

ReqBrain is an AI-based assistant that supports requirements elicitation and specification using fine-tuned language models. It operates generatively on user-provided requirements and can incorporate external knowledge through Retrieval-Augmented Generation (RAG). Its goal is to generate complete and consistent requirements, refining poorly specified ones and identifying missing information. Thus,

it contributes to mitigating both Type 0 RTD (incompleteness) and Type 1 RTD (low specification quality) [Habib et al. 2025].

QuARS is a classic requirements analysis tool based on deterministic rules derived from a quality model. It processes textual requirements documents in natural language through lexical and syntactic analysis, identifying linguistic defects associated with ambiguity, vagueness, and other structural violations. Because it utilizes a strictly rule-based approach, its operation is centered on verifying the textual compliance of the specification, characterizing a strategy for detecting requirement smells. Consequently, QuARS is directed toward the mitigation of Type 1 RTD [Fantechi, Gnesi, and Semini 2023].

NLP4ReF proposes an approach that combines Natural Language Processing (NLP) and machine learning techniques to support knowledge discovery during requirements engineering. The solution operates on initial structured requirements lists (CSV files), analyzing them to identify semantic gaps and suggest requirements not previously considered. By emphasizing the identification of omitted or unanticipated requirements during analysis, the approach explicitly addresses incompleteness of needs, contributing to the mitigation of Type 0 RTD.

The existing tools identified in the literature predominantly address localized aspects of requirements quality or support specific phases of requirements engineering. Rule-based NLP tools, such as PASKA and QuARS, focus on detecting linguistic deficiencies in natural language specifications, mainly targeting requirement smells and mitigating Type 1 RTD at the textual level. ReqGo emphasizes ambiguity handling and prioritization but does not conceptualize technical debt as a measurable or evolving construct. Similarly, ReqBrain and NLP4ReF contribute to mitigating Type 0 and Type 1 RTD by refining requirements and identifying incompleteness; however, they treat requirements as isolated artifacts and do not address RTD from a lifecycle perspective.

In contrast, the approach proposed in this study adopts RTD as the central analytical construct and structures its mechanisms around its identification, characterization, and monitoring. Rather than focusing on isolated textual issues, it conceptualizes RTD as a cumulative and evolving phenomenon that requires continuous management throughout the software lifecycle. By integrating semantic analysis with RTD typology (Types 0 and 1) and decision-support mechanisms, the approach moves beyond quality inspection toward a more systematic and structured management of RTD.

3. Methodology

This study adopts the DSR approach [Wieringa 2014], as it is particularly suitable for investigating practical problems and proposing artifacts aimed at improving an existing situation. DSR enables the integration of scientific knowledge production with the development of technological solutions, being widely used in research focused on Software Engineering. Figure 1 presents the design cycle of this research, detailed next.

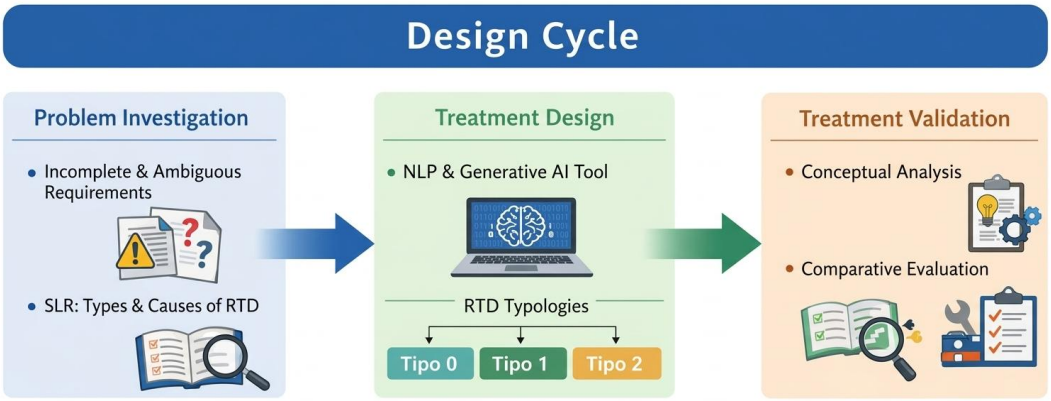

Figure 1. Steps of the Design Cycle in the Proposed Approach

The design cycle of the proposed approach focuses on creating and validating the proposed artifact. The main steps include:

Problem Investigation: this stage aimed to understand the context in which RTD emerges and to identify the phenomena that contribute to its formation. As part of this investigation, an Systematic Literature Review (SLR) was conducted to identify the types of RTD, their recurring causes, and existing approaches for their detection and mitigation [Guilhermino and Lencastre 2026] (The article has been submitted and is awaiting approval). The results provided theoretical and empirical foundation to characterize the problem and justify the need for a solution that explicitly and systematically addresses RTD.

Treatment Design: based on the previous results, the treatment design involved planning an artifact intended to operate within the problematic context to promote improvements. An approach, based on NLP and Generative Artificial Intelligence techniques, grounded in LLMs, is being developed. This design is guided by the RTD typology (Types 0, 1, and 2) and is structured to support the detection of RTD indicators in existing requirements artifacts. The artifact requirements and their main analysis modules have been defined, with an emphasis on identifying completeness gaps and signs of ambiguity in textual requirements. The proposal focuses on the early stages of requirements elicitation and analysis to act preventively in mitigating RTD.

Treatment Validation: Based on the SLR findings, the proposed approach consists of a framework for the structured management of RTD, supported by Natural Language Processing and Generative AI techniques using LLMs. The design is guided by RTD typology (Types 0, 1, and 2) and includes mechanisms to identify indicators in requirements artifacts, characterize their impact, and support monitoring and decision-making throughout the software lifecycle. The evaluation combines quantitative metrics (precision, recall, and F-measure) with qualitative feedback from requirements engineers and domain experts, focusing on utility, clarity, and reliability. The results of each validation cycle are used to iteratively refine the artifact, including its detection mechanisms and interaction features, until its contribution to RTD mitigation is adequately demonstrated.

4. Prototype

The prototype is designed not only to detect ambiguities or requirement smells, but also to: (i) classify RTD indicators according to the established typology (Types 0 and 1); (ii) provide structured interpretation of their potential implications; and (iii) offer decision-support elements to assist requirements engineers in treatment planning. Semantic analysis mechanisms based on NLP and LLMs are integrated with modules that contextualize detected indicators within an RTD perspective and generate explanatory feedback. Rather than functioning solely as a quality inspection tool, the prototype operationalizes the conceptual framework by making RTD manifestations explicit and actionable within requirements engineering activities.

Figure 2 presents the first low-fidelity prototype of the proposed RTD-oriented framework, structured into three panels representing diagnosis, implications analysis, and resolution support. The interface makes explicit the identification and typological classification (Types 0 and 1) of RTD indicators derived from textual requirements analysis, contextualizing manifestations of incompleteness and ambiguity within an RTD perspective. It also provides interpretative cues regarding potential implications and affected areas, along with structured decision-support elements for mitigation planning and prioritization. The prototype is conceptual in nature and was designed to support expert-based evaluation of its coherence, adequacy, and perceived contribution to RTD control in requirements engineering practices.

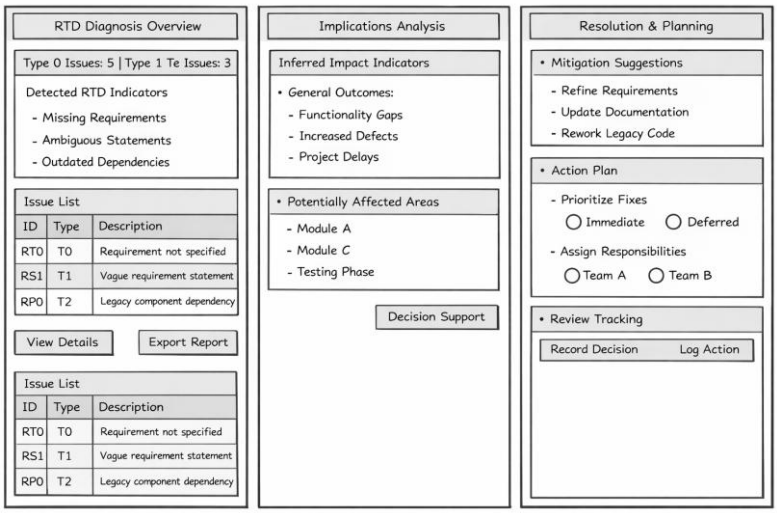

Figure 2. Low fidelity prototype

5. Conclusions

This work presents the initial development of a framework to support the management of RTD in software projects. Grounded in a Design Science Research approach, the proposal conceptualizes RTD as an evolving phenomenon and structures mechanisms for its identification, classification, impact assessment, and monitoring throughout the software lifecycle. Informed by evidence from a Systematic Literature Review, the framework addresses limitations of existing tools that focus primarily on isolated quality issues, positioning RTD as a central element for structured and continuous management.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

References

Barbosa, L., Freire, S., Rios, N., Ramac, R., Taušan, N., Pérez, B., ... & Spínola, R. (2022, August). Organizing the TD management landscape for requirements and requirements documentation debt. In Workshop on Requirements Engineering (WER).

Biazotto, J. P., Feitosa, D., Avgeriou, P., & Nakagawa, E. Y. (2025). Understanding practitioners' reasoning and requirements for efficient tool support in technical debt management. Empirical Software Engineering, 30(5), 134.

Ernst, N. A. (2012, June). On the role of requirements in understanding and managing technical debt. In 2012 Third International Workshop on Managing Technical Debt (MTD) (pp. 61-64). IEEE.

Fantechi, A., Gnesi, S., & Semini, L. (2023). Rule-based NLP vs ChatGPT in ambiguity detection, a preliminary study. In CEUR Workshop Proceedings (Vol. 3378). CEUR WS.

Frattini, J., Fucci, D., Mendez, D., Spínola, R., Mandić, V., Taušan, N., ... & Gonzalez-Huerta, J. (2023). An initial theory to understand and manage requirements engineering debt in practice. Information and Software Technology, 159, 107201.

Guilhermino, F., & Lencastre, M. (2026). Leveraging LLMs to mitigate requirements technical debt: Debt types, tools and limitations – A systematic literature review. In press.

Gupta, A. K., Siddiqui, S. T., Qidwai, K. A., Haider, A. S., Khan, H., & Ahmad, M. O. (2022, December). Software Requirement Ambiguity Avoidance Framework (SRAAF) for Selecting Suitable Requirement Elicitation Techniques for Software Projects. In 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET) (pp. 1-6). IEEE.

Habib, M. K., Graziotin, D., & Wagner, S. (2025). ReqBrain: Task-Specific Instruction Tuning of LLMs for AI-Assisted Requirements Generation. arXiv preprint arXiv:2505.17632.

Koh, S. J., & Chua, F. F. (2023). ReqGo: a semi-automated requirements management tool. International Journal of Technology, 14(4), 713.

Lenarduzzi, V., & Fucci, D. (2019, September). Towards a holistic definition of requirements debt. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-5). IEEE.

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. Journal of systems and software, 101, 193-220.

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., ... & Spinola, R. (2022). Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. Journal of Systems and Software, 184, 111114.

Rusdianto, D. S., Fabroyir, H., & Yuhana, U. L. (2024, August). Innovative approaches to impact analysis of requirement changes using llm in software projects. In 2024

IEEE International Symposium on Consumer Technology (ISCT) (pp. 604-610). IEEE.

da Silva, J. F. G., Lencastre, M., & Castro, J. (2024). Modelagem Conceitual de Dívida Técnica na Engenharia de Requisitos. In 25a Workshop em Engenharia de Requisitos (WER 24).

Veizaga, A., Shin, S. Y., & Briand, L. C. (2024). Automated smell detection and recommendation in natural language requirements. IEEE Transactions on Software Engineering, 50(4), 695-720.

Wieringa, R. (2014). Design science methodology for information systems and software engineering. Springer-Verlag Berlin Heidelberg.