MATHEMATICAL MODELING AND COMPUTING, Vol. 12, No. 2, pp. 628–639 (2025)

Analysis of bottleneck points based on the software requirements data-flow model in Agile projects

Kruk R. A., Zhukovska N. A.

The National University of Water and Environmental Engineering, NUWEE, 11 Soborna Str., 33028, Rivne, Ukraine

(Received 5 February 2025; Revised 17 June 2025; Accepted 19 June 2025)

This paper examines challenges in software requirements management in Agile projects, focusing on critical workflow points affecting requirements quality. Key issues include incompleteness, inconsistency, lack of traceability, and poor structuring, exacerbated by Agile's iterative nature. A comparative analysis of tools like Jira and Confluence highlights their limitations in ensuring requirements quality. Using a stock-and-flow model, the study conceptualizes the requirements management system as an interconnected process, revealing feedback loops that reduce efficiency. Findings indicate that optimizing software specification processes with a complementary system can mitigate these challenges while minimizing human intervention. The research lays the foundation for developing a mathematical model to improve Agile requirements management efficiency.

Keywords: *Agile; requirements engineering; software requirements; SRS quality attributes; system analysis.*

2010 MSC: 68Q85, 68N01, 68N30, 90B50, 65K10 **DOI:** 10.23939/mmc2025.02.628

1. Introduction

The Agile software development methodology, while the most effective in the realities of today's dynamic market, also creates unique challenges in the field of software requirements engineering and management [1]. A survey of the Ukrainian IT sector, conducted during a preliminary phase of this research, confirmed the presence of the phenomenon of software requirements issues in Agile projects, which tend to emerge over time (typically over several years). Furthermore, it was found that, statistically, software requirements issues are characteristic of every second Agile project, based on testimonials from employees in the Ukrainian IT industry. Additionally, it was revealed that the phenomenon of software requirements issues is one of the key factors leading to the failure to meet project commitments, i.e., obligations to the project's customer (state or private business).

A more detailed analysis showed that the essence of the phenomenon of software requirements issues lies in their non-compliance with quality requirements for software specifications due to insufficient effort in creating or maintaining the specification. Among the most critical quality requirements for software specifications, which influence the emergence of subsequent issues, the following were identified:

— Omitted or incomplete software requirements;
— Inconsistencies in software requirements;
— Lack of traceability of software requirements and the inability to identify dependencies between them;
— Low level of software requirements structuring and insufficient completeness of the product specification.

All of the above challenges are caused by the characteristics of the Agile methodology, in particular, its iterative approach to processing software requirements. In traditional development methodologies, such as the waterfall model, the software requirements specification is fully formed before any engineering work begins [2], allowing these challenges to be avoided. However, such a model is ineffective

© 2025 Lviv Polytechnic National University

in the modern market [3, 4], except for projects requiring a high level of completeness (e.g., software for construction, transportation, healthcare, etc.).

Thus, at the current stage of the research, the goal was to analyze the software requirements management systems in Agile projects to identify:

1. The degree to which software requirements management systems comply with the quality assurance requirements for software specifications;
2. Critical points in the data flows of software requirements management systems;
3. A formalized multiprocess ecosystem of an Agile project into a task with defined influencing factors suitable for mathematical and logical modeling;
4. A sequence of steps for developing a mathematical model of a solution that would minimize or completely eliminate the identified critical points in software requirements management systems in Agile projects.

2. Background and related works

This chapter explores prior research in the domain of software requirements management in Agile projects, highlighting the critical gaps identified in current practices and tools. Building on system modeling and process flow frameworks, it further integrates insights from comparative analyses and systematic literature reviews to contextualize the evolution of software requirements management practices in Agile ecosystems. This exploration underscores the necessity of innovative methodologies and models to overcome identified deficiencies, paving the way for the proposed solutions discussed in subsequent sections.

Exclusion and inclusion criteria for the related work research

Inclusion criteria:

1. Focus on Agile Methodologies: Only works explicitly addressing Agile practices or their integration into requirements management were included, as Agile is the central framework of the research.
2. Relevance to Requirements Management Systems: Papers that explore methods, tools, or challenges related to requirements management in software projects were prioritized.
3. Emphasis on Workflow Improvement: Studies that propose enhancements to workflows, including modeling, elicitation, validation, or traceability, were included.
4. Coverage of Critical Quality Issues: Research addressing software requirements quality, such as handling ambiguity, prioritization, or traceability, was included to align with the research focus.
5. Peer-Reviewed Works: Priority was given to peer-reviewed articles from the past two decades, ensuring the relevance and credibility of findings.

Exclusion criteria:

1. Irrelevant Development Methodologies: Papers focusing solely on non-Agile or waterfall models without discussing Agile practices were excluded.
2. Limited Focus on Requirements Management: Research that does not address workflows or systems specific to requirements management was excluded.
3. Overly Theoretical Studies: Articles focusing purely on theoretical concepts without practical or methodological insights were excluded.
4. Duplicate or Redundant Content: Papers presenting similar findings without new contributions to the field were excluded.

Methods and tools of the related work research. To identify and analyze relevant literature, the authors employed a structured methodology leveraging both traditional and AI-driven tools. The process included the following steps and resources:

— **AI-Assisted Literature Search:** The authors utilized **Google Scholar** as the primary database to locate academic papers and articles relevant to the research topic. To enhance the efficiency and comprehensiveness of the search, **Elicit.com** was employed to perform AI-based keyword searches.

This tool allowed the identification of articles that aligned closely with the research objectives and provided insights into the most relevant works.

— **Mapping and Clustering Research:** To understand the connections and trends within the body of literature, the authors used **Litmaps.com**. This tool enabled the visualization of citation relationships and the clustering of related works. By mapping these connections, the authors could identify seminal papers, trends, and gaps in the research field. This process also facilitated the differentiation between original research and secondary sources.

— **Keyword and Topic Selection:** The authors carefully curated keywords and topics based on the themes of software requirements quality, Agile methodology, and software engineering tools. These keywords guided both the AI-based searches and manual reviews to ensure the inclusion of highly relevant and diverse perspectives.

— **AI-Assisted Analysis of Literature:** To streamline the review process and focus on the most relevant studies, the authors employed **GPT-4o** to assist in the initial analysis of the 59 accumulated papers. By leveraging its natural language processing capabilities, the model generated concise summaries for each paper based on keywords, abstracts, introductions, methodologies, key findings, and conclusions. This approach allowed the authors to efficiently identify the most relevant and unique contributions to the research topic, ensuring a comprehensive yet focused literature review.

— **Cross-Checking for Originality:** Using the citation maps and research summaries generated by Litmaps.com and GPT-4o, the authors cross-checked the final 15 identified articles to confirm their originality and relevance. This ensured that foundational works and significant contributions were incorporated into the analysis.

Since the primary focus, and consequently the potential novelty, of this research lies in the domain of systems analysis, a review of related sources was conducted to identify relevant works addressing the issue of requirements management in Agile projects. However, the analysis was approached with an emphasis on examining the problem from the perspective of various disciplines within the field of Computer Science, employing diverse methodologies both for investigating and potentially resolving the issue. This approach aimed to encompass a broader range of potentially relevant studies. Accordingly, the search for related works was carried out across the domains of Systems Analysis, IT Project Management, Information Systems and Networks, Knowledge Bases, and Machine Learning.

The following keywords were used for the search: *Agile, agile requirements engineering, process formalization, agile methodology, software requirements management, overview, knowledge based system, computer model, conceptual model, software requirements management bottlenecks, software requirements workflow, dataflow, dataflow perspective.*

Among other search requests the following keywords combinations were the most successfully applied and that brought the most related research to the table:

1. "Agile requirements engineering process formalization" [5, 6];
2. "Agile software requirements workflow critical blocks" [7, 8];
3. "Agile requirements management from dataflow perspective" [9–12];
4. "Approach to model knowledge based system" [13];
5. "Computer model of agile requirements engineering process" [14, 15];
6. "Conceptual model of agile requirements engineering process" [16];
7. "Agile methodology impact the software requirements management" [17];
8. "Overview the software requirements management bottlenecks" [18, 19].

Common Theme Across All Works. The central theme uniting all 15 academic works is the exploration of innovative approaches to managing requirements in software development to address the increasing complexity, dynamism, and scale of modern software projects.

Agility and Adaptability. Almost all papers emphasize incorporating Agile practices to improve adaptability in requirements engineering (e.g., [5, 6, 11, 14]) and focus on iterative processes, continuous stakeholder engagement, and incremental development is a recurring thread.

Addressing Challenges: managing complexity (functional and non-functional requirements) is a shared concern, addressed differently as NFR management [7,8] and as Large-scale Agile practices [11].

Integrating AI into requirements engineering [9].

Hybrid and Modular Approaches. Several works propose hybrid methodologies to blend Agile with traditional or specialized methods [7,10,18]), and modular frameworks like metamodels are used to enable scalability and flexibility [5,14].

Validation and Practicality. Most papers demonstrate their solutions through real-world validations, such as case studies [5,10,17]. In addition, the industry relevance is emphasized, aiming for solutions that address real challenges in software projects.

Focus on Stakeholders. Prioritizing user needs and continuous feedback is universally stressed, whether through Agile user stories [6] or human-centered design [12,14].

Finally, **Scalability and Globalization:** tackling scalability issues and managing distributed development are recurring challenges highlighted in GSD contexts [17,19].

The comparison analysis was conducted to highlight differences between works, that are significant due to the wide range of approaches used to research the Agile requirements management and engineering issues. The outcomes of the analysis are placed below in Table 1.

Table 1. Comparison analysis of the found works.

Criteria	**IDs and Details**
Topics	Agile Practices and Tools: [5,6,14,17]
	Non-Functional Requirements (NFRs): [7,8]
	Artificial Intelligence in RE: [9]
	Global Software Development: [17,19]
	Hybrid/Modular Approaches: [10,18]
	Large-Scale Agile Development: [11]
	Design Thinking and Data Modeling: [12].
Methodologies	Case Studies: [5,10,11,17]
	Empirical Studies or Surveys: [16,19]
	Literature Reviews: [18,19]
	Design Science Framework: [9]
	Prototyping: [17]
	Quantitative Analysis: [16]
	Modeling Approaches: [5,10,14].
Findings	Effectiveness of Agile: [5,16]
	Challenges in NFR Management: [4,7]
	Hybrid Models for Complex Requirements: [10,17]
	AI Enhancements: [9]
	Scalability and Adaptability: [11,14].
Pros	Innovative Solutions: [10,17]
	Practical Validations: [5,10,17]
	Theoretical Contributions: [9,18]
	Cross-Disciplinary Integration: [12,14].
Cons	Limited Sample Sizes: [6,17,19]
	Scalability Concerns: [5,14]
	Tool Dependencies: [13,18]
	Future Work Needed: [8,19].
Highlighted Patterns	Emerging Trends: AI integration in RE [9]
	Modular approaches [10,17]
	Scalability Limitations: Challenges in large projects [5,14]
	Focus on Practicality: [5,11,17]
	Hybrid Approaches: Combination of methodologies in [10,14,17].

Innovation of this paper. The analysis reveals several key aspects regarding the critical points in the workflow of requirements management systems in Agile projects. For requirements elicitation and analysis, the works demonstrate improved understanding and prioritization of stakeholder needs through techniques such as metamodels and user story clustering. Data-driven approaches leverage big data for actionable insights, while human-centered design enhances alignment with business goals. However, these approaches often rely heavily on expertise, making implementation challenging for smaller teams. Ambiguity in stakeholder needs, especially in AI-driven systems, remains a persistent issue. In the area of requirement modeling and traceability, the studies highlight enhanced traceability through the use of metamodels and modeling tools, ensuring alignment between requirements and implementations. Integration with Agile practices and visual representation tools like flow diagrams also improve communication. Despite these benefits, scalability challenges and dependency on custom tools and languages limit broader applicability. Moreover, integrating non-functional requirements effectively into Agile workflows is an ongoing difficulty.

For requirement validation, iterative approaches allow evolving requirements to be continuously tested, and stakeholder engagement improves accuracy. However, these methods can be resource-intensive and require significant coordination. Some solutions lack sufficient empirical validation, raising concerns about their generalizability across different Agile contexts. Requirement change management benefits from hybrid approaches that structure mechanisms for handling dynamic changes while leveraging knowledge reuse systems. However, frequent changes can lead to overcommitment and technical debt, particularly in large-scale Agile environments. Additionally, limited interoperability between tools complicates the management of changes. In terms of requirement documentation, Agile principles favor minimal documentation, reducing overhead and enabling faster development cycles. Structured artifacts introduced in some works address the need for documenting critical requirements while maintaining agility. However, minimal documentation may be inadequate for large or regulated projects, and consistency issues can arise in global or distributed contexts. Collaboration and communication are enhanced by workflows that improve inter-team interactions and tools that streamline workflows. Despite these advancements, cultural and geographical barriers continue to challenge effective communication, particularly in globally distributed teams. Furthermore, tools requiring specialized training can hinder collaboration and slow onboarding processes.

Only few of the reviewed works fully correspond to the manifested problem statements and the approach of the current research by analyzing requirements management in Agile projects from the perspective of data flow analysis and identifying critical nodes where information throughput fails to meet project needs. Reference [13] partially addresses the mentioned issues, as it focuses on data flow analysis to improve software systems. It proposes a methodology using a domain-specific language (DSL) to create flow diagrams, which can help identify critical nodes. However, its primary focus is on reducing technical debt and improving quality, rather than the specific challenges of Agile projects. Reference [11] is partially relevant to the topic, focusing on managing the flow of requirements in large-scale Agile development. However, it emphasizes strategic planning and communication rather than data flow analysis or identifying critical nodes. Reference [10] explores the integration of diverse data sources into requirements elicitation in Agile, addressing the complexity and diversity of big data. This is partially related to the challenges of information throughput but focuses on adaptive methodologies rather than classical data flow analysis. Reference [5] is also partially relevant, as the metamodel provides traceability between requirements and goals, which could assist in identifying critical nodes. However, it does not explicitly analyze data flows. Reference [6] focuses on clustering user stories using machine learning algorithms but does not address data flow analysis or critical nodes. Reference [7] explores managing non-functional requirements, with no mention of data flow issues in Agile. Reference [8] focuses on non-functional requirements in model-driven development, unrelated to Agile data flow challenges. Other studies [9, 12, 14–19] have limited or no focus on trying to solve issues with the Agile projects' requirements engineering and management via data flow analysis and critical nodes identification.

Thus, based on the analysis of the most relevant studies in terms of the research topic and object, the authors concluded that none of the examined works addressed the issue of requirements management in

Agile projects from the perspective of data flow analysis. Specifically, there was no focus on identifying critical nodes where information throughput fails to meet project needs for various reasons. This gap ultimately leads to long-term challenges in managing and maintaining software requirements in Agile projects, reducing their efficiency and increasing resource consumption. Consequently, this study is highly relevant among related research efforts, which primarily aimed to address isolated symptoms of requirements management issues in Agile or optimize specific aspects of the process as a whole.

3. Previous research steps

The first step of the research. The first step of the research involved aggregating the primary quality requirements for software specifications and validating their importance. Through a comparative analysis of the data collected during the survey study conducted in the previous stage and the identified literature on the quality of software requirements and specifications, 261 relevant sources were discovered using the Google Scholar database, published no later than 2020. Using the systematic literature review method [23], 16 key sources were selected. The list and prioritization of the identified quality requirements for software specifications are presented in Figure 1.

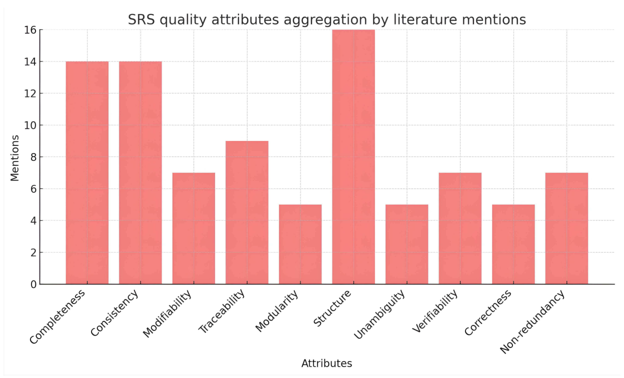

Fig. 1. Prioritized list of identified quality requirements for software specifications.

The second step of the research. The second step involved analyzing the current tools used in software requirements management systems in Agile projects to assess their compliance with the identified quality requirements for software specifications. Atlassian Jira (ITS) and Confluence (Wiki

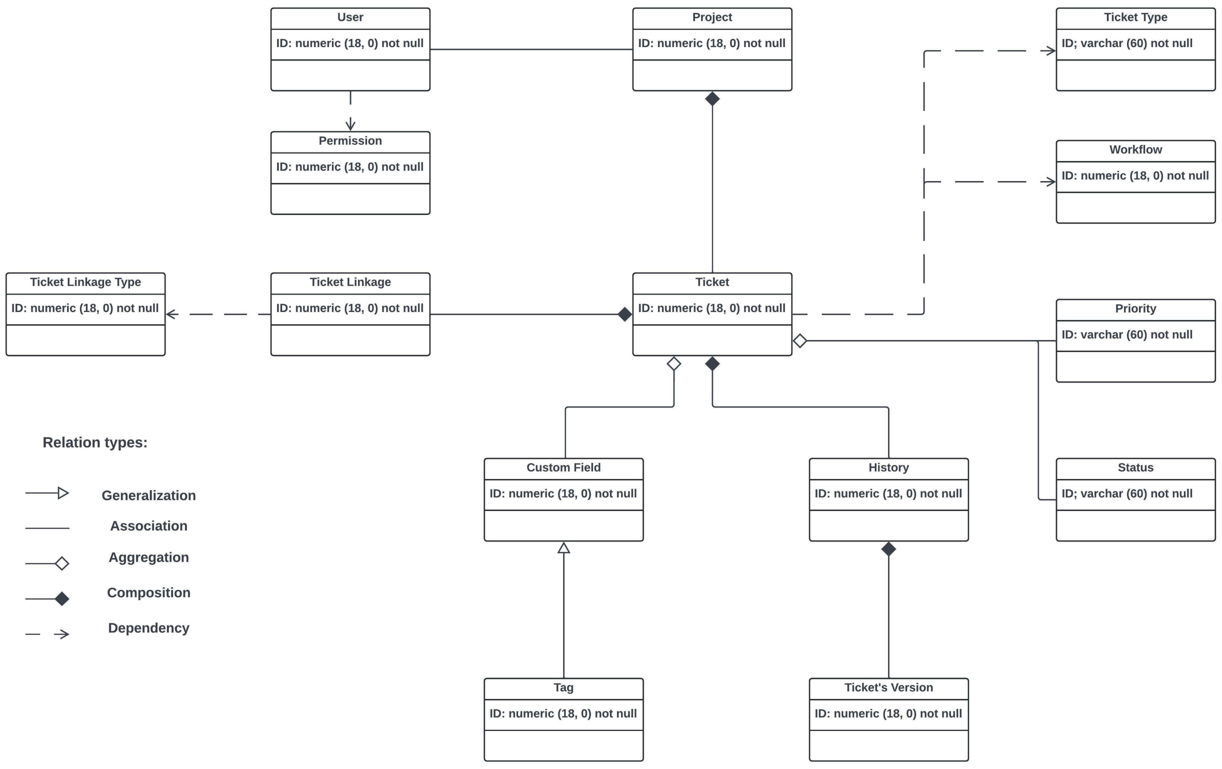
User
ID: numeric (18, 0) not null
Project
ID: numeric (18, 0) not null
Ticket Type
ID; varchar (60) not null
Permission
ID: numeric (18, 0) not null
Workflow
ID: numeric (18, 0) not null
Ticket Linkage Type
ID: numeric (18, 0) not null
Ticket Linkage
ID: numeric (18, 0) not null
Ticket
ID: numeric (18, 0) not null
Priority
ID: varchar (60) not null
Relation types:
Generalization
Association
Aggregation
Composition
Dependency
Custom Field
ID: numeric (18, 0) not null
History
ID: numeric (18, 0) not null
Status
ID; varchar (60) not null
Tag
ID: numeric (18, 0) not null
Ticket's Version
ID: numeric (18, 0) not null

Fig. 2. High-level entity-relationship model of a typical ITS tool.

system) were selected for this analysis as they are among the most widely used tools in the Ukrainian IT industry and offer the most comprehensive set of features [24, 25]. The reverse engineering method and data modeling were applied for the analysis, including a high-level entity-relationship model for this system, presented in Figure 2.

The study revealed that these tools satisfy most quality requirements, such as structure, adaptability, traceability, modularity, and verifiability. However, these qualities apply either to the system's work items or work tickets, which describe task setting and the scope of work in general, including software requirements, or to the project's documentary artifacts. In both cases, supporting the quality of the specification is not a primary function of the system, making it impossible to ensure compliance with software specification quality requirements using these management systems with minimal resource involvement.

4. Methodology

At this stage of the research, an inductive approach was adhered to. The research methodology utilized the system analysis and modeling methodology described in [20, 21], and the recommendations of the International Institute of Business Analysis (IIBA) [22], which are widely used for system analysis in the modern IT industry. Specifically, to achieve the above-mentioned research objectives, the following methods were applied:

1. The method of comparative analysis;
2. The method of reverse engineering as per [21];
3. The observation method as per [22];
4. The method of data flow modeling using notations described in [21, 22];
5. The system stock-and-flow modeling method using notation from [20].

This study employed a data-flow modeling approach to analyze the software requirements flow in software development projects conducted under the Agile methodology. The modeling process was based on an observation of three commercial projects carried out by the software development company SoftServe. These projects were selected to represent distinct project categories to ensure diverse and comprehensive insights into Agile requirements management. Specifically, the observed projects included:

1. **Startup Project**: A newly initiated software development project characterized by dynamic requirements and rapid development cycles.
2. **Legacy Project**: A project focused on maintaining and upgrading existing software systems with a substantial codebase and established workflows.
3. **Continuous Development Project**: An ongoing project involving iterative development and frequent delivery of updates to a mature software system.

The observations were conducted over the course of active development cycles for each project type. Key data points included the flow of requirements through various stages of the Agile process, the interactions between stakeholders, and the application of Agile principles in managing evolving requirements. This methodological triangulation allowed for the identification of patterns and variances in requirements flow across different project contexts, contributing to a nuanced understanding of Agile practices in real-world commercial environments.

5. Findings

The third step involved modeling the data flows of the software requirements management system in Agile projects as a system of processes rather than specific tools. The first model was developed using the notation described in [21, 22] and is presented in Figure 3. The second model, created using the notation described in [20], is shown in Figure 4.

The context diagram in Figure 3, a variation of a data flow diagram, models external data flows and identifies external environments that interact with the software requirements management system in an Agile project. It represents static flows of information exchange but has the limitation of being unable to model causal relationships or depict a hybrid external-internal model of data flows.

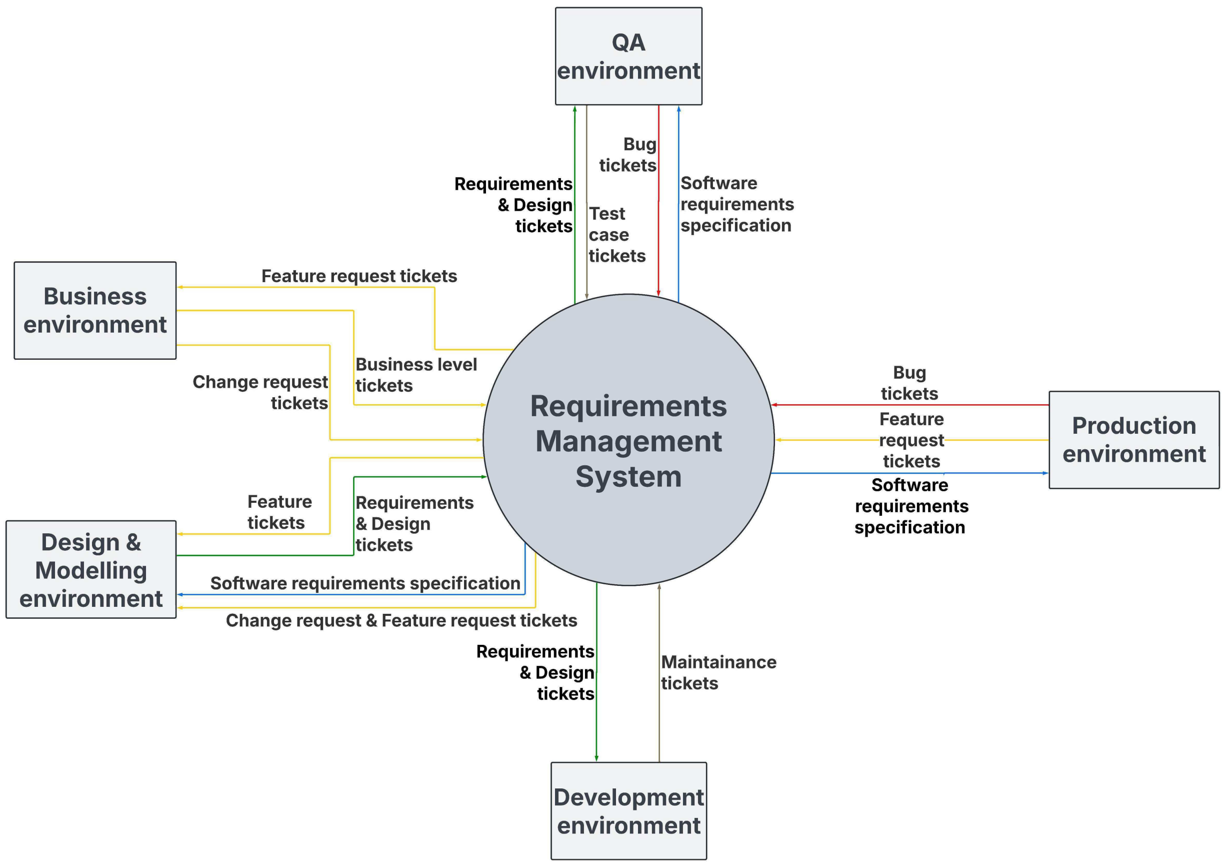
QA environment
Requirements & Design tickets
Bug tickets
Test case tickets
Software requirements specification
Business environment
Feature request tickets
Business level tickets
Change request tickets
Requirements Management System
Bug tickets
Feature request tickets
Software requirements specification
Production environment
Feature tickets
Requirements & Design tickets
Design & Modelling environment
Software requirements specification
Change request & Feature request tickets
Requirements & Design tickets
Maintainance tickets
Development environment

Fig. 3. Context diagram of the software requirements management system.

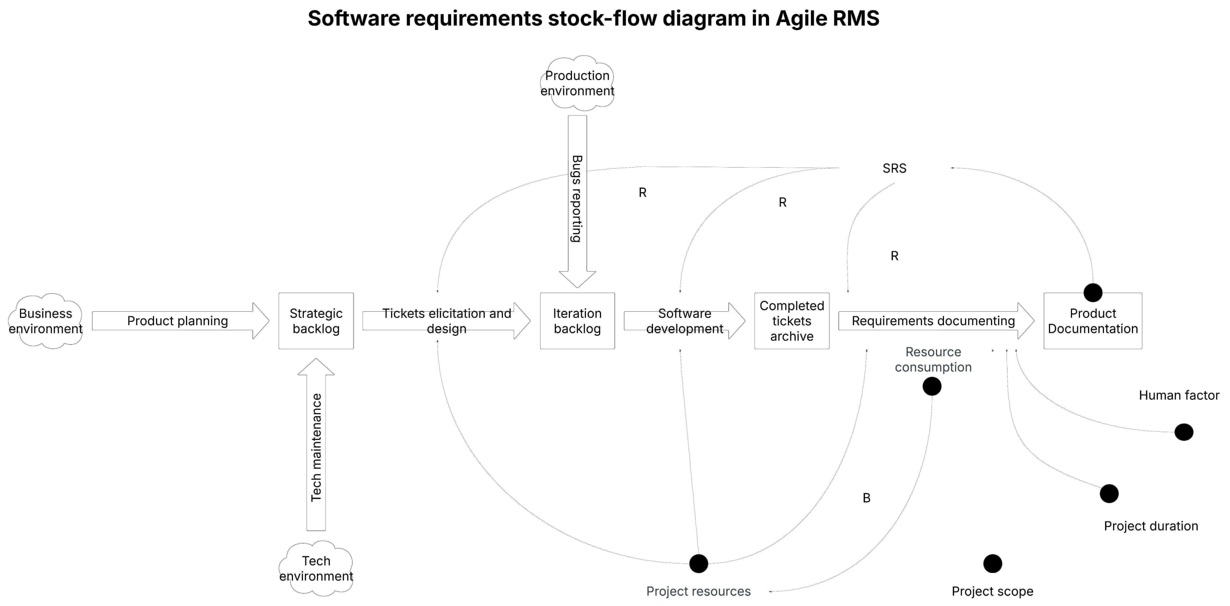

Fig. 4. Stock-and-flow diagram of software requirements in an Agile project.

The stock-and-flow diagram of software requirements in Figure 4, using the notation from [20], addresses the limitation of the previous model. It enables the modeling of a complex multiprocess system for managing software requirements in the context of treating software requirements as units of information about the software, their flow through project pipelines, and stock reservoirs at various project stages. This model illustrates the iterative process of planning, designing, developing, and documenting software requirements, while highlighting the interdependencies between various components of the system.

The process begins with inputs from the business and technical environments, which feed into the product planning stage. This stage generates a strategic backlog, serving as a repository for high-level requirements and objectives. From there, tasks are broken down during the tickets elicitation and design phase, which creates a more actionable iteration backlog. This backlog directs the software development process, culminating in the completion of specific tasks, which are then archived in the completed tickets archive. The model emphasizes the cyclical nature of Agile workflows through multiple feedback loops. The production environment provides bug reporting, which redirects tasks to the iteration backlog, ensuring continuous improvement. Additionally, completed tasks flow into the requirements documenting phase, which refines and formalizes the specifications, ultimately contributing to the creation of product documentation. Critical factors influencing the process are represented as stocks, including project resources, resource consumption, project scope, project duration, and the human factor. These variables interact dynamically, shaping the flow and outcomes of the system. The resource allocation affects the progress of strategic and iteration backlogs, while resource consumption and human factors influence project timelines and deliverables.

The model also accounts for the technical environment's role through tech maintenance, which sustains the strategic backlog and ensures the continuity of technical operations. By integrating these components, the diagram effectively captures the complexity and adaptability of Agile RMS, emphasizing its reliance on iterative cycles, feedback mechanisms, and resource management to produce high-quality software documentation. The function of this system is to "process" software requirements from the initial point of the system to the final one to produce implemented software, which is the goal and benefit of an IT project. Notably, in this model, the flow through each pipeline has a certain cost in terms of the project's resource units. Project resources, as an external factor, are distributed among different pipelines, either increasing or decreasing the number of requirements passing through the pipeline and accumulating in the reservoirs over a unit of time.

A critical observation is that the main benefit of the system from the IT project's perspective is obtained at the end of the "Software Development" pipeline. Therefore, the "Documentation of Requirements" pipeline (Pipeline No. 4) may seem less prioritized in resource allocation. As a result, Pipeline No. 4 may face resource shortages, leading to reduced throughput capacity and, consequently, a deficit in the "Documentation" stock. However, Pipeline No. 4 plays a crucial role in converting software requirements as units of information about the software into software specifications as units of knowledge about the software.

Feedback Loops. The modeling revealed the existence of self-reinforcing feedback loops between the software specification stock and the cost of pipeline throughput, particularly Pipeline No. 4. Specifically: The smaller the stock of software specifications, the higher the cost of pipeline throughput. Additionally, external factors such as project scalability, duration, and human factors influence the increase in the cost of Pipeline No. 4 throughput. A balancing feedback loop was also identified, which activates in response to excessively high throughput costs and encourages resource redistribution to Pipeline No. 4 to replenish the software specification stock and reduce the throughput cost of other pipelines. However, this loop acts as a compensatory mechanism and is too weak in the early stages of a software specification stock shortage.

Further Steps. Based on the findings of the conducted modeling, the authors conclude that, to optimize the process of requirements management within the Agile methodology, the focus should be placed on developing a complementary system that would balance the feedback loops acting on Pipeline No. 4 by reducing the cost of passing this pipeline for software requirements units while ensuring their subsequent transformation into a unit of product knowledge. To achieve this, the authors concentrated on data regarding the most demanded quality attributes of software requirements collected during the first step of the study, with the goal of designing a system capable of satisfying these quality attributes. Within the scope of the project, the authors reject the idea of training neural networks to ensure individual quality attributes of software requirements, considering it reasonable in general but beyond the initial objectives and scope of the study. Thus, since certain quality attributes are difficult to ensure without human intervention due to their definitions, the authors decided to focus on ensuring those quality attributes of software requirements, within the framework of the complementary system for Pipeline No. 4, that do not require specific neural network training. At the same time, the idea of utilizing a general-purpose generative AI model (such as GPT-4o, Gemini, etc.) is deemed unavoidable by the authors. As a result, the quality attributes of software requirements addressed by the complementary system are as follows:

— **Structure and Modularity** – authors anticipate usage of the method of semantic text analysis via generative AI models for keyword extraction and further software requirements clusterization. The same method may be applied to achieve modularity from the business domain perspective.
— **Traceability and Non-redundancy** – authors assume usage of both semantic text analysis and data mining methods to automate the on-fly maintenance of the valid product's data flows and cross-functional dependencies models based on input and output data analysis of system features.
— **Modifiability and Audit** – the standard knowledge and document management system, KDMS functionality stack, populated from the main nowadays requirements management tools, investigated during the second step of the research, will be able to establish ability to update software requirements specification on fly and maintain audit log, that will allow also to meet one of the main needs of developers, systems analysts and quality engineers – ability to quickly get the specification of a feature as of the specific date in the past.
— **Completeness (partially)** – however ability to identify whether the product needs more requirements or not is still the human responsibility, usage of the generative AI for gap analysis of a feature using the method of utilizing generative AI models for automated verification of the provided specification against predefined completion rules.

6. Conclusions

The novelty of this research lies in the analysis of the software requirements management system in Agile projects from the perspective of the stock-and-flow of software requirements. This approach helped identify independent external factors influencing the increasing cost of transforming software requirements as units of information into software specifications as units of knowledge. Additionally, a self-reinforcing feedback loop was discovered between the shortage of software specification stock and the increased cost of pipeline throughput in the model.

A balancing feedback loop was also identified, which forces the project to redistribute resources in response to increased throughput costs. However, this loop is insufficient for timely responses. Thus, a critical point in the system was successfully identified. Addressing it by increasing resources allocated to the "Documentation of Requirements" pipeline does not fully resolve the issue because:

— Independent constant factors still increase the cost of pipeline throughput over time;
— Increasing resource allocation to Pipeline No. 4 reduces resources for other pipelines, diminishing overall project profitability and potentially leading to unprofitability or non-competitiveness.

An alternative and, in the authors' opinion, more logical solution involves developing a mathematical model of a complementary system. This system would minimize the human factor in transforming software requirements into specifications that meet all quality criteria. It would also reduce the impact of project scalability and duration, thereby lowering the cost of Pipeline No. 4 throughput and ensuring an adequate stock of software specifications in the project. Summarizing, the next steps in this research are:

1. Development of a mathematical-logical model of a complementary system;
2. Development of a mathematical model of the processes in the software requirements management system to measure the efficiency of the complementary system.

However, since the data for modeling is collected from applied software projects, which are not publicly available on the global network, this poses a limitation on the model's accuracy. This also increases the risk that the initial versions of the complementary system may be non-universal and only partially applicable.

7. Approbation

This research was conducted within the project environment of SoftServe Inc. The findings were presented and validated at the VII International Scientific and Practical Conference 'Modeling, Control, and Information Technologies (MCIT-2024)' and the XII International Scientific and Practical Conference.

[1] Kruk R., Zhukovska N. Survey Study of Requirements Engineering Issues and Challenges in Agile Projects. Bulletin of the National university of water and environmental engineering. Technical Sciences. **103** (3), 195–212 (2023).

[2] Adenowo A., Adenovo B. A. Software Engineering Methodologies: A Review of the Waterfall Model and Object Oriented Approach. International Journal of Scientific and Engineering Research. **7** (4), 427–434 (2020).

[3] Pargaonkar S. A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. International Journal of Scientific and Research Publications. **13** (8), 120–124 (2023).

[4] Thesing T., Feldmann C., Burchardt M. Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. Procedia Computer Science. **181**, 746–756 (2021).

[5] Haidar H., Kolp M., Wautelet Y. Formalizing Agile Software Product Lines with a RE Metamodel. Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018). 90–101 (2018).

[6] Kumar B., Tiwari U., Dobhal D., Negi H. User Story Clustering using K-Means Algorithm in Agile Requirement Engineering. 2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES). 1–5 (2022).

[7] Rahy S., Bass J. M. Managing non-functional requirements in agile software development. IET Software. **16** (1), 60–72 (2022).

[8] Ameller D., Franch X., Gomez C., et al. Dealing with Non-Functional Requirements in Model-Driven Development: A Survey. IEEE Transactions on Software Engineering. **47** (4), 818–835 (2011).

[9] Dalpiaz F., Niu N. Requirements Engineering in the Days of Artificial Intelligence. IEEE Software. **37** (4), 7–10 (2020).

[10] Franch X., Henriksson A., Ralyté J., Zdravkovic J. Data-Driven Agile Requirements Elicitation through the Lenses of Situational Method Engineering. 2021 IEEE 29th International Requirements Engineering Conference (RE). 402–407 (2021).

[11] Heikkilä V. T., Paasivaara M., Lasssenius C., Damian D., Engblom C. Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. Empirical Software Engineering. **22**, 2892–2936 (2017).

[12] O'Driscol K. The agile data modelling & design thinking approach to information system requirements analysis. Journal of Decision Systems. **25** (sup1), 632–638 (2016).

[13] Poznański P., Wawrowski M. Improving Software Systems by Flow Control Analysis. Computer Science. **13** (2), 81–92 (2012).

[14] Schon E., Sedeño J., Mejias M., Thomaschewski J., Escalona M. J. A Metamodel for Agile Requirements Engineering. Journal of Computer and Communications. **7** (2), 1–22 (2019).

[15] Soundararajan S. Agile Requirements Generation Model: A Soft-structured Approach to Agile Requirements Engineering (2008).

[16] Serrador P., Pinto J. K. Does Agile work? — A quantitative analysis of agile project success. International Journal of Project Management. **33** (5), 1040–1051 (2015).

[17] Alsahli A., Khan H., Alyahya S. Toward an Agile Approach to Managing the Effect of Requirements on Software Architecture during Global Software Development. Scientific Programming. **2016**, 8198039 (2016).

[18] Khan M. N. A., Khalid M., Ulhaq S. Review of Requirements Management Issues in Software Development. International Journal of Modern Education and Computer Science (IJMECS). **5** (1), 21–27 (2013).

[19] Yaseen M, Ali Z., Khan M. H. Requirements Management Model (RMM): A Proposed Model for Successful Delivery of Software Projects. International Journal of Computer Applications. **178** (17), 32–36 (2019).

[20] Meadows D. H. Thinking in systems. White River Junction, Vermont, Chelsea Green Publishing (2008).

[21] Wiegers K., Beatty J. Software Requirements (3rd ed.). Microsoft Press (2013).

[22] IIBA A Guide to the Business Analysis Body of Knowledge (BABOK Guide) Version 3.0. International Institute of Business Analysis (2015).

[23] Kitchenham B., Charters S. Guidelines for performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report, EBSE-2007-0 (2007).

[24] Jira Cloud resources. https://support.atlassian.com/jira-software-cloud/resources/.

[25] Confluence Cloud resources. https://support.atlassian.com/confluence-cloud/resources/.

Аналіз вузьких місць на основі моделі потоку даних вимог програмного забезпечення в проектах Agile

Крук Р. А., Жуковська Н. А.

Національний університет водного господарства та природокористування, НУВГП, вул. Соборна, 11, 33028, Рівне, Україна

У цій статті розглядаються виклики управління вимогами до програмного забезпечення в Agile-проєктах, зосереджуючись на критичних точках робочих процесів, що впливають на якість вимог. Основні проблеми включають неповноту, непослідовність, відсутність простежуваності та недостатню структурованість, що посилюється ітеративною природою Agile. Порівняльний аналіз інструментів, таких як Jira та Confluence, виявляє їхні обмеження у забезпеченні якості вимог. Використовуючи модель запасів і потоків, дослідження розглядає систему управління вимогами як взаємопов'язаний процес, виявляючи зворотні зв'язки, що знижують ефективність. Результати вказують, що оптимізація процесів специфікації програмного забезпечення за допомогою допоміжної системи може пом'якшити ці проблеми, мінімізуючи людське втручання. Дослідження закладає основу для розробки математичної моделі, спрямованої на підвищення ефективності управління вимогами в Agile-проєктах.

Ключові слова: *Agile; інженерія вимог; вимоги до ПЗ; концепнуальна модель; системний аналіз.*