Enhancing Agile Workflows with AI-Driven, Sustainability-Aware Requirements Engineering: A Design Science Approach

Oshani Weerakoon[1(✉)], Shola Oyedeji[2], Md Abdus Samad[2], Abdulkadir Abubakar[2], Ahsan Ishan[2], Muhammad Shayan[2], Tuomas Mäkilä[1], and Erkki Kaila[1]

[1] University of Turku, Turku, Finland
osweer@utu.fi
[2] LUT University, Lappeenranta, Finland

Abstract. Generative AI presents growing opportunities across various fields, including Requirements Engineering (RE). RE, which is the backbone of software projects, drives the entire product development toward respective business goals. However, sustainability is not considered a primary need during the requirement elicitation, and as a result, software engineers are usually unable to envision the sustainability impacts of the products they build. To explore this gap, we introduce Reqwire, an AI-driven, sustainability-aware multi-agent system that (i) generates user stories from software requirement documents, (ii) enriches them with sustainability attributes, and (iii) integrates with common agile project management tools, like Jira. The system consists of specialized agents, namely as root, distributor, user story generator, and Jira. We followed a Design Science Research (DSR) approach under seven iterative cycles, incorporating feedback from an industry partner and academia to design and evaluate the Reqwire workflow. Our results indicate that Reqwire reduces manual effort by generating structured user stories, estimating story points, and assigning sustainability tags across five sustainability dimensions: environmental, economic, social, individual, and technical. The multi-agent-based framework enables integration with third-party tools, supporting consistent, systematic project tracking. Reqwire shows promise for enhancing agile workflows and promoting sustainable software practices from initial test rounds with the client.

Keywords: Requirement Engineering · Agile Development · User Stories · Generative AI · AI Agents · Sustainability

1 Introduction

Effective software systems begin with a clear understanding of user needs and requirements, and aligning functionality with user expectations helps prevent

G. Herzwurm et al. (Eds.): ICSOB 2025, LNBIP 574, pp. 212–229, 2026.
https://doi.org/10.1007/978-3-032-14518-5_17

functional discrepancies [35]. Requirements Engineering (RE) is the discipline that identifies a system's real-world objectives, intended functions, and operational constraints [21]. Within agile software development, user stories are commonly used [7] to capture requirements during the elicitation of requirements. They offer a lightweight format to concisely record the who, what, and why of a functionality and are readily understood by stakeholders with diverse technical backgrounds [36]. However, manually created user stories are prone to inconsistency, incompleteness, and error [10].

Furthermore, sustainability has emerged as an important factor in RE and software development because of the sustainability impacts of software systems solutions on people, the environment, and society [8]. Sustainability within software requirements, design, and development can be viewed from these five dimensions: environmental, social, technical, economic, and individual [27]. In software development, early design choices incorporated into requirements have an impact on whether the sustainability of a software system is enabled or impeded, which may lead to *sustainability debt* [4,24]. However, software development practitioners lack the understanding of how to operationalize sustainability during early RE stages and agile software development [25]. The absence of systematic tools to identify, prioritize, and link requirements to each of the sustainability dimensions at the same level as functional and non-functional related requirements [5,8,30], further contributes to this issue.

The recent advances in generative artificial intelligence (AI) and large language models (LLMs) provide a good opportunity for addressing both the challenges that come with manual user story generation and linking requirements (user stories) to sustainability during requirement elicitation and software development. Within RE, several recent studies have leveraged LLM-based approaches to explore the state-of-the-art RE tasks [31,33], such as generating elicitation interview scripts [12], evaluating the ability to transform natural language requirements to post-conditions [11], generating specifications [37], user story generation [29], user story quality evaluation [39], and multiple AI agents handling requirement-related tasks [32]. Despite its capabilities, LLMs also exhibit notable limitations, such as generating factually incorrect information (hallucinations) [38] and potentially impeding one's cognitive or creative processes [18]. These challenges must be carefully monitored in human–AI collaborative contexts before LLMs are fully integrated into operational workflows.

The central research question guiding this study is: ***How can generative AI be leveraged to automate user story creation and incorporate sustainability considerations into agile requirements engineering workflows?***. To explore this question, we developed and evaluated *Reqwire*, a sustainability-aware, AI-driven requirements engineering (RE) workflow designed to automate the generation of structured user stories derived from product specification documents, estimate story points, and annotate them with sustainability-related tags. Reqwire also integrates seamlessly with the Jira project management environment.

Reqwire was piloted and evaluated in collaboration with a leading telecommunications and digital services enterprise in Finland, which served as the industrial partner. Within this organization, product managers had previously encountered challenges stemming from ambiguous or incomplete requirements, resulting in miscommunication, rework, prolonged sprint cycles, and misalignment with strategic objectives. Furthermore, the company demonstrated a growing interest in embedding sustainability as a first-class concern during early requirements elicitation, aligning with its broader corporate sustainability strategy. The novelty of this study lies in the automation of user story generation and the integration of sustainability considerations by automatically tagging user stories according to their corresponding sustainability dimensions: environmental (e.g., energy consumption, CO_2 emissions), social (e.g., accessibility), technical (e.g., usability, maintainability), and economic. This approach positions sustainability as a core development objective alongside traditional software engineering goals.

The rest of the paper is structured as follows: Sect. 2 reviews related work, while Sect. 3 outlines the research methodology and design of the AI-driven approach. Section 4 details the system implementation across several cycles of design iterations, while Sect. 5 discusses the findings and limitations. Finally, Sect. 6 concludes the study and suggests avenues for future research.

2 Related Work

The core aim of RE is to systematically gather, define, and document requirements from stakeholders in a clear and structured manner, establishing a foundation for reliable and user-centered software systems [14]. Inam [15] characterizes a well-specified requirement as a software feature that accurately reflects client needs, emphasizing the necessity for precision at this stage. A study examining 32 software projects conducted between 2003 and 2005 within a large enterprise development division in Tokyo investigated the relationship between requirements specification quality and project success [17], where the researchers observed that balanced and comprehensive Software Requirement Specification (SRS) descriptions are characteristic of successful projects. While improperly engineered requirements are not the only reason for project failures, proper requirements contribute substantially to the overall success of software projects [15]. Furthermore, minimal documentation and neglect of non-functional requirements undermine traceability, contribute to budget overruns, and reduce system usability, as demonstrated in a systematic literature review of 21 studies [16].

Sustainability, also as a non-functional requirement, can play an important role in incorporating broader perspectives and strategic goals to the project [30]. The Karlskrona Manifesto advocates and provides guidelines for integrating sustainability principles into software system requirements, design, and development [3]. Complementing this, Lago et al. introduced a sustainability analysis framework that can assist professionals in evaluating software qualities across environmental, social, technical, and economic dimensions, including their interdependencies [20] as well as, Penzenstadler et al. introduced a generic sustainability model that integrate both process and product-specific instances to help

requirements engineers assess their work across multiple dimensions and identify areas for improvement [27]. Extending this line of research to more RE focus, requirements have been identified as the driving force behind sustainability [4] and as a crucial non-functional requirement of the 21st century [28]. The underlying rationale is that software developers can better understand and manage the sustainability impact of the systems they design through properly elicited sustainability-aware requirements. Therefore, the role of requirement elicitation with the mindset of sustainability is crucial for developing better software systems for everyone on the planet.

Early research studies also explored the application of Natural Language Processing (NLP) and Machine Learning (ML) to RE processes, including the classification of functional and non-functional requirements [6,19], assessment of requirements quality [26], and management of requirements by identifying requirement changes [22]. However, the process was labor-intensive, language-specific, and brittle, unable to generalize across domains or capture true semantics. Since the introduction of publicly accessible generative AI models, such as ChatGPT in 2022[1], several generative AI models have enhanced language processing capabilities in software engineering [1], ranging from generating [29] and refining user stories [31] to producing structured requirement documentation [1]. More recent RE-related developments include LLM-powered agentic systems [32], and different prompt engineering techniques like, Refine and Thought (RAT) [29]. As Generative AI and RE continue to advance rapidly, we identify the need for exploring the capability of defining sustainable aspects, encompassing environmental, social, technical, individual, and economic dimensions of requirements during the elicitation stage.

3 Research Methodology

This section details the research design and methodology used to develop and evaluate Reqwire. Our approach is fundamentally rooted in the Design Science Research (DSR) paradigm, which systematically addresses real-world problems through the creation and investigation of innovative artifacts [13]. In Information Systems (IS) and Software Engineering (SE), DSR comprises two primary activities: designing the artifact and subsequently investigating it within its practical context. This context encompasses a wide array of elements, including stakeholders and their objectives, existing scientific and engineering theories, current designs and products, and the accumulated experience and common sense of researchers. Specifically, we employed Hevner's three-cycle view of DSR [13], which emphasizes the interconnected and iterative nature of DSR activities. The cycles are listed below.

– **The Relevance Cycle**grounds DSR in real-world contexts by identifying domain problems and opportunities, setting acceptance criteria, and field testing artifacts, with outcomes guiding further iterations [5]. In Reqwire,

[1] https://openai.com/index/chatgpt/.

this involved addressing inefficiencies in manual agile artifact creation and enabling sustainability tagging, validated by our industry partner. The evaluation of Reqwire considered task completion rates, number of corrections required, quality of user stories, accuracy of sustainability tagging, and the correlation of story point estimation with expert reference points.

- **The Rigor Cycle** ensures scientific validity by grounding design in established theories, methods, and artifacts, while contributing new knowledge through innovation and evaluation [5]. In our case, this involved leveraging existing knowledge in RE, AI, and sustainability frameworks to inform the design of Reqwire.

- **The Design Cycle** is the core iterative component of DSR, involving repeated construction and evaluation of artifacts until requirements are met [34]. It draws requirements from the Relevance Cycle and theories and methods from the Rigor Cycle, balancing both in iterative testing before field deployment. For Reqwire, this meant prototyping, incorporating stakeholder feedback, and refining features such as sustainability tags and Jira integration.

3.1 Participants

We engaged two main types of participants in the various cycles of our research.

- **IT Practitioners:** Our primary industry partner was a leading telecommunications and digital services provider, which contributed through the Head of Service Development and 12 Developers. They were central to the Relevance Cycle, engaging in workshops and feedback loops to identify and refine problem scope, emphasize key needs (like sustainability tagging and Jira integration), and provide continuous feedback on preliminary designs. They also played a key role in the evaluation and empirical validation, offering insights on integration depth, usability, and tagging accuracy during the Minimum Viable Product (MVP) demonstrations. Data collected included insights from collaborative workshops, feedback from bi-weekly sessions, and observations from client-focused seminars, all contributing to the iterative refinement of Reqwire.

- **Students and Academic Researchers:** From academia, 22 software engineering master's students with industry experience and 4 researchers in a Green ICT and sustainability-focused software engineering program contributed primarily during the second validation phase. They provided peer evaluation on the robustness of the AI pipeline, the novelty of integrating sustainability metadata into user stories, and the system's modularity. Data gathered from seminars and technical reviews offered qualitative feedback that informed assessments of Reqwire's scientific contribution and its potential for future research.

Two workshops, each lasting 4–5 h, were held with industry stakeholders at the client's company premises to test and provide feedback, followed by a final seminar intended for both industry and academic participants.

4 Implementation of Reqwire

Reqwire was developed using DSR to address the challenge of anticipating sustainability effects in IT products and services. This section discusses the DSR process across the relevance, rigor, and design cycles introduced in Sect. 3. Figure 1 provides an overview of all cycles. Each cycle contributed to designing the Reqwire prototype and verifying relevance in practice.

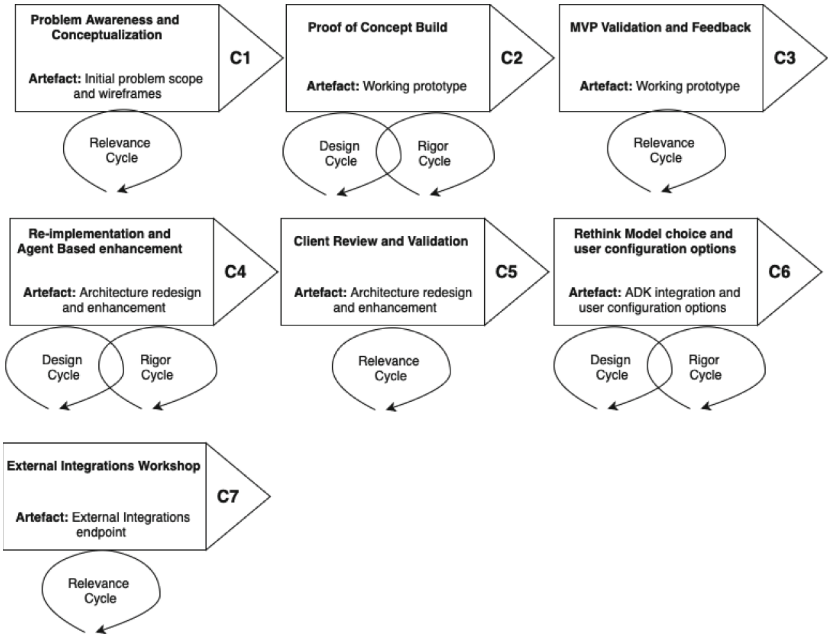

Fig. 1. Seven cycles of design and evaluation of Reqwire.

4.1 [Relevance] Cycle 1 Problem Awareness and Conceptualization

This cycle focused on the problem space. The industry partner highlighted inefficiencies in the Software Development Life Cycle (SDLC), particularly in the manual requirements elicitation process and the poor integration of AI within the SDLC. Discussion among the industry stakeholders and the project team refined the scope to a key challenge to address: the lack of useful AI tool support for automating the generation of actionable user stories in the RE phase of

agile software development. The industry stakeholders emphasized three critical requirements for adoption of such an AI tool: (i) the ability to generate user stories from varying depths of specifications, (ii) attach sustainability metadata to the user stories, and (iii) seamlessly integrate with agile project management tools.

Participants: Industry side stakeholders, academic researchers, and the project research team.

Lessons Learned: This cycle revealed that while AI tools support many SDLC phases, the RE stage, especially the formulation of clear and actionable user stories, remains underserved in practice. The industry partner emphasized this as a critical bottleneck that affects sprint velocity, project budget, developer clarity, and sustainability tracking.

4.2 [Design and Rigor] Cycle 2 Proof-of-Concept (POC) Build

Designed Artifacts: Early requirements, architecture, and an initial working prototype. Based on Cycle 1, sets of preliminary requirements were identified that covered key capabilities, and these were visualized using low-fidelity wireframes and interaction sketches that illustrated the envisioned workflow.

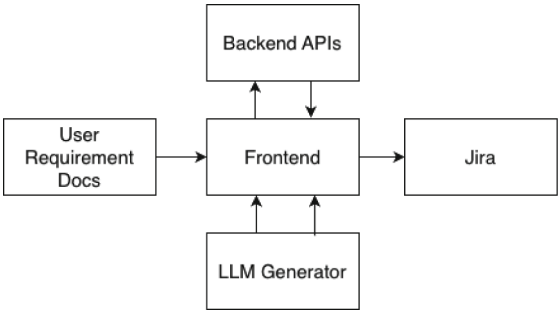

Fig. 2. First working prototype blackboard architecture with five components.

In this cycle, a preliminary architecture was designed as shown in Fig. 2; a simple blackboard architecture with five components. The Frontend serves as a central blackboard, integrating updates from Backend APIs, user inputs, and knowledge sources like the LLM Generator and Jira, enabling independent components to exchange data and coordinate effectively. The first working prototype comprised the following three functionalities:

1. **Local Project Management:** Reqwire enables users to initiate and manage projects within a local workspace. Projects are organized in a modular structure and can also be linked with Jira for centralized tracking.

2. **Intelligent User Story Generation:** The central feature of the platform is its AI-powered user story generation engine (Fig. 3). These are the details of the prompt design, the input, and output of the system.
   (a) **Prompt Structure:** The prompt defines an *Agentic Requirements Engineer* composed of cooperating sub-agents that simulate distinct stages of a human requirements process: *Planner*, *Generator*, *Sustainability Analyst*, *Reviewer*, and *Finalizer*. Each sub-agent operates under explicitly defined roles, performing specialized reasoning tasks that contribute to a coherent, traceable workflow.

      i. **Input Specification:** The input to the system is natural-language software requirements expressed as free text. Each requirement describes a desired function, constraint, or condition of a system. The model interprets this text as the source for generating structured user stories.
      ii. **Rules and Workflow:** The workflow consists of five sequential roles. The *Planner* defines internal subtasks based on the requirement, but does not output them. The *Generator* converts each subtask into candidate user stories following a strict template containing title, description, acceptance criteria, priority, and story points. The *Sustainability Analyst* evaluates each story across five predefined dimensions: environmental, economic, social, individual, and technical, based on the Sustainability Awareness Framework (SusAF) [5]. Impacts are assigned a polarity (`+`, `-`, or `+/-`) and accompanied by short, evidence-based justifications. The *Reviewer* enforces quality through deterministic checks on structure, testability, and completeness, allowing one regeneration attempt for any rejected story. Finally, the *Finalizer* compiles all accepted stories into a validated output array.
      iii. **Output Design:** The model produces a strictly formatted JSON array containing the generated set of user stories, which follow the widely adopted user story template popularized by Mike Cohn [9]: "As a [user], I want to [action], so that [value]." Each story is represented as an object with the following fields: *title*, *description*, *acceptance criteria*, *priority*, *story points*, and *sustainability* (consists of dimension, polarity, and reasoning). The output is required to be syntactically valid JSON, UTF-8 encoded, and free from explanatory or intermediate text.

   (b) **Measures to Minimize Hallucination:** The prompt incorporates several safeguards to reduce hallucination and maintain factual grounding. First, strict output constraints prohibit additional commentary or unsupported reasoning traces. Second, sustainability tagging is explicitly evidence-based: dimensions are omitted if cues are not clearly present in

the requirement or story context. Third, the *Reviewer* role enforces concrete rejection criteria, such as detecting vague acceptance conditions or incomplete story structure. Fourth, the prompt instructs the model to prefer omission when confidence is low, preventing speculative or inconsistent content. Finally, few-shot examples are provided to anchor the model's responses to consistent patterns of style, detail, and sustainability reasoning.

3. **Sustainability Tagging**: As mentioned in the previous Subsect. 2, *Sustainability Analyst* analyzes each user story across the five sustainability dimensions using an explicit, rule-based reasoning sequence: defining context, mapping relevant dimensions, and assessing polarity (`+`, `-`, or `+/-`). These results are then embedded into the final user story as structured sustainability tags. For example: environmental, + (feature reduces server load), - (requires frequent heavy computation); economic, + (reduces maintenance cost), - (increases cost); technical, + (reusable APIs), - (hard-coded dependencies); social, + (improves accessibility), - (reduces accessibility); individual, + (supports privacy), - (invasive data collection). From the sample user story generated by Reqwire, shown in Fig. 4, on the real-time location tracking of the delivery partner, the user story has a positive technical impact on system's technical quality but introduces a negative environmental impact due to increased energy and resource consumption. Accordingly, Reqwire labels this story with a Technical (+) and Environmental (–) tag. This feature allows teams to gain early insight into the sustainability impacts of their requirements and enables them to make informed trade-offs and adjustments.

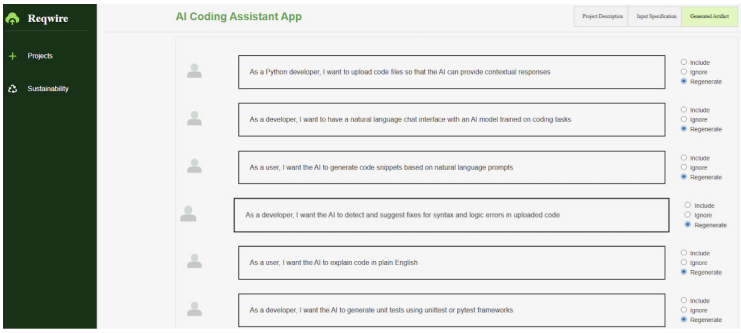

Fig. 3. Interface 2: Generated user stories page.

Participants: Industry client, academic researchers, and the project researchers team.

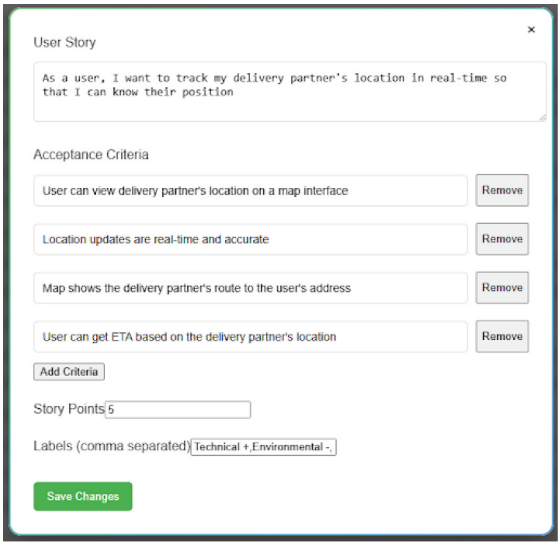

Fig. 4. Interface 3: A generated user story with assigned story points and sustainability label.

Lessons Learned: Building a functional vertical slice early exposes integration and accuracy issues, explainability for sustainability tags is essential for trust, and error handling is required for smooth tool adoption.

4.3 [Relevance] Cycle 3 MVP Validation and Feedback

Artifacts to Be Validated: The MVP developed in Cycle 2 was evaluated for usability, alignment with agile workflows, accuracy of generated stories and tags, and overall feasibility in addressing the identified RE gaps.
Participants: Industry-side software product development team, software engineering master's student with industry experience, and the project research team (facilitators).
Data Collected: Interaction notes from the on-site workshop and quantitative metrics such as task completion times for generating and exporting user stories, and feedback from the instructor and peer engineering students.
Evaluation: The MVP was presented and tested during an on-site seminar with the client's product teams and also tested with 22 Software engineering master's students with industry experience at the university. Hands-on interaction with the tool included uploading sample requirements, generating user stories, applying sustainability tags, and exporting to Jira. Feedback was gathered through open discussions, with key results highlighting high satisfaction with the generated stories, the potential time savings for requirements engineers provided by the solution, and the intuitive flow of the application. However, there was a concern regarding the linearity of the workflow. The industry product development

team noted that in most real-world scenarios, the flow from RE to project management (PM) tools is often non-linear, so the workflow needs to be modified to allow regeneration of artifacts.

Additional concerns were raised about tag accuracy, for example, occasional misclassifications of environmental impacts, which heavily depend on the level of detail in the specifications. Since Reqwire generates user stories from specifications of varying levels and depth, it can lack sufficient detail to accurately match features to sustainability tags. This becomes an issue when the generated tags are fully trusted without verification. Furthermore, the client expressed a desire for the user stories, sustainability tags, and estimated story points to be made available for integration with sustainability evaluation and visualization tools, to enable sustainability efforts tracking from the user story level.

Lessons Learned: To better align with real-world practices, the workflow should be redesigned to support iterative artifact regeneration and non-linear progression. Enhancing the system's ability to handle varying levels of specification detail will improve tag accuracy, potentially through more advanced natural language processing or user-guided refinements. Addressing these issues will increase the tool's practicality and trustworthiness in diverse project contexts.

4.4 [Design and Rigor] Cycle 4 Re-implementation and Agent-Based Enhancements

Designed Artifacts: Drawing from the feedback in Cycle 3, this cycle focused on re-implementation to address key limitations, particularly the need for non-linear workflows, improved tag accuracy, and greater flexibility in handling diverse specification inputs. A major redesign involved shifting from the initial blackboard architecture to a hybrid agent-based and layered architecture (Fig. 5), which introduced multi-agent coordination to enable conversational interactions, dynamic task delegation, and improved interoperability. The new architecture comprises four layers:

1. **Presentation Layer**: A web-based GUI for uploading requirements and managing projects, plus a new agent chat interface for natural language interactions (e.g., users can conversationally refine stories or issue commands like *"regenerate this story with more environmental context"*).
2. **Agent Coordination Layer**: A Root Agent serves as the central coordinator, interpreting user intents via the Agent Control Protocol (ACP) and delegates tasks. A Distributor Agent handles exports and communications with external systems using the Agent-to-Agent (A2A) protocol.
3. **Tool and Processing Layer**: Houses the User Story Generator Agent (powered by Gemini 1.5 Pro) with structured prompting for bias mitigation and context awareness, backend RESTful APIs for operations such as regeneration, and modular task tools (e.g., Create Jira Project, Export to Jira).
4. **External Integration Layer**: Connections to third-party services like Jira via authenticated REST APIs, with an External Agent placeholder for future expansions.

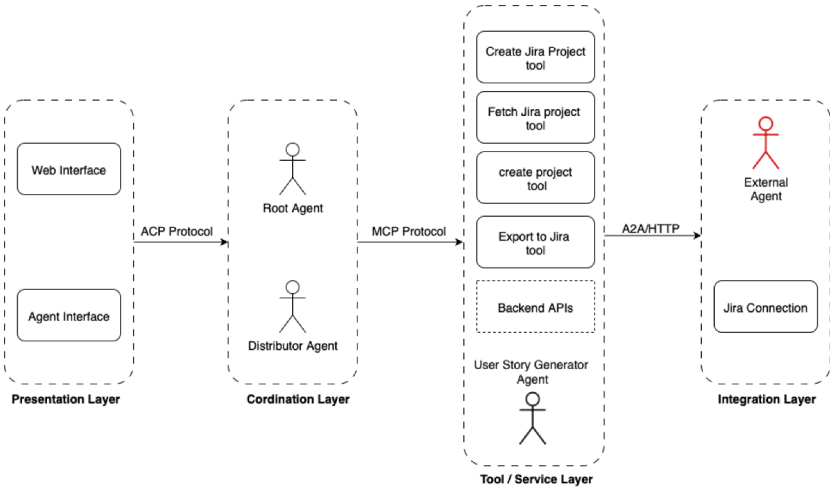

Fig. 5. Hybrid agent-oriented and layered architecture.

This agent-based design supports iterative, non-linear workflows by allowing agents to handle complex instructions adaptively, reducing hallucinations through guardrails and validation loops. Accessibility was enhanced, and human-in-the-loop controls were strengthened, enabling users to override tags or provide additional context during regeneration.

Participants: Project research team, industry side development team, and solutions architects for iterative reviews.

Evaluation: Internal testing assessed the new architecture's performance, including workflow flexibility (e.g., regeneration success rates and ease), tag accuracy, and latency. Bi-weekly feedback sessions with the industry stakeholders helped validate enhancements.

Lessons Learned: Agent-oriented designs significantly boost automation and adaptability in RE tools, but require robust protocols (e.g., ACP, A2A) to prevent coordination failures. Addressing AI biases and hallucinations demands layered safeguards like custom prompts and validation, while emission optimizations must be balanced with functionality to maintain sustainability goals.

4.5 [Relevance] Cycle 5 Client Review and Validation

Artifacts to Be Validated: The refined MVP, now incorporating the agent-based architecture, enhanced sustainability tagging, dynamic regeneration, and Jira integration, was evaluated for overall robustness, novelty, and enterprise readiness. Key focus areas included the accuracy of AI-generated artifacts, interoperability potential, and contribution to sustainable RE practices.

Participants: Academic researchers, software engineering students, industry-side team, and the project research team.

Data Collected: Seminar notes and qualitative insights from discussions on modularity and future extensibility.
Evaluation: The updated artifact was presented and tested in a technical seminar to client stakeholders and in class to the academic researchers and peer software engineering master's students with industry experience. Attendees interacted with the tool, testing features such as conversational refinements via the agent chat, multi-dimensional sustainability labeling, and Jira exports. Feedback was highly positive on the integration of sustainability metadata into user stories and the agent architecture for modularity and potential for broader tool interoperability. A key suggestion from this cycle was to address the challenge of integrating the newer AI models into the system with minimal changes. Therefore, the system should incorporate a mechanism to seamlessly adapt to and integrate the latest model of choice with minimal or no modifications to the overall architecture.
Lessons Learned: The rapid evolution of generative AI models emphasizes the need for flexible, model-agnostic designs that facilitate seamless upgrades with minimal architectural changes, ensuring long-term adaptability and relevance in the landscape of applied generative AI.

4.6 [Design and Rigor] Cycle 6 Rethink Model Choice and User Configuration Option

Artifact Designed and to Be Validated: Drawing from the feedback in Cycle 5, the agent was re-implemented using the Google Agent Development Toolkit[2], which provides out-of-the-box tools for implementing agents. Gemini 1.5 was selected as the base model, but it is swappable via the configuration interface without code or architectural modifications. The configuration allows to use of both commercial and self-hosted models through configuration fields on the application GUI. The input from this configuration is used to set the model parameter of the agent.
Participants: Project research team, industry-side development team.
Lessons Learned: Adopting a standard Agent Development Kit (ADK), such as the Google Agent Development Toolkit, enabled seamless use of the latest advancements in generative AI. It also allowed focus on the specific use case, rather than foundational implementation details, which are provided out-of-the-box. This enables some form of confidence in the future-proof nature of the solution.

4.7 [Relevance] Cycle 7 External Integrations Workshop

Artifacts to Be Validated: A workshop to test Reqwire and its endpoints for integrating with other systems that evaluate and visualize sustainability efforts across projects.

[2] https://google.github.io/adk-docs/.

Participants: Industry-side stakeholders, the project's research team, and teams working on sustainability tracking, estimation, and visualization.
Data Collected: Workshop session notes and cross-team discussion notes, participant feedback, system metrics such as API response times, and error rates during data exchanges.
Evaluation: Participants connected Reqwire's endpoints (e.g., via REST APIs and A2A protocols) to existing sustainability tools, such as internal dashboards for sustainability efforts estimation and visualization platforms (for example, custom green metrics trackers[3]). Simulated workflows included generating user stories, exporting tagged data to Jira, and pulling sustainability metadata into visualization tools for aggregate reporting across projects. Feedback indicated strong performance in data interoperability, with visualizations effectively highlighting trends like negative environmental impacts from high-energy features.
Lessons Learned: External integrations are crucial for enterprise adoption, as they extend Reqwire's value beyond isolated RE tasks to holistic sustainability management in software development workflows. Workshops expose practical scalability issues early, emphasizing the need for secure and optimized APIs with standardized data formats.

5 Discussion

The findings of this study provide evidence that generative AI can enhance agile requirements engineering by addressing two enduring challenges: the inefficiency of manual user story creation and the lack of structured sustainability integration during requirements generation in the early stage of software development. The results show that Reqwire makes sustainability trade-offs visible and actionable within agile workflows. Reqwire's automated user story generation and sustainability tagging feature (Fig. 3 and 4) to the five sustainability dimensions reduces the manual task with problems of inconsistency, incompleteness, errors in user story generation [10] and manual efforts for requirement engineers interested in sustainability during requirement gathering. By embedding these sustainability dimensions at the user story level, Reqwire operationalizes the sustainability principles proposed in earlier sustainability design frameworks [3,20,27] by bringing sustainability concerns closer to requirement elicitation.

Recent studies on generative AI in RE have primarily emphasized productivity gains and automation capabilities [1,32], our study highlights the added value of embedding sustainability tagging of requirements (in the form of user stories) during the requirement elicitation. This position requires, as an extension of prior AI-based RE tools, offering not only improved efficiency of user story generation but also advocating sustainability as a non-functional requirement in software development. Furthermore, the growing body of work on AI in RE has primarily focused on requirement generation and analysis [2,23]. However, few studies [32] have explored multi-agent architectures that are capable

[3] https://greenmetrics.co/.

of dynamic regeneration and continuous improvement of requirement artifacts. Reqwire's design, which allows integration with AI models like Gemini and open-source LLMs, contributes to this emerging paradigm by enabling adaptability and model-agnostic scalability.

The iterative development process was essential for aligning Reqwire's capabilities with real-world agile practices. Nevertheless, challenges remain that sustainability tagging accuracy depends on input detail, and LLMs carry limitations as discussed in Sect. 2. We also identify the need for wider testing of Reqwire, which will be conducted in a future client project.

6 Conclusion and Future Work

This study presents Reqwire, a generative AI–driven, sustainability-aware Requirements Engineering (RE) tool that automates user story creation and integrates sustainability tagging at the user story level, addressing a gap in recent RE and sustainability studies. Future work will include expanding the range of input modalities (diagrams, meeting transcripts, etc.), refining agent-to-agent protocols, and mitigating the model's limitations, like hallucinations (using techniques like Retrieval-augmented generation (RAG)). Most importantly, we plan to conduct wider testing in industry projects to validate Reqwire's effectiveness and adoption potential in real-world settings.

Declaration on Generative AI. During the preparation of this work, the author(s) utilized Generative AI tools to ensure grammatical correctness and improve sentence structure. Following the use of these tools, the authors thoroughly reviewed the content to the best of their knowledge.

Acknowledgment. This work has been supported by FAST, the Finnish Software Engineering Doctoral Research Network, funded by the Ministry of Education and Culture, Finland. And also, the SE4GD - Software Engineers for Green Deal program under Grant 619839 from the Erasmus Mundus Instrument of the European Union.

References

1. Arora, C., Grundy, J., Abdelrazek, M.: Advancing requirements engineering through generative AI: assessing the role of LLMs. In: Nguyen-Duc, A., Abrahamsson, P., Khomh, F. (eds.) Generative AI for Effective Software Development. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-55642-5_6
2. Barzamini, H., Nazaritiji, F., Brockmann, A., Ferdowsi, H., Rahimi, M.: An AI-driven requirements engineering framework tailored for evaluating AI-based software. In: 2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN), pp. 138–149. IEEE (2025)

3. Becker, C., et al.: Sustainability design and software: the Karlskrona Manifesto. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, pp. 467–476. IEEE (2015)
4. Becker, C., et al.: Requirements: the key to sustainability. IEEE Softw. **33**(1), 56–65 (2016). https://doi.org/10.1109/MS.2015.158
5. Betz, S., et al.: Lessons learned from developing a sustainability awareness framework for software engineering using design science. ACM Trans. Softw. Eng. Methodol. **33**(5), 1–39 (2024)
6. Binkhonain, M., Zhao, L.: A review of machine learning algorithms for identification and classification of non-functional requirements. Exp. Syst. Appl. X **1**, 100001 (2019)
7. Brockenbrough, A., Feild, H., Salinas, D.: Exploring LLMs impact on student-created user stories and acceptance testing in software development. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education, SIGCSETS 2025, vol. 2, pp. 1401–1402. ACM, New York (2025)
8. Chitchyan, R., et al.: Sustainability design in requirements engineering: state of practice. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 533–542 (2016)
9. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional (2004)
10. Dos Santos, C.A., Bouchard, K., Minetto Napoleão, B.: Automatic user story generation: a comprehensive systematic literature review. Int. J. Data Sci. Anal. **20**(1), 1–24 (2025)
11. Endres, M., Fakhoury, S., Chakraborty, S., Lahiri, S.K.: Can large language models transform natural language intent into formal method postconditions? Proc. ACM Softw. Eng. **1**(FSE) (2024). https://doi.org/10.1145/3660791
12. Gorer, B., Aydemir, F.B.: GPT-powered elicitation interview script generator for requirements engineering training. arXiv arXiv:2406.11439 [cs], June 2024. https://doi.org/10.48550/arXiv.2406.11439
13. Hevner, A.: A three cycle view of design science research. Scand. J. Inf. Syst. **19**(2), 87–92 (2007)
14. Hussain, A., Mkpojiogu, E., Kamal, F.: The role of requirements in the success or failure of software projects. Int. Rev. Manag. Mark. **6**, 306–311 (2016)
15. Inam, A.: A Study of requirements engineering practices among software developers at uum information technology. Master's thesis, Universiti Utara Malaysia, Malaysia (2015)
16. Inayat, I., Salim, S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51B**, 915–929 (2015)
17. Kamata, M., Tamai, T.: How does requirements quality relate to project success or failure? In: 15th IEEE International Requirements Engineering Conference, RE 2007, pp. 69–78 (2007)
18. Kosmyna, N., et al.: Your brain on ChatGPT: accumulation of cognitive debt when using an ai assistant for essay writing task. arXiv preprint arXiv:2506.08872 (2025)
19. Kurtanović, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 490–495. IEEE (2017)
20. Lago, P., Koçak, S., Crnkovic, I., Penzenstadler, B.: Framing sustainability as a property of software quality. Commun. ACM **58**(10), 70–78 (2015)

21. Laplante, P.A., Kassab, M.: Requirements Engineering for Software and Systems, 4th edn. Auerbach Publications (2022)
22. Lyutov, A., Uygun, Y., Hütt, M.: Managing workflow of customer requirements using machine learning. Comput. Ind. **109**, 215–225 (2019)
23. Myllynen, S., Suominen, I., Raunio, T., Karell, R., Lahtinen, J.: Developing and implementing artificial intelligence-based classifier for requirements engineering. J. Nucl. Eng. Radiat. Sci. **7**(4), 041201 (2021)
24. Oyedeji, S., Naqvi, B., Penzenstadler, B., Adisa, M.O., Abdulkareem, M., Seffah, A.: The interplay between usability, sustainability and green aspects: a design case study from a developing country. In: ICT4S (2019)
25. Oyedeji, S., Shamshiri, H., Adisa, M.O., Capilla, R., Chitchyan, R.: Integrating sustainability into scrum agile software development: an action research approach. In: Papatheocharous, E., Farshidi, S., Jansen, S., Hyrynsalmi, S. (eds) Software Business, ICSOB 2024. LNBIP, vol. 539, pp. 236–250. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-85849-9_20
26. Parra, E., et al.: A methodology for the classification of quality of requirements using machine learning techniques. Inf. Softw. Technol. **67**, 180–195 (2015)
27. Penzenstadler, B., Femmer, H.: A generic model for sustainability with process- and product-specific instances. In: Proceedings of the 2013 Workshop on Green in/by Software Engineering, GIBSE '13, pp. 3–8. ACM, New York (2013)
28. Penzenstadler, B., Raturi, A., Richardson, D., Tomlinson, B.: Safety, security, now sustainability: the nonfunctional requirement for the 21st century. IEEE Softw. **31**(3), 40–47 (2014)
29. Rahman, T., Zhu, Y.: Automated user story generation with test case specification using large language model. arXiv preprint arXiv:2404.01558 (2024). https://doi.org/10.48550/arXiv.2404.01558
30. Raturi, A., Penzenstadler, B., Tomlinson, B., Richardson, D.: Developing a sustainability non-functional requirements framework. In: Proceedings of the 3rd International Workshop on Green and Sustainable Software, pp. 1–8 (2014)
31. Ronanki, K., Cabrero-Daniel, B., Berger, C.: ChatGPT as a tool for user story quality evaluation. In: Kruchten, P., Gregory, P. (eds.) Agile Processes in Software Engineering and Extreme Programming – Workshops, XP XP 2022 2023. LNBIP, vol. 489. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-48550-3_17
32. Sami, M., Waseem, M., Zhang, Z., Rasheed, Z., Systä, K., Abrahamsson, P.: AI based multiagent approach for requirements elicitation and analysis. arXiv preprint arXiv:2409.00038 (2024). https://doi.org/10.48550/arXiv.2409.00038
33. dos Santos, C.: Leveraging text generation for enhanced user story quality. Ph.D. thesis, Université du Québec à Chicoutimi, Chicoutimi (2025). https://constellation.uqac.ca/id/eprint/10176/
34. Simon, H.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge, MA (1996)
35. Swarnalatha, K., Srinivasan, G., Dravid, N., Kasera, R., Sharma, K.: A survey on software requirements engineering for real time projects based on customer requirements. Int. J. Adv. Res. Comput. Commun. Eng. **3**(1), 5045–5050 (2014)
36. Tikayat Ray, A., Cole, B.F., Pinon Fischer, O.J., Bhat, A.P., White, R.T., Mavris, D.N.: Agile methodology for the standardization of engineering requirements using large language models. Systems **11**(7), 352 (2023). https://doi.org/10.3390/systems11070352
37. Xie, D., et al.: How effective are large language models in generating software specifications? pp. 1–12 (2025). https://doi.org/10.1109/SANER64311.2025.00014

38. Xu, Z., Jain, S., Kankanhalli, M.: Hallucination is inevitable: an innate limitation of large language models. arXiv preprint arXiv:2401.11817 (2024)
39. Zhang, Z., Rayhan, M., Herda, T., Goisauf, M., Abrahamsson, P.: LLM-based agents for automating the enhancement of user story quality: an early report. In: Šmite, D., Guerra, E., Wang, X., Marchesi, M., Gregory, P. (eds.) XP 2024. LNBIP, pp. 117–126. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-61154-4_8