Prompt engineering for facilitating requirement elicitation from multi-format inputs

Mantas Razinskas[1]

[1] *Department of Information Systems, Faculty of Informatics, Kaunas University of Technology, 51368 Kaunas, Lithuania*

Abstract

Requirements Engineering (RE) increasingly relies on semi-structured and multi-format inputs such as textual descriptions, stakeholder notes, wireframes, and technical specifications. This paper proposes a hybrid methodology that combines structured prompt engineering techniques, a conceptual metamodel, and a conceptual agent-based architecture to facilitate requirements generation from heterogeneous sources. The approach operationalizes independent, negotiable, valuable, estimable, small, and testable (INVEST) framework through prompt templates and applies a metamodel to formalize the transformation process, enabling evaluation and refinement of outputs. The conceptual architecture introduces modular agents for extraction, validation, and refinement tasks, facilitating collaboration and continuous improvement. Initial experiments demonstrate that prompt-based elicitation could help in eliciting requirements. The methodology addresses gaps in existing Artificial Intelligence (AI) and model-based RE approaches and contributes a framework for integrating Large Language Models (LLMs) into Agile requirements workflows. Future work includes prototyping, large-scale validation, and domain-specific adaptation.

Keywords

requirements engineering, large language models, model-based development, requirements elicitation

1. Introduction

Requirements Engineering plays an important role in Agile software development, yet it remains a challenging activity due to the informality of inputs, evolving stakeholder needs, and time-constrained iterations [1, 2]. Traditional RE methods often lack the flexibility to accommodate the dynamic nature of Agile projects, leading to inconsistencies, communication gaps, and delayed clarification of requirements [3, 4].

Recent advances in AI, particularly the emergence of Large Language Models, have shown potential in supporting requirement elicitation, classification, and specification through natural language understanding and generation [5, 6, 7]. LLMs can process unstructured information and generate requirement artifacts; however, they frequently lack domain adaptation, traceability mechanisms, and assurance of output quality [8, 9, 10].

Simultaneously, Model-Based Development (MBD) and Engineering (MBE) approaches offer structured methodologies for representing, validating, and evolving requirements using models such as Unified Modeling Language (UML), domain-specific diagrams, or formal specifications

BIR-WS 2025: BIR 2025 Workshops and Doctoral Consortium, 24th International Conference on Perspectives in Business Informatics Research (BIR 2025), September 17-19, 2025, Riga, Latvia.

mantas.razinskas@ktu.edu (M. Razinskas)

0009-0003-5172-3428 (M. Razinskas)

[11, 12]. These methods support traceability and consistency but are often perceived as resource-intensive and difficult to integrate into lightweight Agile workflows.

Despite individual advances, there is limited work on combining the flexibility of LLMs with the formal rigor of model-based techniques to support RE in Agile contexts [12, 13]. This paper addresses this gap by introducing a hybrid methodology that employs structured prompt engineering patterns, a conceptual metamodel, and conceptual agent-based architecture. The goal is to support Agile teams in transforming heterogeneous and often informal inputs—such as stakeholder notes, wireframes, diagrams, and early technical drafts—into initial requirement formulations that are clearer, testable, and traceable. These outputs are not meant to replace stakeholder-derived requirements but serve as structured starting points to accelerate elicitation and refinement cycles. In Agile contexts, where time and iteration pressure often lead to fragmented requirement documentation, such LLM-assisted transformation could improve clarity and reduce rework, while ensuring that final validation remains in the hands of stakeholders. Importantly, any generated requirement is treated as a proposal—its confirmation and adoption would always go through regular stakeholder review and agreement processes, preserving the essential human-in-the-loop principle.

The paper is structured as following: Section 2 defines the problem context and presents the research questions; Section 3 reviews related work on AI and model-based approaches in Requirements Engineering; Section 4 outlines the research methodology and development process of the proposed solution; Section 5 introduces the core components of the methodology including prompt templates, metamodel, and agent-based architecture; Section 6 presents preliminary results; Section 7 describes the evaluation plan; Section 8 discusses future work directions, and Section 9 concludes the paper.

2. Problem statement and research questions

Organizations face challenges in managing requirements elicitation, analysis, and specification processes in Agile projects. While Artificial Intelligence tools, such as Large Language Models, demonstrate potential in automating requirements-related tasks, they often struggle to adapt to domain-specific requirements and to support effective stakeholder collaboration [5, 8, 6]. Similarly, Model-Based Development practices provide structured methodologies for requirements representation and modeling, such as UML diagrams, but can be resource-intensive and challenging to apply in Agile workflows [11, 12]. This creates a need for an integrated approach that leverages AI techniques, in conjunction with MBD practices to streamline requirements elicitation, analysis and specification, enhance communication and visualization, and generate consistent models that align with Agile methodologies [6, 8].

Accordingly, this research aims to address the following questions:

1. How can model-based engineering (MBE) principles be integrated into Agile requirements engineering workflows supported by large language models?
2. How can prompt-based interactions with large language models be used to elicit requirements from heterogeneous inputs such as textual descriptions, visual diagrams, and domain documents?
3. How can the quality, consistency, and relevance of AI-generated requirements be evaluated and improved in Agile workflows using both automated and expert-driven criteria?

4. How can an agent-based tooling architecture coordinate LLM components to support continuous elicitation, validation, and refinement of requirements in collaborative settings?

3. Related work

Recent advancements in Requirements Engineering research have focused on leveraging AI technologies and integrating model-based practices to address the limitations of traditional methods.

Umar and Lano [8] provide a systematic review of automation in RE, identifying that analysis and elicitation are the most commonly automated phases, with tools often relying on Natural Language Processing (NLP) techniques to generate UML models. Arora et al. [6] propose using LLMs to enhance RE through requirement extraction and specification, while Ronanki et al. [9] explore prompt engineering patterns for RE tasks like classification and traceability. Cheng et al. [5] and Norheim et al. [7] emphasize the potential of GenAI and LLMs in various RE phases, highlighting both their promise and challenges such as data limitations and model interpretability.

Further studies such as Belzner et al. [14], Vogelsang and Fischbach [10], and Sami et al. [15] explore broader applications of LLMs across the software lifecycle, including requirement generation, validation, and prioritization. Mehraj et al. [5] present a tertiary review of AI4RE, while Spoletini and Ferrari [13] propose integrating formal RE techniques with LLMs to improve reliability.

Regarding model-based integration, Huss et al. [12] introduce the Scrum Model-Based System Architecture Process (sMBSAP), demonstrating how MBSE can be embedded into Agile workflows. Agile MERODE [11] offers another integration of Agile and MDSE, emphasizing user stories and domain modeling to ensure traceability and iterative development support.

Despite these advancements, there is limited work on combining LLM elicitation with structured modeling practices to support Requirement Engineering tasks in Agile environments—a gap this research aims to address.

4. Research methodology

The research is carried out using the Design Science Research (DSR) method [16, 17]. The method consists of developing a solution concept (artifact) for an identified problem and evaluating it in a relevant context. The research is carried out in the following steps, combining already implemented activities and planned future work:

Research problem definition. An analysis has been conducted on Agile requirements engineering challenges, particularly related to the lack of structure, traceability, and integration of AI-generated requirements. The problem is positioned in the context of combining LLM-based elicitation with modeling.

Potential of research. The potential of applying AI tools (LLMs, prompt engineering) and model-based approaches in Agile requirements engineering has been analyzed to determine gaps and improvement opportunities.

Problem domain analysis. A comparative analysis of literature and tool support in the domain of AI-based RE and MBD integration has been initiated. Relevant RE tasks are being mapped against possible automation opportunities using AI and prompt techniques.

Solution conception. A hybrid methodology is being developed that combines structured prompt patterns, a conceptual metamodel, and a conceptual modular agent-based architecture. The solution is intended to enable the transformation from diverse inputs into user stories.

Implementation and evaluation. Initial experiments are being conducted to generate user stories from various input formats (text, images, diagrams). Prompt patterns are being applied and iteratively refined. Output evaluation using the INVEST criteria and expert review is planned and partially in progress.

Evaluation of the application of the solution. The feasibility of applying the methodology in Agile RE scenarios will be assessed through structured real-world use case scenarios. Planned use cases include extraction and refinement of requirements from diverse input types such as stakeholder notes, diagrams, and domain documents.

Theoretical relevance analysis. The developed methodology will be compared with existing RE approaches in the literature. The proposed approach is positioned as a contribution toward combining AI-based elicitation with model-based structure in Agile contexts.

The methodology follows an iterative design-evaluate-refine loop, aligning with the DSR process. Although presented sequentially for readability, the development and evaluation of artefacts are inherently cyclical and informed by continuous feedback.

5. Proposed contribution

The proposed solution is a hybrid methodology aimed at supporting requirements engineers during early Agile requirements engineering activities, primarily elicitation and clarification. The methodology integrates structured prompt engineering techniques, large language models, and conceptual metamodel. Its main components are as follows:

Prompt engineering patterns

A set of structured prompt templates is designed to guide LLMs in extracting requirements from heterogeneous inputs, including stakeholder descriptions, diagrams, and annotated images. These prompts follow repeatable patterns and are adapted to support Agile-specific requirements such as INVEST-compliant user stories. An example of a structured prompt template can be accessed via the following GitHub repository: https://gist.github.com/ntelio/c4c9737576844020fd37e94f9a1037e4.

The presented prompt template operationalizes Agile requirements engineering principles by guiding large language models in the extraction and structuring of user stories from heterogeneous input formats. It enforces a consistent user story structure aligned with the INVEST criteria and introduces mandatory quality controls such as clarity, testability, and business-driven prioritization. The template includes processing strategies for various input types (text, wireframes, technical specifications, annotated images), ensuring coverage of both functional and system-level requirements. Furthermore, it enhances traceability by requiring each user story to include its source and a justification of priority.

Conceptual metamodel

A conceptual metamodel (as illustrated in Figure 1) has been developed to represent the structure and relationships between the inputs, transformation processes, generated artifacts, and evaluation mechanisms involved in the AI-assisted requirements engineering workflow.

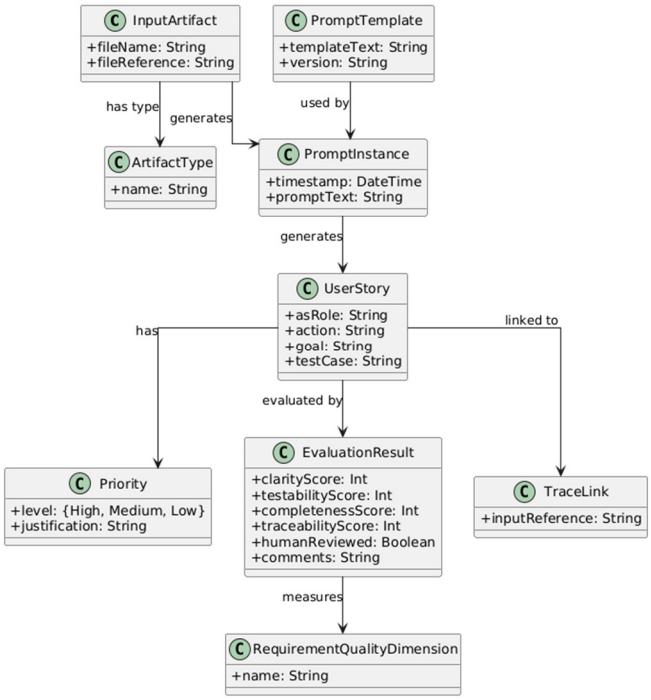

Figure 1: Conceptual metamodel for AI-assisted requirements engineering workflow.

The metamodel includes the following core entities:

- InputArtifact – represents the original requirement sources (textual documents, images, diagrams, wireframes, or specifications). Each artifact is referenced by its file metadata and categorized by type.
- PromptTemplate – defines reusable structured prompt patterns, which guide the large language model in transforming raw input into requirements.
- PromptInstance – represents a single invocation of the LLM using a specific prompt and input artifact. It captures the actual prompt text and timestamp of execution.
- UserStory – the main output artifact generated by the LLM. Each user story follows a structured format including role, action, goal, testing scenario, priority level, justification, and a reference to its source.
- EvaluationResult – captures the quality assessment of a generated user story based on dimensions such as clarity, testability, completeness, and traceability. It records human review status and feedback.
- TraceLink – maintains traceability between generated user stories and their originating input artifacts to ensure requirement coverage and auditability.

Agent-based architecture (conceptual)

A conceptual architecture is outlined for future implementation, to support the orchestration of AI-assisted requirements engineering workflows as shown in Figure 2. At the center of this architecture is a CoordinatorAgent, which manages the overall process and serves as the communication point with the human Analyst. The CoordinatorAgent assigns responsibilities

to modular, task-specific agents, each handling a distinct phase of the requirements transformation process.

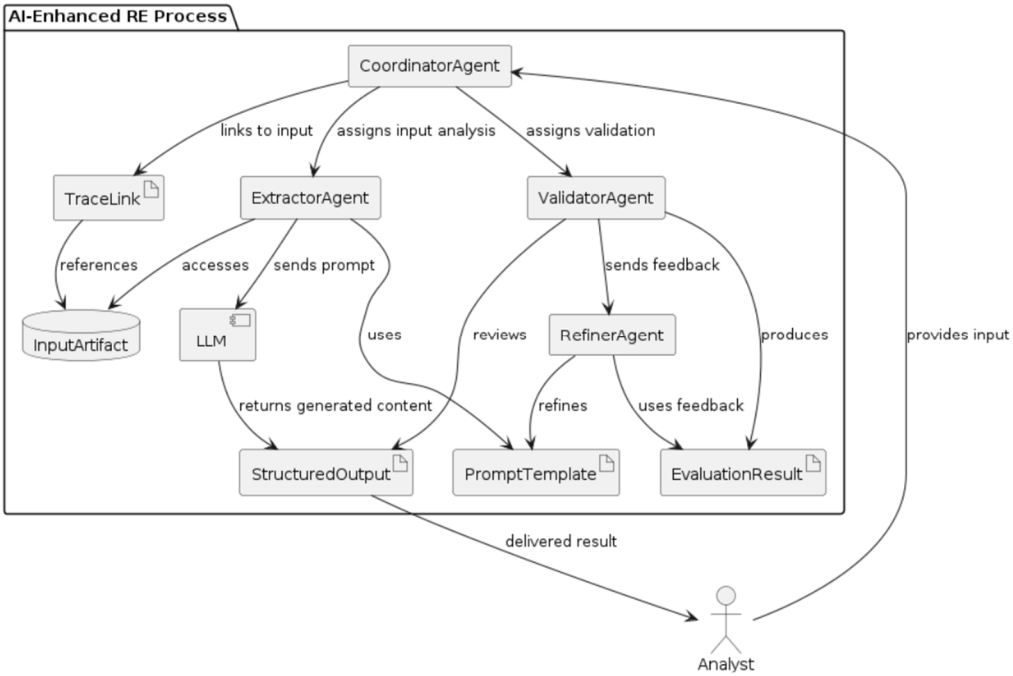

Figure 2: Conceptual agent-based architecture for orchestrating AI-assisted requirements engineering tasks.

The architecture includes the following agents and artifacts:

- ExtractorAgent – accesses the initial InputArtifact (e.g., textual descriptions, diagrams, images, domain documents) and applies a predefined Prompt Template to formulate structured prompts. These prompts are submitted to the large language model, which returns generated content. Validator
- Agent – reviews the Structured Output received from the LLM and produces an Evaluation Result based on defined quality criteria. It also forwards feedback to the Refiner
- Agent. RefinerAgent – receives feedback from the ValidatorAgent and refines the Prompt Template accordingly. It may also use information from Evaluation Result to improve future prompt formulations. Coordinator
- Agent – orchestrates the entire process by managing task delegation (input analysis, validation), linking generated outputs to their sources through Trace Link, and returning the final structured results to the Analyst. Trace
- Link – maintains references between the original Input Artifact and the corresponding Structured Output, ensuring transparency and traceability throughout the process. The Coordinator Agent manages the flow between these agents and maintains traceability by linking requirements to their sources.

6. Preliminary results

Initial experimentation was conducted to test the effectiveness of prompt engineering strategies for LLM-based user story generation. Experiments were carried out using multiple prompting styles: few-shot, one-shot, and zero-shot, across different data sources such as textual descriptions, stakeholder notes, and document-based requirements across several real-world projects.

The experiments included two distinct cases:

Case 1 – Small-scale project: A domain with 34 user stories. The input set consisted of 8 files, including stakeholder notes, diagrams, and wireframes. All three prompting strategies—zero-shot, one-shot, and few-shot—were applied to the same input data.

Case 2 – Larger system: A more complex domain with 70 user stories. The input set included over 60 diverse files such as stakeholder notes, screenshots, diagrams, and structured documentation. Again, all three prompting strategies were tested for comparative purposes.

In both cases, prompts were executed using the ChatGPT web interface. A golden standard set of reference user stories was manually constructed based on original domain requirements, and used as ground truth for comparison. Generated outputs were evaluated by comparing requirement coverage. Some key results include:

- Few-shot prompting demonstrated higher alignment with the golden standard than one-shot or zero-shot strategies.
- Input sensitivity was observed: clearer, more structured inputs led to more accurate outputs.
- In one of the evaluated projects, few-shot prompting achieved up to 94% requirement coverage, with results varying based on the applied prompting strategy.
- Challenges included variability in LLM output phrasing, redundancy, and hallucination of features.

These findings suggest that prompt design influences output quality and provide a foundation for further evaluation and iterative refinement of the methodology.

7. Evaluation plan

The evaluation of the proposed methodology will be conducted in several phases. The goal is to assess the feasibility of prompt-based requirement extraction and the usability of the methodology in Agile Requirements Engineering contexts. The evaluation will focus on the following dimensions:

Prompt effectiveness evaluation

Experiments will compare different prompt strategies (few-shot, one-shot, zero-shot) applied to various input formats such as textual descriptions, stakeholder notes, and diagrams. The generated requirements will be evaluated against a manually constructed golden standard by domain experts. The evaluation criteria will include: correctness of extracted requirements, alignment with INVEST framework, usefulness in real-world Requirements Engineering scenarios.

Scenario-based usability testing

Structured scenarios will simulate realistic Agile meetings and documentation workflows. Analysts will be provided with multi-format input artifacts and will assess the usability of the generated outputs in activities such as decision-making, requirement clarification, and backlog formulation.

Comparison with baseline methods

Where applicable, baseline methods—such as manually written requirements or LLM-generated outputs without prompt guidance—will be used for comparative analysis to evaluate the added value of the structured methodology.

Feedback-driven iteration

Expert and analyst feedback will be used to identify opportunities for improving prompt templates, metamodel structures, and process workflows. This refinement loop aims to ensure the methodology remains adaptable and context-aware.

8. Future work

The current research has focused on developing the core components of the methodology: prompt templates, a conceptual metamodel, and initial experimentation with LLM-based requirement generation. Future work will extend and deepen the evaluation and implementation of these components in the following directions:

Agent-based architecture framework

The proposed agent-based framework will be described and demo could be implemented. Each agent (Extractor, Validator, Refiner) will be assigned distinct roles and integrated through structured interaction protocols.

Tool prototype development

A working prototype will be developed to support input submission, prompt execution, output visualization, and traceability mapping. The interface will be designed for Requirement Engineering analysts working in Agile contexts.

Expanded evaluation scenarios

Additional use cases from different domains will be used to test generalizability. Structured interviews and usability feedback will help refine the tool.

9. Conclusion

This paper presented a hybrid methodology that combines structured prompt engineering, a conceptual metamodel, and a conceptual agent-based architecture to support requirement elicitation from heterogeneous sources in Requirements Engineering. The proposed approach addresses the lack of structure, traceability, and quality control in current LLM-based requirement generation by integrating prompt patterns and traceable transformation workflows. Preliminary experiments demonstrate that prompt design impacts output coverage and quality, while the conceptual architecture provides a foundation for modular orchestration and continuous improvement. Future work will focus on implementing a working prototype, expanding evaluation scenarios, and exploring domain-specific adaptations to further validate the methodology and its applicability in real-world software development environments.

Acknowledgements

Work is supervised by Assoc. Prof. Dr. Lina Čeponienė, Kaunas University of Technology.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT, Grammarly in order to: Grammar and spelling check, paraphrase. After using this tool, the author reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] I. Sommerville, *Software Engineering*, 9th ed., Pearson, Boston, 2011.

[2] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*, 4th ed., Springer, Cham, 2017.

[3] [3] K. Beck *et al.* Manifesto for agile software development, *Agile Manifesto*, 2001. URL: https://agilemanifesto.org/

[4] P. Sawyer, I. Sommerville, and S. Viller. "Requirements process improvement through the phased introduction of good practice." *Software Process: Improvement and Practice* 3.1 (1997): 19-34. doi: https://doi.org/10.1002/(sici)1099-1670(199703)3:1%3C19::aid-spip66%3E3.0.co;2-x.

[5] A. Mehraj, Z. Zhang, and K. Systä, A Tertiary Study on AI for Requirements Engineering, in: D. Mendez and A. Moreira (Eds.), *Requirements Engineering: Foundation for Software Quality,* volume 14588 of Lecture Notes in Computer Science, Springer, Cham, 2024, pp. 159–177. doi: https://doi.org/10.1007/978-3-031-57327-9_10.

[6] C. Arora, J. Grundy, and M. Abdelrazek, Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs, in: A. Nguyen-Duc, P. Abrahamsson, and F. Khomh (Eds.), *Generative AI for Effective Software Development*, Springer, Cham, 2024, pp. 129–148. doi: https://doi.org/10.1007/978-3-031-55642-5_6.

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

[8] M. A. Umar and K. Lano, "Advances in automated support for requirements engineering: a systematic literature review." *Requirements Engineering* 29.2 (2024): 177-207. doi: https://doi.org/10.1007/s00766-023-00411-0.

[9] H. Cheng *et al.*, Generative AI for Requirements Engineering: A Systematic Literature Review, *arXiv (Cornell University)* (2024). doi: https://doi.org/10.48550/arxiv.2409.06741.

[10] A. Vogelsang and J. Fischbach, Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline, *arXiv (Cornell University)* (2024). doi: https://doi.org/10.48550/arXiv.2402.13823.

[11] M. Snoeck and Y. Wautelet, "Agile MERODE: a model-driven software engineering method for user-centric and value-based development." *Software and Systems Modeling* 21.4 (2022): 1469-1494. doi: https://doi.org/10.1007/s10270-022-01015-y.

[12] M. Huss, D. R. Herber, and J. M. Borky, "An Agile Model-Based Software Engineering Approach Illustrated through the Development of a Health Technology System." *Software*, 2.2 (2023): 234-257. doi: https://doi.org/10.3390/software2020011.

[13] P. Spoletini and A. Ferrari, The Return of Formal Requirements Engineering in the Era of Large Language Models, in: D. Mendez and A. Moreira (Eds.), *Requirements Engineering: Foundation for Software Quality, volume 14588 of Lecture Notes in Computer Science,* Springer, Cham, 2024, pp334-353. doi: https://doi.org/10.1007/978-3-031-57327-9_22.

[14] L. Belzner, T. Gabor, and M. Wirsing, Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study, in: B. Steffen (Ed.), B*ridging the Gap Between AI and Reality, volume 14380 of Lecture Notes in Computer Science,* Springer, Cham, 2023, pp. 355–374. doi: https://doi.org/10.1007/978-3-031-46002-9_23.

[15] M. A. Sami, Z. Rasheed, M. Waseem, Z. Zhang, T. Herda, and P. Abrahamsson, Prioritizing Software Requirements Using Large Language Models, *arXiv (Cornell University)* (2024). doi: https://doi.org/10.48550/arxiv.2405.01564.

[16] A. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly* 28.1 (2004): 75-105. doi: https://doi.org/10.2307/25148625.

[17] P. Johannesson and E. Perjons, *An Introduction to Design Science*, 1st ed. Springer, Cham, 2014. doi: https://doi.org/10.1007/978-3-319-10632-8.