ISSN: 2692-5397
DOI: 10.33552/MCMS.2026.07.000683
Modern Concepts in Material Science
Iris Publishers
Research Article
Copyright © All rights are reserved by Torsten Hielscher

LLM-based Requirements and Systems Engineering

Mike Moore[1], Torsten Hielscher[2*]

[1]*MiniTec GmbH, Schönenberg-Kübelberg, Germany*

[2]*Department of Mechanical Engineering (FHK), Institute QM3, University of Applied Sciences Kaiserslautern, Germany*

***Corresponding author:** Torsten Hielscher, Department of Mechanical Engineering (FHK), Institute QM3, University of Applied Sciences Kaiserslautern, Germany.

Received Date: October 20, 2026

Published Date: November 17, 2026

Abstract

In the special-purpose machinery sector, which is dominated by small and medium-sized enterprises, the effort required to analyze industrial customer requirements specifications (CRSs) is increasing. This is driven by both the growing volume of externally imposed requirements and the increasing complexity and interdependencies of the requirements contained in these documents. Against this background, a research project investigated the extent to which large language models (LLMs) can perform tasks in requirements engineering (RE) and systems engineering (SE) and support the processing of CRSs, which has so far been largely manual. An artificial intelligence (AI)-based assistance system comprising six specialized agents was developed for this purpose. These agents extract, check, and structure requirements, establish links between them to support traceability, and assign them to system elements using the requirements, functional, logical, and physical (RFLP) framework. The system also generates Mermaid diagrams inspired by the Systems Modeling Language (SysML) to represent system structures and requirements, as well as drafts of system requirements specifications (SRSs).

The potential of the assistance system was investigated in a prototype study at MiniTec GmbH, a manufacturer of machinery and industrial systems. Two locally deployed and two cloud-based LLMs were used. The system design and selection of AI models placed particular emphasis on protecting sensitive customer and project data and on the applicable data processing conditions. Nextcloud, OpenWebUI, and Obsidian provide interfaces for information exchange and interaction with the assistance system. Agent orchestration and the control of workflows, tool calls, and data flows are handled by n8n; Supabase provides the shared database. An integrated human-in-the-loop process enables engineers to review the technical content of the generated results, edit them, and approve them. This paper describes the architecture, data flow, and technical implementation of the assistance system and investigates its application using actual industrial project documentation.

Keywords: Large Language Models; Requirements Engineering; Systems Engineering; Retrieval-Augmented Generation; Multi-Agent Systems; Model-Based Systems Engineering

Introduction

At MiniTec, customer inquiries and customer requirements specifications (CRSs) for special-purpose machinery range from single-page briefs to documents comprising several hundred pages. Their manual analysis requires considerable personnel resources. Relationships between individual requirements and their connections to existing technical solutions are often not systematically documented and maintained throughout the project. Established methods and procedures from requirements engineering (RE) and systems engineering (SE) provide a basis for addressing these tasks in a structured manner. RE focuses in particular on eliciting, checking, and maintaining requirements [1,2]. SE takes a holistic, interdisciplinary approach to the development of technical systems [3]. In

This work is licensed under Creative Commons Attribution 4.0 License

model-based systems engineering (MBSE), requirements, system structures, and system behavior are linked within consistent system models [4].

However, the consistent application of these methods requires considerable effort and can place demands on the available personnel and time, particularly in the special-purpose machinery sector with its many small and medium-sized enterprises. When CRSs are extensive, this makes it difficult to apply established RE and SE methods consistently in day-to-day project work. Large language models (LLMs) have the potential to support selected tasks within these processes and reduce manual processing effort. Their use in RE requires, among other things, sufficient domain context and precisely formulated instructions [5]. Industrial applications also require relevant documents, company-specific rules, and technical master data to be made available as needed.

The research project therefore aims to support the processing of customer requirements using artificial intelligence (AI). An assistance system was developed to help engineers analyze CRSs, process requirements in a structured manner, and prepare draft system requirements specifications (SRSs). The engineers' technical expertise is explicitly integrated into the workflow. This paper describes the system architecture and technical implementation, as well as its prototype evaluation using actual industrial project documentation. The focus is on the division of tasks among specialized agents, the integration of company-specific and project-specific knowledge, and the findings on processing times, costs, and practical limitations.

System Architecture and Prototype Study

The assistance system is designed as a multi-agent system with a two-stage workflow. Its architecture draws on the open-source reference architecture proposed by Lambert et al. [6], which describes an approach to deploying local LLMs in compliance with data protection requirements. This approach was adapted to the processing of requirements and technical specifications in special-purpose machinery engineering.

Figure 1 illustrates the components of the assistance system and the data exchanged between them. In the first stage, Agents 1 to 3 extract, check, and structure the requirements from the CRS. In the second stage, an engineer works interactively with the results using Agents 4 to 6. Agent 4 assigns requirements and other relevant entries to system elements according to the requirements, functional, logical, and physical (RFLP) framework and proposes possible relationships between them. On this basis, Agent 5 generates block definition diagrams (BDDs) and requirement diagrams (req diagrams) inspired by Systems Modeling Language (SysML) [4]. Agent 6 prepares draft SRSs based on the modeled system structure, item master data, and a solution concept provided by the engineer. The specification follows the provisions of VDI 2519 Part 1 [7].

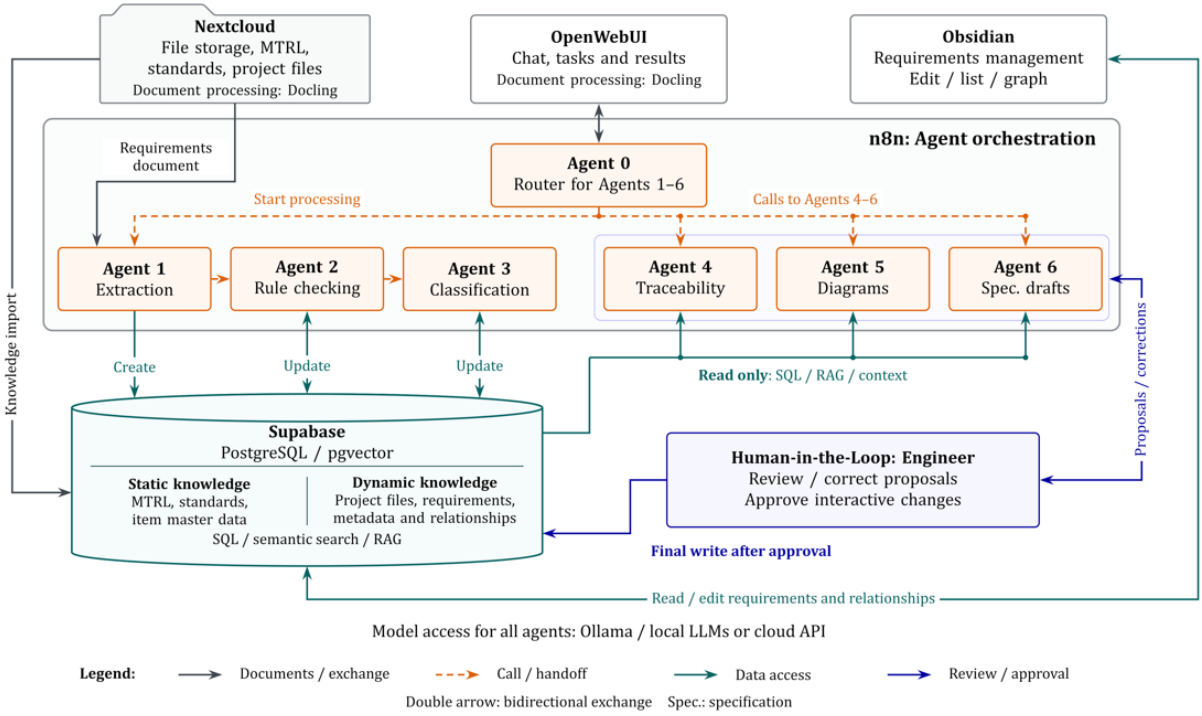


Figure 1: System architecture.

Each agent is assigned specific instructions and the tools required for its task. The system prompts are designed around requirements elicitation, specification, and accompanying modeling within the V-model according to VDI/VDE 2206 [8]. Roles, tasks, tool-use rules, processing instructions, and output formats are specified separately. This functional division allows individual process steps to be checked independently and adjusted as needed.

The results of Agents 1 to 3 are initially stored as AI-generated proposals. Changes arising from interactive processing with Agents 4 to 6 are committed only after an engineer has reviewed their technical content and approved them. This human-in-the-loop approach is intended to prevent AI-generated proposals from being used in subsequent project work without review [9]. Technical assessment and approval of the results, as well as the verification and validation activities of the V-model, remain the engineer's responsibility. The multi-agent system is orchestrated using n8n. The platform controls agent calls and data transfer between individual process steps. It also prepares document text, cleans search results, and provides the information required for each task. Targeted input selection and preparation are intended to avoid unnecessarily large amounts of context or information of limited relevance to the task [10]. Agent 0 selects which agents to call.

The assistance system provides several interfaces for interacting with external systems and supplying the required knowledge. In addition to relevant standards and project-specific documentation, Nextcloud stores the MiniTec guidelines (MTRLs) developed for the assistance system. These contain company-specific rules for checking customer requirements. OpenWebUI provides the chat interface to n8n, while Obsidian is used to edit and visualize requirements and their relationships. Supabase, based on PostgreSQL and pgvector, provides centralized data storage. Docling converts documents into machine-readable Markdown text while taking their layout and tables into account [11].

The relevant content is made available centrally through Supabase and is also vectorized using the Qwen3-embedding:8b embedding model. Depending on the task, access is provided through Structured Query Language (SQL) queries, semantic search as part of retrieval-augmented generation (RAG), or a combination of both methods. Knowledge management distinguishes between static company knowledge and dynamic project knowledge. Static company knowledge includes MTRLs, item master data, standards, and guidelines. Dynamic project knowledge includes, in particular, CRSs, project emails, agreements, and product data sheets whose contents may change over the course of a project. Requirements are stored under a unique identifier (ID) and linked to their source, review notes, classification, and associated relationships [1,12]. This creates a structured database that supports both the traceability of individual requirements and their placement within the technical system context.

The assistance system was deployed as a prototype at MiniTec GmbH, a medium-sized manufacturer of machinery and industrial systems. Two locally deployed and two cloud-based LLMs were connected for the study. Local inference used Qwen 3.6:35B and Gemma 4:31B through Ollama with Q4_K_M quantization. Gemini 2.5 Pro and Gemini 3.5 Flash were used as cloud-based models through Google Vertex AI with zero data retention (ZDR) and European Union (EU) endpoints. The virtual test environment provided an NVIDIA RTX 8000 with 48 GB of virtual graphics processing unit (vGPU) memory and 64 GB of random-access memory (RAM).

Three representative projects were selected from 37 completed customer projects. Their CRSs comprised 15, 25, and 99 pages of text, respectively. The results of Agents 1 to 3 were examined using 15 manually assessed reference requirements for each of the three CRSs. The interactive Agents 4 to 6 were examined using the longest CRS. To test Agent 2, ten prototype guidelines were initially created and supplemented with 90 additional guidelines that were not project specific. This was intended to simulate a larger collection of company guidelines. The links proposed by Agent 4 were assessed qualitatively. The results of Agents 5 and 6 were assessed against reference solutions using an LLM-as-a-Judge approach. GPT 5.6 Luna was used as the evaluation model. The diagrams generated by Agent 5 were assessed using criteria for syntactic and logical correctness adapted from the MermaidSeqBench sequence diagram benchmark [13]. The draft SRSs generated by Agent 6 were assessed in terms of traceability, retrieval of the necessary data, specification quality, and identification of missing information.

The runtimes of the automated processing steps were also recorded. The costs of application programming interface (API)-based model calls were calculated using the token consumption of selected configurations and the token prices applicable at the time of the experiments. For the longest CRS examined, the costs of interactive use were extrapolated to the entire project based on estimated numbers of calls.

Results and Discussion

The following sections first present the results of the individual agents. The observed processing times and costs are then analyzed. Finally, practical experience gained during prototype development and testing is discussed, together with the resulting limitations.

Prototype Results

Listing 1 shows an example record containing the results of Agents 1 to 4. The original requirement source in the CRS, a possible conflict with a MTRL, the classification, and the proposed relationships are stored together with the requirement. The information relevant to the technical review therefore remains available in one central location.

```
{
  "ID": 75, "Obligation": "Mandatory", "RFLP": "R",
  "Source": "LH_V1, Section 2.1, Project description",
  "Requirement": "System availability according to VDI 3423: at
     least 99%.",
  "Assessment": "MTRL-009: No guaranteed system availability as
     standard.",
  "Status": "Red",
  "Comment": "Contractual clarification required.",
  "VDI_Section": "3.8.6",
  "VDI_Title": "Reliability",
  "Traceability": "<<contains>> ID 111; <<satisfies>> ID 48;
     <<derives>> ID 82"
}
```


Listing 1: Results of Agents 1–4.

Agent 6 prepared a draft SRS based on the relationships proposed by Agent 4, the Mermaid code generated by Agent 5, the item master data, and the solution concept provided by the engineer. It checked whether the solution concept covered the existing requirements. For requirements that had not yet been addressed, or where information was missing or unclear, it marked the relevant entries with [Clarification required] and asked the engineer targeted questions. The engineer's additions were incorporated into the subsequent elaboration of the draft.

The prototype implementation thus demonstrates how the structured capture of requirements, their assignment to system elements, the visualization of their relationships, and the preparation of draft specifications can be combined in a continuous workflow. Technical review and additional input by the engineer form an integral part of this process.

Processing Times and Costs

For a single processing run of the longest CRS through Agents 1 to 3, the cumulative processing time was approximately 176 minutes using Gemma 4:31B and approximately 31 minutes using Gemini 3.5 Flash. These values are based on individual trials using different versions of the prompts. They therefore do not support an unqualified direct comparison of the models but provide an indication of the runtimes observed under the respective experimental conditions.

Diagram generation took approximately two to four minutes with the locally deployed configurations. The cloud-based models required less than one minute. The measured times apply only to the configurations examined and exclude the additional effort required for technical review and correction of the results. For the longest CRS examined, API costs of approximately EUR 40 and electricity costs of approximately EUR 6 for local graphics processing unit (GPU) inference were calculated. The electricity cost estimate followed Pan et al. [14] and took into account the inference duration, GPU power consumption, and assumed electricity price. Both cost figures cover a single processing run of the CRS and the estimated interactive use over the course of the project. Hardware acquisition or provisioning, other IT operations, development and maintenance of the assistance system, and personnel effort for technical review and correction were excluded. The calculated amounts should therefore be interpreted as usage-related estimates for the scenarios examined, rather than as the total costs of the system.

Practical Findings and Limitations

The prototype application of the assistance system required repeated testing and adjustment of the n8n workflows, database queries, and system prompts. The observed results were influenced by more than the choice of language model. Input preparation, input size, and the available tools also affected the quality of the results. Consequently, the overall system must be tuned by considering the interaction between the model, database, tools, and process control.

For processes that can be clearly described by fixed rules, direct implementation in n8n proved useful. The LLMs were primarily used for content analysis and language generation. This functional division distinguishes deterministic process steps from tasks requiring context-dependent interpretation.

When checking company-specific guidelines, an excessively high retrieval threshold prevented a relevant MTRL from being retrieved. As a result, the requirement concerned was incorrectly classified as unproblematic. This observation illustrates that incomplete knowledge provision can directly lead to incorrect assessments. The quality of the results therefore depends on both the reasoning capabilities of the LLM and reliable access to relevant information. Short search excerpts also proved insufficient in some cases when preparing draft SRSs. In particular, retrieved passages from data sheets covering multiple product variants sometimeslacked the information needed to identify the correct variant. The system architecture was therefore extended to allow access to the full document text when needed. This mechanism is part of the op-

timized architecture.

In addition, the relationships proposed by Agent 4 between requirements and system elements were not technically correct in all cases. Careful technical review therefore remains necessary, particularly when linking requirements, system elements, and specific technical solutions. The results also highlight the limitations of the study. The evaluation is based on three deliberately selected customer projects, which were also used to refine the system, and a limited number of manually assessed reference requirements. The findings therefore cannot be readily generalized to other companies, industries, CRS structures, or system configurations. In addition, the interactive agents were examined partly through qualitative assessment and partly against reference solutions assessed by another LLM. No independent comparison of the LLM assessments with human assessments is documented. A comprehensive quantitative assessment of technical correctness and reliability under varying operating conditions was outside the scope of the study. The system must be tested again with additional CRSs and after changes to models, prompts, or workflows. The time required for human review was likewise not recorded systematically. The available results therefore provide only a limited basis for conclusions about the actual reduction in overall processing effort.

Conclusion

An AI-based assistance system was developed as part of the research project and tested as a prototype in an industrial special-purpose machinery setting. The results show that locally deployed and cloud-based LLMs can be used for selected tasks in RE and SE. The multi-agent system supports the extraction, checking, and structuring of requirements, proposes relationships with system elements, and generates diagrams inspired by SysML, as well as draft SRSs. Orchestration with n8n and a shared, structured database enable these tasks to be integrated into a continuous workflow. The combination of SQL queries, semantic search, and supplementary access to full document texts supports the provision of company-specific and project-specific information. The integrated human-in-the-loop process ensures that technical review and approval of the results by an engineer are included in the workflow.

The study also demonstrates that the quality of the results depends on the model used, the particular CRS, and the configuration of the overall system. In particular, document preparation, selection of relevant context, process control, and system prompt design must be coordinated and validated together. The sensitivity of the overall system to changes in models, prompts, and processing settings could increase the effort required for subsequent verification and validation. Even capable cloud-based models produced incorrect results in some cases. The generated content should v reviewed before it is adopted.

The prototype study provides indications of the approach's potential but does not yet support general conclusions about the system's reliability or economic viability. Further assessment requires more extensive quantitative studies involving a larger number of projects, consistent experimental conditions, and systematic quality metrics. These should focus in particular on the technical correctness of the extracted requirements and proposed relationships, the quality of the generated specifications, and the time required for human review and correction. Only on this basis, the extent to which the assistance system actually reduces the overall effort in industrial RE and SE can be assessed.

Acknowledgement

The authors thank the University of Applied Sciences Kaiserslautern for supporting the publication and MiniTec GmbH for providing the hardware and project documentation.

Declaration of generative AI and AI-assisted technologies in the writing process

During preparation of this manuscript, the authors used OpenAI Codex to assist with drafting and revising portions of the text and with creating LaTeX/TikZ code for schematic figures. All AI-assisted content is subject to review and revision by the authors, who take full responsibility for the final manuscript.

References

1. B Bender, K Gericke (2021) "Entwickeln der Anforderungsbasis: Requirements Engineering", In: Pahl/Beitz Konstruktionslehre pp. 169-209.
2. D Göhlich, T A Fay (2021) "Arbeiten mit Anforderungen: Requirements Management", In: Pahl/Beitz Konstruktionslehre pp. 211-229.
3. I Gräßler, C Oleff (2022) "Einführung in Systems Engineering", In: Systems Engineering: Verstehen und industriell umsetzen pp. 1-27.
4. I Gräßler, C Oleff (2022) "Voraussetzungen für die industrielle Umsetzung", In: Systems Engineering: Verstehen und industriell umsetzen pp. 29-82.
5. K Ronanki, S Arvidsson, J Axell (2025) Prompt Engineering Guidelines for Using Large Language Models in Requirements Engineering.
6. M Lambert, T Schuster, N Döring, R Krüger (2025) Datenschutzkonformer LLM-Einsatz: Eine Open-Source-Referenzarchitektur.
7. VDI. VDI 2519 Blatt 1: Vorgehensweise bei der Erstellung von Lasten-/Pflichtenheften. 2001. VDI record.
8. VDI/VDE. VDI/VDE 2206: Entwicklung mechatronischer und cyber-physischer Systeme. 2021. VDI record.
9. M A Abbasi, P Ihantola, T Mikkonen, N Mäkitalo (2025) Towards Human-AI Synergy in Requirements Engineering: A Framework and Preliminary Study. 2025.
10. K Hong, A Troynikov, J Huber (2025) Context Rot: How Increasing Input Tokens Impacts LLM Performance. Technical report.
11. N Livathinos, C Auer, M Lysak, A Nassar, M Dolfi (2025) Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion.
12. K Pohl (2025) Requirements Engineering: Fundamentals, Principles, and Techniques. Springer-Verlag Berlin Heidelberg.
13. B Shbita, F Ahmed, C DeLuca (2026) MermaidSeqBench: An Evaluation Benchmark for NL-to-Mermaid Sequence Diagram Generation.
14. G Pan, V Chodnekar, A Roy, H Wang (2025) A Cost-Benefit Analysis of On-Premise Large Language Model Deployment: Breaking Even with Commercial LLM Services.