SuPReA: Modular Agent-Based Generation of Requirements Artifacts Using LLMs

Jacek Dajda[1,†], Maria Eckes-Kondak[1,*,†], Krzysztof Wysocki[2,†], Jakub Żywiecki[2,†] and Bartosz Pawłowski[1,†]

[1]University of Krakow, Mickiewicza Ave. 30, 30-059 Krakow, Poland
[2]Warsaw University of Technology, Plac Politechniki 1, 00-661 Warsaw, Poland

Abstract

We present SuPReA (System Supporting Project Requirements Analysis), an open-source, web-based tool that supports early-stage requirements engineering by generating a coherent package of specification artifacts from a short natural-language project description. SuPReA orchestrates lightweight, stateless AI agents, each responsible for a specific artifact type such as product vision, functional requirements, personas, use cases, UML diagrams, data models, UI sketches, and supporting artifacts such as scenarios or roadmaps, using a configurable prompt template and structured JSON outputs to enable downstream rendering and interoperability. The JSON-based output format is embedded in the prompt template to guide the structure of model responses rather than to enforce a formal schema definition. To improve robustness, SuPReA validates generated outputs and automatically re-queries the underlying model when responses are incomplete or malformed, enabling refinement-oriented workflows. The tool provides an artifact workspace organized by projects, where users can iteratively edit, regenerate, prioritize, and export selected elements, supporting collaboration in exploratory design. We demonstrate SuPReA's architecture and end-to-end workflow and report an initial verification focused on output completeness and format reliability. The paper focuses on the tool design and workflow rather than on a full empirical evaluation.

Keywords

Requirements Engineering, AI Agents, Generative Models, Automation, UML

1. Introduction

Requirements Engineering (RE) remains a critical activity in software development, shaping how systems respond to evolving stakeholder needs and organizational constraints. Analysts and requirements engineers elicit, refine, and specify requirements in ways that guide implementation while adapting to rapidly changing project contexts, particularly in agile and hybrid environments [1, 2, 3]. As software systems grow in complexity and delivery cycles shorten, early-stage work such as defining product vision, identifying actors and personas, developing usage scenarios, and sketching initial system features and structures (hereafter referred to as early RE activities) becomes increasingly demanding [4, 5].

Recent work points to a growing role of Artificial Intelligence (AI), particularly large language models (LLMs), in supporting early RE activities (which includes vision statement, goals, product motto, actors, main requirements). Prior studies report that LLMs can provide analysts with elicitation support, clarification, prioritization, and the generation of early-stage requirements artifacts [6, 7]. LLMs have also shown potential in generating more formal artifacts such as UML class diagrams, sequence diagrams, and structured requirement descriptions; however, issues of structure, completeness, and cross-artifact consistency remain recurring challenges [8, 9, 10]. These limitations are especially visible when artifacts are generated in isolation, without tool support for iterative refinement and without mechanisms that help keep related outputs coherent over multiple generations [8, 10].

Joint Proceedings of REFSQ-2026 Workshops, Doctoral Symposium, Posters Tools Track, and Education and Training Track. Co-located with REFSQ 2026. Poznan, Poland, March 23-26, 2026

*Corresponding author.
†These authors contributed equally.
dajda@agh.edu.pl (J. Dajda); meckes@agh.edu.pl (M. Eckes-Kondak); krzyswys3@gmail.com (K. Wysocki); jakubzywiecki1@gmail.com (J. Żywiecki); bartek.pawlowski00@gmail.com (B. Pawłowski)
0000-0001-8617-4981 (J. Dajda); 0000-0002-9049-4440 (M. Eckes-Kondak)

Existing AI-assisted RE solutions typically focus on individual tasks (e.g., user story generation or UML synthesis) and rarely provide a unified, extensible workflow for producing multiple early-stage artifacts as a coherent, modifiable package. This creates a gap between the capabilities of LLMs and the practical needs of analysts, who must continuously align diverse specification elements while iterating with stakeholders and development teams.

To address this gap, we introduce SuPReA (System Supporting Project Requirements Analysis), an open-source tool designed to support requirements engineers and business analysts in producing coherent sets of early requirements artifacts using a modular, agent-based approach. The main contribution lies in combining modular, prompt-driven agent orchestration with multimodal artifact generation and built-in validation mechanisms, offering a practical and extensible tool for early-stage requirements engineering.

The paper first introduces the tool concept, then outlines its architecture and workflow, and finally presents a use case with evaluation results and future work.

2. Tool Overview

A core feature of SuPReA is to support requirements analysis. The primary users of SuPReA are requirements engineers and business analysts; other roles, such as developers or designers, typically consume generated artifacts without directly interacting with the tool. Users can automatically generate a whole set of different project artifacts and further work on them. The tool validates outputs and automatically re-queries the LLM when results are incomplete or malformed. This feedback loop is intended to mitigate known weaknesses of generative models while keeping analysts in control of the evolving specification. Validation ensures structural correctness, while semantic consistency and refinement remain the analyst's responsibility.

The term *agent* in SuPReA refers to a software execution module responsible for a single AI-based generation task. It should not be confused with actors in requirements engineering or UML, which denote roles or entities interacting with a system. We retain agent to reflect the current convention in AI toolchains and agent-oriented frameworks.

Each artifact type (e.g. personas, use cases, UI structures, UML diagrams, data models, scenarios, or roadmaps) is generated by a lightweight, stateless AI agent instantiated on demand and operating independently. Agents rely on configurable prompt templates and produce structured JSON-based outputs that facilitate parsing, storage, editing, and downstream processing. This design emphasizes extensibility through prompt and agent configuration, rather than rigid, schema-centric enforcement. The tool is implemented as a web-based application providing a project-oriented workspace for interactive artifact generation, review, editing, regeneration and collaboration.

3. Workflow and General Architecture

The overall workflow implemented by SuPReA is summarized in Figure 1. To initiate the process, the user provides a short, plain-text description of the envisioned system, reflecting the typical level of detail available at the beginning of a project. Figure 1 illustrates the complete generation and validation loop from user input to artifact refinement.

Based on this description, SuPReA activates a set of lightweight, task-oriented AI agents, each dedicated to producing a specific type of artifact. The selection of agents is configuration-driven and can be adjusted per project, allowing users to control which artifact types are generated. These agents operate independently and are instantiated on demand, which allows the system to remain flexible and extensible as new artifact types or generation strategies are introduced.

Each user request is routed through a controller responsible for:

- Selecting the appropriate agent for the required artifact (e.g., personas, features, use cases, UML diagrams, data models, UI structures).

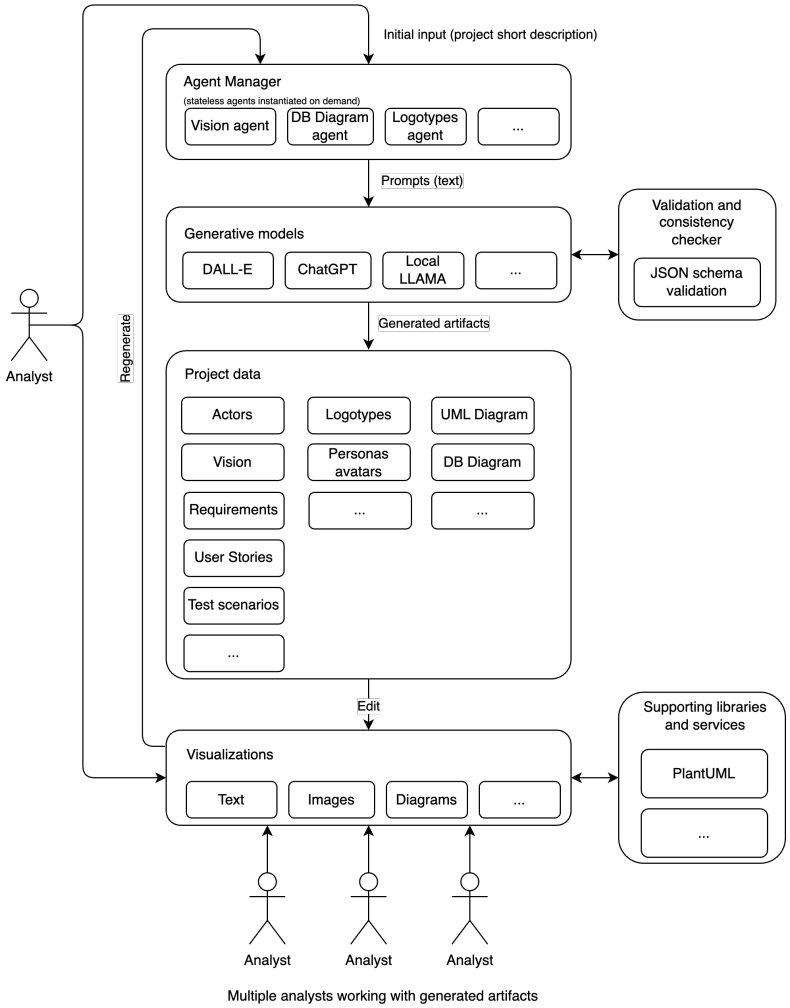

Figure 1: Overview of the SuPReA architecture and workflow: from initial analyst input to generation, validation, and collaborative refinement of multimodal artifacts.

- Constructing the prompt using the modular template, which embeds the task description, user-provided context, and expected output format.
- Invoking the designated model, either local or cloud-based, depending on the task (e.g., textual LLMs for structured artifacts, image generators for UI sketches).
- Validating the generated output to ensure syntactic completeness and correct formatting.
- Triggering automatic re-querying when outputs are incomplete or inconsistent—a mechanism included to compensate for known LLM limitations reported in previous studies [8, 10].
- Storing successful results in the project workspace, where they are available for editing, regeneration, or export.

Validation rules and JSON output structures are embedded in the configurable prompt templates used by individual agents. The current validation focuses on structural correctness, while cross-artifact consistency is supported through shared contextual inputs rather than through a dedicated semantic consistency mechanism. All generated artifacts are stored within a project workspace and can be edited collaboratively. Users may inspect results through structured views, export them to external formats, or regenerate specific elements when additional details become available. This modular, agent-based design ensures that SuPReA can be extended with new artifact generators or additional LLMs without modifying the core workflow, reflecting the extensibility requirement identified in prior RE tools research.

A key feature of the workflow is its iterative refinement loop. Users may regenerate specific components without affecting others, allowing incremental improvement of the specification. When a user edits an artifact, SuPReA can suggest revised versions of related elements or offer regeneration options based on updated context. This mechanism enables analysts to quickly explore alternative design variants without manually rewriting large portions of documentation.

4. LLM Integration Using an Agentic Approach

SuPReA is based on a modular, agent-based architecture that enables the automated generation of diverse requirements engineering artifacts from minimal input. Each generated artifact, such as personas, use cases, UML diagrams, or UI sketches, is generated by a dedicated stateless AI agent that follows a uniform and configurable prompt template. These agents are instantiated dynamically, execute a single task, and terminate, enabling scalability, responsiveness, and extensibility.

To cover different artifact types the tool integrates multiple large language models (LLMs) and multimodal generation pipelines into a single, AI-assisted workspace.

Agents in SuPReA rely on a unified, template-driven prompt that is parametrized for each artifact type:

> Generate `{what}` for `{for_who}` creating an application for `{doing_what}`. Return the result **exactly** according to the following JSON-based format: `{expected_answer_format}`.
>
> Additional information: `{additional_info}`.

The template specifies the generation goal, intended user role, system purpose, and the expected structure of the output using a specific JSON-based format designed for each artifact type.

A central controller orchestrates the entire workflow. Upon receiving the user's project description, it identifies the appropriate artifact type, selects the best-suited generative model, constructs the prompt, validates the model output, and stores the results in a persistent project workspace. If the output is malformed or incomplete, the controller triggers re-querying with adjusted parameters implementing a feedback loop that mitigates common LLM limitations without user intervention. Adjusted parameters typically include stricter format instructions and minor prompt refinements to enforce valid structured output. Generated artifacts are stored within a project-centric workspace that allows users to inspect, refine, and export elements individually.

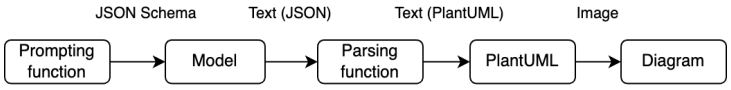

Figure 2: Example agentic pipeline for generating UML diagram

SuPReA supports multiple model types: textual LLMs such as GPT-4 mini and LLaMA 3.2 for generating narratives and structured descriptions; image generators like DALL·E 3 for visual artifacts such as personas or logos; and a hybrid pipeline for diagrams, where a textual representation (e.g., PlantUML) is first generated and then rendered as a visual asset. An example pipeline is illustrated in Figure 2. This design allows for comparing different LLM models and task types, a feature we leverage in our evaluation. Preliminary experiments included minor prompt variations to improve output stability and structural completeness.

5. Evaluation

The validation of the tool was performed by applying it to a set of representative example cases. For each case, the tool was used to generate a collection of design-related artifacts, which were subsequently subjected to expert assessment. The evaluation focused on examining the internal coherence, conceptual

plausibility, and perceived usefulness of the generated artifacts with respect to their potential to support subsequent activities in early-stage requirements engineering and system design.

In this section, we overview one of the tested business scenarios: restaurant management system. This scenario is well understood, yet sufficiently rich to exercise multiple requirements engineering activities, such as actor identification, functional decomposition, prioritization, and scenario-based reasoning. The system is intended to support daily restaurant operations, including employee management, shift scheduling, task assignment, and internal communication.

We initialized the project by providing SuPReA with a short natural-language description of the system goals and intended users, without defining actors, requirements, or models upfront (Figure 3).

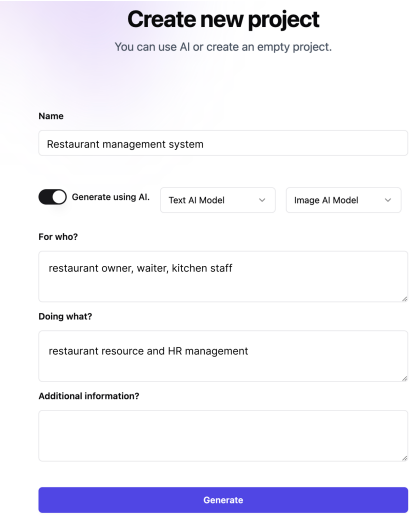

Figure 3: SuPReA project initialization interface used to provide an informal system description and target stakeholders.

Based on this minimal input, SuPReA generated a set of artifacts typically produced during early requirements engineering. In this use case, the system produced 18 functional requirements, 3 system actors, and 5 usage scenarios without requiring any manual pre-structuring of the input. All artifacts were generated during a single project initialization cycle.

The generated artifacts included a list of system actors, narrative usage scenarios, a short system vision description, and a structured collection of functional requirements with automatically assigned priorities. In addition, SuPReA produced auxiliary artifacts such as visual branding elements and preliminary UI-oriented content, which were intended to support early communication and shared understanding rather than detailed design.

The functional requirements covered core operational aspects of the restaurant domain, including user authentication, employee and role management, shift planning, task coordination, and internal messaging. The generated usage scenarios represented five typical operational situations, such as scheduling staff for a shift and coordinating tasks during service hours. Actors, scenarios, and requirements were mutually consistent and cross-referenced through the generated content, supporting a coherent early-stage specification. This consistency emerged from template alignment rather than from an explicit cross-artifact reasoning mechanism.

The requirements were presented in a dedicated requirements view (Figure 4), where each requirement included a short description and a priority level.

After initial generation, we reviewed the artifacts within the SuPReA interface and performed lightweight refinement activities typical for early requirements work. These included clarifying requirement descriptions, aligning terminology between actors and requirements, and adjusting scenario narratives. Selected artifacts were regenerated using SuPReA's iterative regeneration mechanism, allowing focused updates without restarting the entire generation process. Most refinements involved

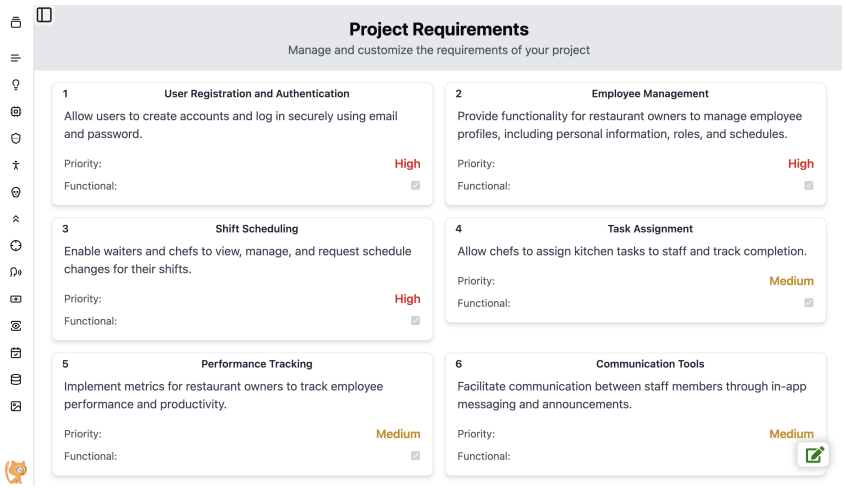

Figure 4: Requirements view generated by SuPReA for the restaurant management system, showing functional requirements with descriptions and priorities.

minor clarifications rather than structural changes.

The complete workflow from providing the initial textual description to obtaining a refined and exportable set of artifacts was completed in under 15 minutes. The resulting artifacts covered the majority of expected system features being a good starting point for further requirements specification.

From an implementation perspective, most structured artifacts were generated correctly on the first attempt when using instruction-tuned large language models. Smaller models more frequently produced incomplete or malformed outputs; however, these cases were handled by SuPReA's built-in output validation and regeneration mechanisms. Across iterations, text-based artifact generation proved more robust than image-based generation, which aligns with observations made during system development.

Overall, this exploratory use case demonstrates that SuPReA is technically capable of supporting early-stage requirements engineering by rapidly producing a coherent set of heterogeneous artifacts from informal input. While the evaluation is limited in scope and does not aim to measure productivity gains or quality improvements quantitatively, it provides evidence that the proposed tool and workflow are feasible and practically usable.

6. Conclusion and Future Work

The preliminary evaluation provided positive indications regarding the usefulness of the tool, which offers a practical and flexible approach to orchestrating generative models for early-stage requirements engineering tasks. By delegating the generation of individual artifacts to dedicated, stateless agents guided by configurable prompts, the platform supports automation while preserving analyst oversight and control over the overall process.

One of the strengths of this approach lies in the ability to selectively regenerate specific components of the specification without starting from scratch. This supports iterative refinement workflows, which are particularly important in agile and evolving project contexts. The separation of concerns across independent agents also facilitates parallelism and future extensibility of the system.

Nonetheless, several challenges remain. One open issue is the consistency between related artifacts. While SuPReA allows analysts to manually regenerate and adjust artifacts, there is currently no formal mechanism to track dependencies or ensure semantic coherence across components. For example, modifying a persona does not automatically propagate changes to associated user stories or UI models. Addressing this may require introducing lightweight cross-artifact consistency mechanisms or traceability links.

Another challenge is related to the quality of the generated output, which still depends heavily on the selected model and the specificity of the prompt. Although the regeneration mechanism mitigates

this to some extent, future improvements may include adaptive prompt tuning or hybrid pipelines that combine LLMs with rule-based validation.

Future research plans include a more thorough evaluation with properly designed metrics to assess the quality and completeness of the requirements. In this context, an evaluation should also be conducted that involves a focus group of analysts.

Acknowledgments

The research presented in this article received support from funds provided by the Polish Ministry of Science and Higher Education and assigned to AGH University of Krakow.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-5 in order to perform grammar and spelling check.

References

[1] L. Cao, B. Ramesh, Agile requirements engineering practices: An empirical study, IEEE Software 25 (2008) 60–67. doi:10.1109/MS.2008.1.

[2] A. De Lucia, A. Qusef, Requirements engineering in agile software development, Journal of Emerging Technologies in Web Intelligence 2 (2010) 212–220.

[3] U. S. Ramjaun, K. H. Abdullah, M. E. Rana, User centered agile software development: Emerging themes and future trends, in: Proc. 2023 4th Int. Conf. Data Analytics for Business and Industry (ICDABI), 2023, pp. 643–646.

[4] I. Udousoro, Effective requirement engineering process model in software engineering, Software Engineering 8 (2020) 1–5.

[5] K. E. Wiegers, J. Beatty, Software Requirements, Pearson Education, 2013.

[6] N. Marques, R. R. Silva, J. Bernardino, Using chatgpt in software requirements engineering: A comprehensive review, Future Internet 16 (2024) 180. doi:10.3390/fi16060180.

[7] J. S. Yeow, M. E. Rana, N. A. A. Majid, An automated model of software requirement engineering using gpt-3.5, in: Proc. 2024 ASU Int. Conf. Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), 2024, pp. 1746–1755. doi:10.1109/ICETSIS61505.2024.10459458.

[8] C. Arora, J. Grundy, M. Abdelrazek, Advancing requirements engineering through generative ai: Assessing the role of llms, in: Generative AI for Effective Software Development, Springer, Cham, 2024. doi:10.1007/978-3-031-55642-5_6.

[9] N. Feng, et al., Normative requirements operationalization with large language models, in: Proceedings of the 32nd IEEE International Requirements Engineering Conference (RE), 2024, pp. 129–141. doi:10.1109/RE59067.2024.00022.

[10] Z. Wang, J. Li, G. Li, Z. Jin, Chatcoder: Chat-based refine requirement improves llms' code generation, arXiv (2023). arXiv:2311.00272.

A. Online Resources

The tool is available at: https://github.com/jzywiecki/SuPReA

Demo video: https://youtu.be/jAfbAxT0spE