Structuring complexity: A functional evaluation of Jira tools for requirements management

Agnese Rozenberga

Institute of Applied Computer Systems, Riga Technical University, Kipsalas iela 6A, Riga LV 1048, Latvia

Abstract

In complex software environments, effective requirements management (RM) is essential for aligning stakeholder needs, managing uncertainty, and maintaining traceability across development stages. While platforms like Jira are widely used, they lack comprehensive native support for core RM functions, leading to reliance on third-party extensions. This paper presents a structured, lifecycle-oriented method for evaluating and selecting RM tools within the Jira ecosystem. Building on established RM functional dimensions, a decision support tool was developed to help users identify the tools best suited to their project-specific needs. The tool enables users to prioritize RM capabilities and compare them against actual tool performance across six lifecycle stages: elicitation, analysis, specification, validation, management, and traceability. Expert feedback validated the relevance and usability of the tool. The findings underscore the importance of structured tool selection in managing RM complexity, especially in environments where no single solution offers full lifecycle support. The study contributes a replicable approach to RM tool evaluation and offers practical guidance for improving decision-making in tool adoption and configuration.

Keywords

requirements management, Jira, software complexity, decision support tool, project-specific tool selection

1. Introduction

Modern software development is defined by escalating complexity driven by globally distributed teams, rapidly evolving requirements, and increased regulatory and business demands. In this context, Requirements Management (RM) has emerged as a strategic function rather than a secondary support activity. RM plays a central role in reducing ambiguity, aligning stakeholder expectations, and ensuring traceability throughout the development lifecycle, thus minimizing costly rework and delivery risks [1, 2].

Although tools like Jira [3] are widely adopted for agile project coordination, their native functionality provides only partial support for formal RM processes. To address these limitations, the Atlassian Marketplace offers a growing ecosystem of extensions aimed at enhancing RM capabilities within Jira. However, the overlapping, inconsistent, and sometimes opaque feature sets of these tools present a significant challenge to teams seeking solutions aligned with specific project contexts, particularly in high-stakes or compliance-driven environments.

This research addresses that gap by:

1. Evaluating Jira and four Jira-compatible RM tools (rmCloud [4], RMsis [5], EaseRequirements [6], TraceCloud [7]) against six RM lifecycle stages: elicitation, analysis, specification, validation, management, and traceability;
2. Applying a structured evaluation framework to assess each tool's functional coverage in these lifecycle phases;
3. Developing a decision-support tool that helps users select RM tools based on project-specific functional priorities;
4. Validating the selection tool through expert feedback and acknowledging recommendations for future improvements.

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

agnese.rozenberga@edu.rtu.lv (A. Rozenberga)
0009-0001-9256-6938 (A. Rozenberga)

By linking project-specific RM needs with actual tool capabilities, this work contributes a practical method for supporting structured, context-aware tool adoption. The outcome helps teams better navigate complexity and improve requirements practices in software projects of varying scale and criticality.

2. Related work

Research on requirements engineering (RE) tools has long emphasized the need for structured evaluation frameworks that account for functionality across the lifecycle. Hoffmann et al. [8] outlined essential requirements for RM tools, highlighting the importance of features such as change management, traceability, and version control. More recent systematic reviews, such as Fontoura and Otávio [9], continue to document persistent challenges in RM practice and the fragmented tool support available. Several studies have investigated requirements management in agile and large-scale contexts, which are highly relevant to platforms like Jira. Fucci et al. [10] reported on the needs and challenges of platforms to support large-scale requirements engineering, emphasizing the lack of integrated solutions. Kamal et al. [11] examined success factors for agile requirements change management in distributed projects, while Käpyaho and Kauppinen [12] demonstrated how prototyping can enhance agile RE practices. Similarly, Lan and Balasubramaniam [13] conducted an empirical study of agile RE practices, underscoring the need for lightweight but structured tool support. These works establish that agile environments demand tools capable of balancing flexibility with rigorous requirements tracking and validation. Despite the widespread adoption of Jira for agile project management, research directly addressing its suitability for RM is limited. Filion et al. [14] presented an industrial case study on using Atlassian tools for requirements management, showing their potential but also exposing limitations in areas such as traceability and specification. More recently, Raatikainen et al. [15] highlighted the challenges of managing issue dependencies in large collaborative projects, a problem highly relevant to Jira's issue-tracking foundation.

To validate this gap, a targeted literature search was conducted across Google Scholar, IEEE Xplore, ACM Digital Library, ScienceDirect, Cambridge Journals Online, Wiley Online Library, and O'Reilly. Search queries included *"Jira requirements management"*, *"requirements engineering tools in Jira"*, *"Jira plugins for requirements management"*, *"requirements traceability in Jira"*, and *"Jira marketplace requirements"*. The search did not return studies that systematically evaluate Jira-compatible RM extensions. This lack of research confirms that while generic RE tool assessments exist, there is no dedicated guidance on how Jira Marketplace tools address RM shortcomings.

This paper builds on lifecycle-oriented evaluation approaches [16] but extends them by applying a structured functional framework specifically to Jira-compatible tools. By combining lifecycle-based assessment with a decision-support tool that integrates project-specific priorities, it bridges the gap between generic RE tool surveys and the plugin-driven ecosystem of Jira, offering actionable guidance for practitioners.

3. Mapping requirements management phases to Jira functionality

To effectively support RE in complex projects, it is essential to understand how each stage of the RM lifecycle maps to the actual capabilities of Jira. This chapter examines how Jira's built-in components align with the six core RM phases elicitation, analysis, specification, validation, management, and traceability highlighting their functional roles across the project lifecycle.

The lifecycle-oriented method presented in this study is not bound to a specific software development methodology. While Jira is commonly used in agile contexts, the evaluation framework applies equally to other lifecycle models. The term "lifecycle-oriented" refers to the alignment of tool capabilities with the six established RM stages elicitation, analysis, specification, verification and validation, management and tracebility rather than to a particular development model. RM spans multiple interrelated activities throughout the project lifecycle, each aimed at ensuring that stakeholder needs are captured, understood, and fulfilled effectively [17]. As summarized in Table 1, each phase of the RM lifecycle is associated with specific Jira features.

Table 1
Mapping of RM lifecycle stages to Jira components

RM lifecycle stage	Supporting Jira components
Elicitation	Forms, Backlog, Goals
Analysis	Backlog, Kanban, Board, Forms, Goals, Estimation, Reports
Specification	Forms, All Work, Backlog
Validation	Estimation, Backlog, Forms, Reports
Management	Backlog, Summary, Board and Kanban, Sprint, All Work, Calendar, Reports
Traceability	Board, List, Goals, Kanban, All Work

The process begins with elicitation - gathering the foundational knowledge needed to define solution expectations. This includes identifying system goals, user concerns, and relevant constraints, forming the basis for further analysis [17]. In Jira, this is supported by components such as *Forms*, which collect structured input from stakeholders; the *Backlog*, which organizes early-stage requirements; and *Goals*, which define intended outcomes.

In analysis stage, requirements are clarified, organized into logical categories, and reviewed for conflicts [17]. Reaching consensus among stakeholders is essential, especially when priorities or expectations diverge. A clear, consistent requirements set is necessary to define a viable solution strategy. In Jira, this is supported by components such as the *Backlog*, where items are reviewed and prioritized; *Kanban* and *Board*, which help visualize task flow and identify bottlenecks; *Forms*, which ensure consistent input for updates; and *Goals*, which provide alignment with business objectives. Additionally, *Estimation* allows for evaluating effort, while *Reports* offer insights into requirement trends and progress.

Specification involves formally documenting requirements in a way that aligns with their intended use and audience [17]. Specifications can vary in format and level of detail but serve as both a knowledge base and a reference for managing future changes or uncertainties in system development. In Jira, this is supported by components like *Forms*, which standardize input formats; *All Work*, which provides a comprehensive overview of all documented items; and the *Backlog*, where specified requirements are stored and structured for future development.

Validation ensures that requirements are accurate, feasible, and aligned with business goals [17]. Because most requirements originate from human input often informal or ambiguous validation is a critical filter, ensuring that decision-making rests on well-defined, traceable information. In Jira, this process is supported by *Estimation*, which helps evaluate implementation effort; the *Backlog*, where requirements can be reviewed before development; *Forms*, which support structured approval workflows; and *Reports*, which provide insights into progress, quality, and completion status.

Management stage includes structuring requirements in repositories, applying metadata (such as status or priority), and ensuring version control [17]. This supports both day-to-day coordination and long-term adaptability. In Jira, this is supported by multiple components: the *Backlog* enables tracking and reprioritization of ongoing work; *Summary* provides a high-level project snapshot; *Board* and *Kanban* views manage real-time task flow; *Sprint* organizes iterative work; *All Work* centralizes access to tasks; *Calendar* tracks timelines and deadlines; and *Reports* provide structured insights into team performance and project health.

Traceability connects requirements to related deliverables such as test cases, designs, and strategic objectives [17]. It plays a central role in impact analysis, especially when requirements change. Without a functioning traceability framework, it becomes difficult to maintain project coherence or ensure that the solution fulfills its intended purpose. In Jira, traceability is supported by the *Board*, which provides a clear visual representation of work status; the *List*, which enables structured tracking of individual items; *Goals*, which connect requirements to strategic outcomes; *Kanban*, which manages flow and dependencies; and *All Work*, which offers visibility into related issues. Although traceability is often categorized in literature as a sub-activity of requirements management, in this study it is treated as a distinct stage. This separation is motivated by its cross-cutting role across lifecycle phases and its explicit implementation in many Jira plugins, making it more practical to evaluate as independent functionality.

While this section mapped the RM lifecycle stages to Jira's built-in components, such alignment is not sufficient for conducting a structured evaluation or comparison. To transition from descriptive mapping to functional assessment, it is necessary to establish a concrete set of evaluation criteria. These criteria must capture the essential activities and quality objectives associated with each phase of the RM lifecycle.

4. Functional criteria for requirements management tools

The study "*Requirements Engineering Tools: Capabilities, Survey and Assessment*" [16] proposed a classification framework for evaluating RM tools based on the functional aspects they support across different stages of the RM lifecycle. Building on this framework, this work adopts and refines these functional dimensions to establish a consistent set of evaluation criteria. An overview of these criteria, organized by lifecycle stage and accompanied by their abbreviated labels (shown in brackets), is presented in Table 2. These labels are used throughout the remainder of the analysis to ensure clarity and traceability across tool comparisons.

Table 2
Functional capabilities of RM tools across lifecycle stages

Stage	Functional capabilities
1. Elicitation	Tools should support identifying relevant stakeholders (*Stakeholder identification*), documenting various types of requirements (business, user, functional, non-functional) (*Requirement capture*), and tracking the information gathered during this stage (*Traceability at elicitation*).
2. Analysis	This stage focuses on breaking down high-level requirements into detailed elements (*Requirement breakdown*), evaluating their feasibility (*Feasibility*), ranking their importance (*Priority setting*), resolving conflicting inputs (*Conflict detection*), and identifying ambiguous or incomplete information (*Ambiguity identification*).
3. Specification	Effective specification involves clearly documenting what the system must achieve, along with any constraints, using accessible and structured formats (*Formal requirement documentation*). Some tools enable the creation of formal or visual representations of requirements to enhance clarity (*Requirement modeling support*).
4. Verification & Validation	This function ensures that the specified requirements are accurate, testable, and aligned with business objectives, using review and approval mechanisms (*Verification & validation support*).
5. Management	Tools should assist with prioritization, tracking changes, maintaining historical versions, and ensuring that evolving requirements remain consistent with the product vision (*Change & maintenance support*).
6. Traceability	Traceability refers to the ability to monitor and manage the progression of requirements throughout the project lifecycle (*Lifecycle documentation*), to establish connections between related requirements such as dependencies or hierarchies (*Traceability mechanisms*), and to track modifications over time, including their rationale and impact (*Change tracking*).

5. Requirements management functionality analysis in the Jira environment

To assess Jira's ability to support key RM functions, each stage of the RM lifecycle was evaluated using a structured scoring system. The evaluation focused on determining which functional aspects Jira provides fully, supports partially, or lacks entirely. This method enables a precise understanding of how Jira performs across each phase and where it may require additional configuration or third-party tools to meet project needs.

Each RM function was rated on a three-point ordinal scale:

- 2 – Fully Supported: the function is available and usable without significant modification.
- 1 – Partially Supported: the function is available but requires manual configuration, workarounds, or has limited scope.
- 0 – Not Supported: the function is unavailable or cannot be meaningfully implemented.

This structured approach allows for a clear comparison of Jira's built-in and configurable components across six RM stages: elicitation, analysis, specification, verification & validation, management, and traceability. It highlights both Jira's current strengths and functional limitations in supporting comprehensive RM practices, especially in complex project environments.

The scoring was conducted by the author based on hands-on testing of the tools, supported by vendor documentation and trial usage.

5.1. Jira functional support by requirement management stage

Elicitation involves identifying stakeholders and gathering preliminary requirements in various formats [16]. In Jira, stakeholder engagement begins with user and role assignments, enabling collaboration and access control. Tools such as Forms support structured input, while Backlog and Goals help organize early-stage needs. However, Jira lacks native mechanisms for capturing different requirement types (e.g., business vs. technical), requiring users to manually configure issue types and custom fields. Conclusion: Jira provides partial support for elicitation. While stakeholder identification and basic input collection are manageable, the system does not inherently support structured or diverse requirement capture. (*See Supplementary Table A1 of [18] for scoring details.*)

Analysis phase focuses on breaking down high-level requirements, evaluating feasibility, prioritizing tasks, and resolving conflicts [16]. In Jira, Kanban and Board components facilitate decomposition via linked issues and epics. Forms and Backlog support structured planning, while Estimation allows teams to assess effort. Priority can be assigned during issue creation, and conflicting or ambiguous items may be refined iteratively through Sprint planning. Conclusion: Jira offers strong support for prioritization and feasibility evaluation. With customization, it also accommodates breakdown and conflict detection, although ambiguity resolution remains largely manual. (*See Supplementary Table A2 of [18] for scoring details.*)

Specification requires the formal documentation of requirements in structured and accessible formats [16]. In Jira, custom issue types and Forms allow users to define and organize requirements. The All Work and Backlog views provide overviews of documented items. However, Jira lacks modeling capabilities or formal specification frameworks out of the box, making it less suitable for projects that require visual or testable requirement structures. Conclusion: Jira partially supports specification. While it enables documentation and organization, it falls short in offering modeling or formal verification features.(*See Supplementary Table A3 of [18] for scoring details.*)

Verification and Validation ensures that requirements are complete, and aligned with business goals [16]. In Jira, this is approximated through Estimation, Backlog review, and approval workflows via Forms. However, these features are designed more for task management than RM validation. Conclusion: Jira allows for basic verification and validation activities with component customization. It does not natively provide features for test coverage or requirement reviews beyond general issue tracking. (*See Supplementary Table A4 of [18] for scoring details.*)

Management involves tracking changes, versioning, and organizing requirements throughout the lifecycle [16]. Jira's components including Backlog, Summary, Calendar, and Sprint enable dynamic task management and progress tracking. The Board and Kanban views support workflow transparency, while issue history and status fields provide basic traceability. Conclusion: Jira supports RM primarily through its task tracking features. However, dedicated requirement versioning or structured change control is not available by default. (*See Supplementary Table A5 of [18] for scoring details.*)

Traceability links requirements to related artifacts (e.g., tasks, tests, goals) and allows teams to monitor changes over time. Jira supports issue linking, hierarchical relationships, and change history. Strategic alignment can be visualized using Goals, while tools like Board and List facilitate navigation across dependencies. Still, traceability in Jira requires consistent user discipline or plugin support to maintain reliability over time. Conclusion: Jira provides partial support for traceability. While its linking mechanisms are flexible, they lack the structure required for formal RM processes. (*See Supplementary Table A6 of [18] for scoring details.*)

This evaluation reveals that Jira's native functionality offers broad but uneven support for RM across the lifecycle. While components like Backlog, Forms, and Kanban enable task-driven requirement tracking, they fall short of meeting the needs of structured RM environments, particularly in specification, validation, and traceability.

These findings underline the importance of context-aware tool selection and motivate further analysis of Jira extensions, which is explored in the next chapter.

5.2. Evaluation of Jira compatible requirements tools

This section extends the evaluation logic introduced in Section 5 (*Requirements Management Functionality Analysis in the Jira Environment*) to selected Jira Marketplace extensions, hereafter referred to as "tools" for simplicity. Although these are technically Marketplace apps, they are assessed in the same way as Jira itself using the structured scoring framework (0–2 scale) that measures the extent to which specific RM functions are supported across the lifecycle.

To identify relevant Jira-compatible tools for RM, a targeted search was conducted in the Jira Marketplace using the keyword phrase "Requirement management." From this search, four tools were selected for evaluation:

- rmCloud [4]
- RMsis [5]
- EaseRequirements [6]
- TraceCloud [7]

These tools were chosen because they are explicitly designed for requirements management, with feature sets clearly targeting RM activities. Selection was further guided by the clarity of their feature documentation and availability for user access (via free trial or demo versions). By applying the same structured comparison framework as in Section 5, the analysis highlights which RM capabilities are directly supported, which require workarounds, and where functional limitations may introduce complexity or risk. While the full evaluation dataset is embedded in the decision-support tool developed for this study, the subsections below summarize each tool's performance across the six RM phases: elicitation, analysis, specification, verification & validation, management, and traceability. Functional limitations captured in stage-specific "Notes" were not incorporated into the tool's scoring logic, but they are reported here to aid interpretation of tool suitability.

Elicitation phase for analyzed tools in Jira Marketplace:

rmCloud excels in structured requirement entry with fields like title, priority, and type. It also supports traceability during early exploration, but its tendency to duplicate issues when re-linked introduces coordination challenges.

RMsis provides detailed filtering and attribute capture for requirements but lacks stakeholder-specific access or role management.

EaseRequirements uses Jira's issue types and custom forms to support detailed requirement input. However, it lacks formal mechanisms for traceability during elicitation, relying on informal tags.

TraceCloud enables hierarchical structuring and progress tagging of early requirements, but like most other tools, it lacks stakeholder role visibility and mapping. (*See Supplementary Table B1 of [19] for scoring details.*)

Notes:

- **rmCloud** creates duplicate Jira issues on re-linking.
- **EaseRequirements** needs predefined "requirement" issue type in Jira.

Analysis phase for analyzed tools in Jira Marketplace:

rmCloud supports feasibility evaluation and manual prioritization through configurable fields but lacks mechanisms for identifying ambiguity or conflicts, which limits its effectiveness in high-stakes projects.

RMsis offers strong support across all analysis functions, including structured sorting and explicit fields for conflict and ambiguity tracking though these must still be managed manually.

EaseRequirements enables hierarchical decomposition and uses Jira's built-in prioritization, but offers no direct support for feasibility or inconsistency checks, making it less suited for complex analysis tasks.

TraceCloud allows folder-based organization and progress tracking but lacks prioritization and conflict-handling capabilities, reducing its utility for navigating uncertainty. (*See Supplementary Table B2 of [19] for scoring details.*)

Notes:

- TraceCloud lacks native prioritization support.

Specification phase for analyzed tools in Jira Marketplace:

rmCloud provides structured documentation using customizable fields and folders. It enables sorting and categorization and supports basic exports, which help teams communicate and archive specifications. However, it lacks support for formal modeling or approval workflows.

RMsis delivers robust support for specification, offering detailed data fields (e.g., criticality, dependencies) and baseline history for version control. Requirements can be linked logically or grouped hierarchically, aiding traceability and structured refinement.

EaseRequirements stands out for collaborative specification. It allows sub-requirements, team comments, and versioning directly within Jira. It also supports requirement modeling, making it better suited for distributed teams dealing with complex or evolving systems.

TraceCloud offers flexible documentation through folders and metadata, with version history and export options. However, it lacks formal approval mechanisms, limiting control over the specification's state transitions.

Jira itself can be adapted to support requirement specification via custom issue types and fields but does not support formal modeling or consistency checks without third-party extensions. (*See Supplementary Table B3 of [19] for scoring details.*)

Notes:

- EaseRequirements requires Jira permission setup.
- TraceCloud lacks active status change visibility for approval.

Verification phase for analyzed tools in Jira Marketplace:

rmCloud supports basic validation through change-tracking and status transitions (e.g., "under review"). Although it lacks formal approval workflows or reviewer roles, its visible edit history provides a transparent audit trail.

RMsis enables lightweight validation via a thumbs-up approval and commenting system, making it suitable for fast feedback loops. However, it lacks deeper integration with test cases or automated verification workflows.

EaseRequirements emphasizes traceability but does not provide native support for formal validation or verification steps. Approval processes and test linkage must be managed outside the tool.

TraceCloud allows users to assign validation states like "draft" or "approved" and supports collaborative input via comments. While mostly manual, it provides a basic structure for tracking requirement approval.

Jira, in its default form, offers limited validation functionality. Requirements can be reviewed and assigned statuses, but structured validation processes require additional customization or plugins. (*See Supplementary Table B4 of [19] for scoring details.*)

Notes:

- rmCloud shows changes but lacks version history.
- RMsis only uses a single-icon validation with limited feedback.
- EaseRequirements lacks approval workflow.

Management phase for analyzed tools in Jira Marketplace:

rmCloud enables links between Jira issues and requirements, with status tracking and field-level change visibility. However, it lacks full versioning support, which can be limiting in high-complexity projects requiring historical traceability.

RMsis offers strong change management capabilities, including full version history, dependency tracking, and an activity stream. This makes it well-suited for environments where requirement volatility and traceability are critical.

EaseRequirements supports collaborative editing and centralized visibility. While it doesn't offer native versioning, it leverages Jira's built-in tracking (e.g., activity logs), making it effective in teams already embedded in the Jira ecosystem.

TraceCloud allows status updates and version comparisons but lacks detailed change tracking or rollback capabilities, reducing its effectiveness in projects with frequent requirement evolution.

Jira offers basic support through issue status changes and activity logs, but dedicated requirement change management features must be configured or added via plugins. (*See Supplementary Table B5 of [19] for scoring details.*)

Notes:

- rmCloud change history limited.
- EaseRequirements relies on Jira for full audit trail.

Traceability phase for analyzed tools in Jira Marketplace:

rmCloud supports traceability via baselines and a traceability matrix that links requirements to Jira issues. While helpful for visibility, it lacks editable baselines and relies on manual updates for historical tracking.

RMsis provides strong end-to-end traceability. It includes a matrix for visualizing dependencies, supports both internal and external links, and maintains full version history making it well-suited for high-regulation or safety-critical environments.

EaseRequirements offers comprehensive traceability, including link matrices, version history, and detailed logs of user actions. Each requirement can reference artifacts and change data, enabling transparent audit trails and impact assessments.

TraceCloud uses a hierarchical folder structure and status tracking to maintain full forward and reverse traceability between requirements and related Jira items. Version control is preserved, supporting in-depth change analysis.

Jira, without extensions, provides basic traceability through issue linking and activity history. It requires configuration or plugins for deeper lifecycle documentation and trace relationships. (*See Supplementary Table B6 of [19] for scoring details.*)

6. A lifecycle-oriented method for selecting Jira requirement management tools based on functional priorities

As software projects grow in scope and complexity, selecting the right tools for managing requirements becomes increasingly critical. This paper, addresses this challenge by introducing a structured method for aligning tool selection with the functional priorities of a given project.

To support this aim, a decision-support tool was developed [20] (available in Latvian) to help users evaluate Jira Marketplace tools based on how well they fulfill core RM functions across the entire lifecycle. The tool builds on the RM functional dimensions proposed [16] and applies them specifically to Jira-compatible tools evaluated in this study.

The selection tool operates through a two-level evaluation repeated for each functionality within the RM lifecycle:

1. **Importance rating.** For every RM stage, the user rates its importance for the project on a scale from 1 (least important) to 5 (most important). This ensures that project-specific priorities are explicitly captured rather than assumed.
2. **Tool performance rating.** For the same functionality, the user then evaluates how well each of the analyzed Jira tool supports it by providing a description how the app provide this functionality. This is done on a scale from 1 (least effective) to 10 (most effective). The evaluation is repeated for all tools and all functionalities within the lifecycle stage.

This nested procedure is applied across all RM lifecycle stages. To ensure relevance, only RM functions that demonstrated at least minimal tool support (score ≥ 1) looked at in section Evaluation of Jira-Compatible Requirements Tools are included in the interface. This filtering reduces noise and allows the tool to focus only on features with real world application. Notably, partially supported functions (rated 1) are retained in the evaluation, as they still contribute significantly to user experience, configuration effort, and practical decision-making (see Figure 1).

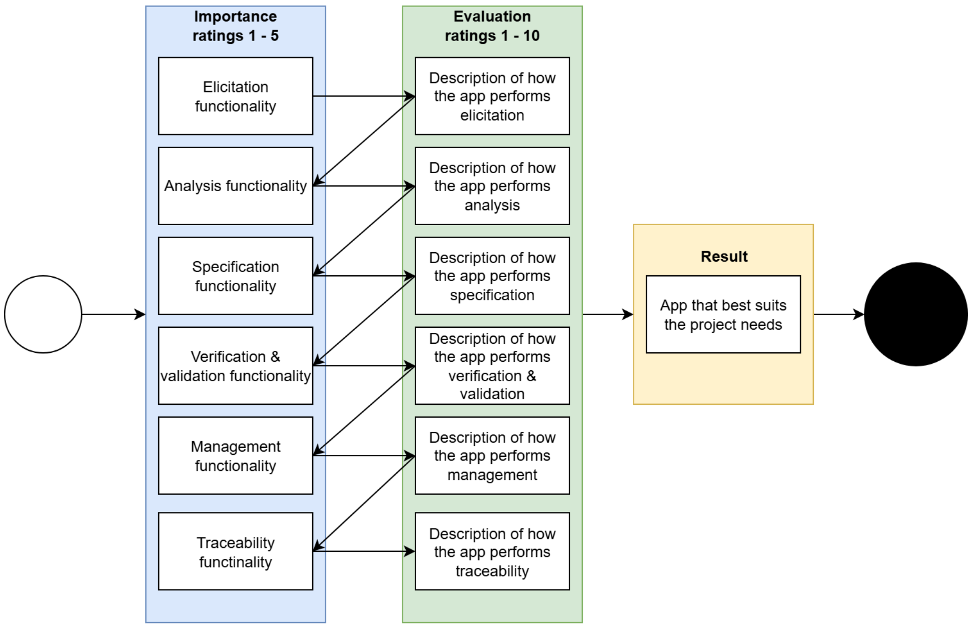
Importance ratings 1 - 5
Elicitation functionality
Analysis functionality
Specification functionality
Verification & validation functionality
Management functionality
Traceability functinality
Evaluation ratings 1 - 10
Description of how the app performs elicitation
Description of how the app performs analysis
Description of how the app performs specification
Description of how the app performs verification & validation
Description of how the app performs management
Description of how the app performs traceability
Result
App that best suits the project needs

Figure 1: Decision support tool's logic

The tool then aggregates results by multiplying the importance rating (1-5) with the performance rating (1-10) for each functionality–tool pair, as expressed in

$\text{Result}_t = \sum_{i=1}^{S} I_i \times R_{i,t}$,

where I_i denotes the importance assigned to functionality i, $R_{i,t}$ the rating of tool t for functionality i, and S the total number of evaluated functionalities.

This design ensures that the ranking reflects not only objective tool capabilities but also the relative importance of each functionality in a specific project context. To avoid bias, the tool is designed so that users do not know which Jira app they are rating during the evaluation. For each functionality, the interface presents descriptive text blocks in a fixed but anonymous order (internally corresponding to rmCloud, RMsis, EaseRequirements and TraceCloud). The user only sees the description and then provides a performance rating (1–10), without being informed which tool it belongs to. This logic is repeated identically for every lifecycle functionality and for every tool, ensuring that each app is assessed under the same conditions. All ratings are stored in structured form and combined with the user-defined importance weights.

Finally, the system generates a ranking in which the tool with the highest total score is presented as the most suitable option for the given project context. In addition, partial scores are provided per lifecycle stage to highlight relative strengths and weaknesses of each tool.

To assess the usability and effectiveness of the developed tool, two domain experts a business analyst and a senior project manager were invited to test it and provide feedback. Both regularly work with Jira and are experienced in RM. Overall, the experts found the tool intuitive, easy to navigate, and potentially valuable for supporting tool selection decisions.

The business analyst raised concerns about lengthy descriptions, which could reduce usability. In contrast, the senior project manager found the structure logical and user-friendly. The senior project manager viewed the tool's structure as logical, but suggested adding visual aids (e.g., screenshots or demo videos) to enhance clarity.

Both experts considered the tool useful in practice: the business analyst emphasized its suitability for smaller-scale projects, while the project manager highlighted its value for less experienced professionals in choosing appropriate RM tools. Suggestions for improvement included adding a comments section for qualitative feedback, providing more concise descriptions of tool functionality, and integrating visual examples.

The expert evaluations reflected differing preferences regarding the suitability of the tools assessed. One expert the business analyst indicated a preference for RMsis, while rmCloud was rated comparatively lower. The second expert the senior project manager favored EaseRequirements, with TraceCloud receiving the lowest overall score in their assessment. These contrasting results illustrate that tool effectiveness is not universally perceived and may depend on users' roles, experience, and specific project contexts. Both experts, however, emphasized the importance of foundational RM functionalities particularly the ability to define business, user, functional, and non-functional requirements, as well as maintaining traceability and enabling validation.

7. Conclusion

This paper, Structuring Complexity: Functional Evaluation of Jira Tools for Requirements Management, introduced a structured, lifecycle-oriented method for evaluating and selecting Jira-compatible RM tools based on their functional coverage. Using a standardized scoring model, the study assessed how Jira and four selected Marketplace applications support key RM functions across six lifecycle stages: elicitation, analysis, specification, validation, management, and traceability.

The evaluation revealed that neither native Jira nor any of the assessed tools provide full support across all RM lifecycle phases. While some tools performed well in specific areas such as traceability or specification none offered complete, out-of-the-box coverage for managing requirements in complex projects. This finding highlights the need for deliberate tool selection based on contextual priorities, supported by structured decision-making.

To address this, a decision-support tool was developed to help users align project-specific RM needs with available tool capabilities. Expert feedback confirmed the tool's utility while pointing to potential improvements, particularly in the areas of clarity, visual guidance, and result interpretation.

Future work should focus on expanding the evaluation framework to include a broader range of Jira-compatible tools, especially those that offer AI-assisted or automated RM functions.

Further development of the selection tool could include:

- Automating app data retrieval from the Jira Marketplace
- Incorporating dynamic warnings for functional limitations - in previous chapter ”Notes”
- Improving interface usability through integrated visual examples and context-aware recommendations

Additionally, applying this framework outside the Jira ecosystem could test its generalizability across platforms, while also highlighting how platform constraints shape RM tool functionality. Another important direction is exploring how Jira could be extended with modeling tools that support enterprise-level RE a functionality currently missing from all tools evaluated in this study, despite Jira’s widespread use in large-scale software projects.

Declaration on Generative AI

During the preparation of this work, the author used GPT-4o and DeepL for grammar checking, translation, and rephrasing. All AI-assisted outputs were carefully reviewed and revised by the author, who take full responsibility for the final version of the manuscript.

References

[1] C. Hokanson, K. Wiegers, Software requirements essentials: Core practices for successful business analysis, Addison-Wesley, Boston, MA, USA, 2024.

[2] P. A. Laplante, M. Kassab, Requirements engineering for software and systems, 4 ed., CRC Press, New York, 2022.

[3] Atlassian, Jira software, 2025. URL: https://www.atlassian.com/software/jira, [Online; accessed 20-July-2025].

[4] Atlassian, Rmcloud – requirements management, 2025. URL: https://marketplace.atlassian.com/apps/1217241/rmcloud-requirements-management?tab=overview&hosting=cloud, [Online; accessed 21-July-2025].

[5] Atlassian, Rmsis – requirements management for jira, 2025. URL: https://marketplace.atlassian.com/apps/30899/rmsis-requirements-management-for-jira?tab=overview&hosting=cloud, [Online; accessed 21-July-2025].

[6] Atlassian, Easerequirements – requirements management for jira (r4j), 2025. URL: https://marketplace.atlassian.com/apps/1213064/easerequirements-requirements-management-for-jira-r4j?tab=overview&hosting=cloud, [Online; accessed 21-July-2025].

[7] Atlassian, Tracecloud – projects & requirements, 2025. URL: https://marketplace.atlassian.com/apps/1231062/tracecloud-projects-requirements?tab=overview&hosting=cloud, [Online; accessed 21-July-2025].

[8] M. Hoffmann, N. Kuhn, M. Weber, M. Bittner, Requirements for requirements management tools, in: Proceedings of the 12th IEEE International Requirements Engineering Conference (RE 2004), IEEE, Kyoto, Japan, 2004, pp. 301–308. doi:10.1109/ICRE.2004.1335687.

[9] L. Fontoura, M. Otávio, Challenges in requirements engineering and its solutions: A systematic review, in: Proceedings of the International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), 2022, pp. 70–77. doi:10.5220/0011079000003179.

[10] D. Fucci, et al., Needs and challenges for a platform to support large-scale requirements engineering: A multiple case study, arXiv preprint arXiv:1808.02284 (2018). URL: https://arxiv.org/abs/1808.02284. doi:10.48550/arXiv.1808.02284.

[11] T. Kamal, Q. Zhang, M. A. Akbar, M. Shafiq, A. Gumaei, A. Alsanad, Identification and prioritization of agile requirements change management success factors in the domain of global software development, IEEE Access (2020). doi:10.1109/ACCESS.2020.2976723.

[12] M. Käpyaho, M. Kauppinen, Agile requirements engineering with prototyping: A case study, in: Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE 2015), IEEE, Ottawa, Canada, 2015, pp. 334–343. doi:10.1109/RE.2015.7320450.

[13] C. Lan, R. Balasubramaniam, Agile requirements engineering practices: An empirical study, IEEE Software 25 (2008) 60–67. doi:10.1109/MS.2008.1.

[14] L. Filion, N. Daviot, J.-P. Le Bel, M. Gagnon, Using atlassian tools for efficient requirements management: An industrial case study, in: Proceedings of the Annual IEEE International Systems Conference (SysCon), IEEE, Montreal, Canada, 2017, pp. 1–6. doi:10.1109/SYSCON.2017.7934769.

[15] M. Raatikainen, et al., Improved management of issue dependencies in issue trackers of large collaborative projects, IEEE Transactions on Software Engineering 49 (2023) 2128–2148. doi:10.1109/TSE.2022.3212166.

[16] J. Carrillo de Gea, R. Nicolás, J. Fernández-Alemán, A. Toval, C. Ebert, Requirements engineering tools: Capabilities, survey and assessment, Information and Software Technology 54 (2012) 1142–1157. doi:10.1016/j.infsof.2012.04.005.

[17] K. Zmitrowicz, Business analysis done right: Lessons learned and pitfalls avoided, 1 ed., Springer Nature, Cham, 2024. doi:10.1007/978-3-031-62194-9.

[18] A. Rozenberga, Table a – jira evaluation results, 2025. URL: https://ej.uz/AppendixA-ARozenberga, [Online; accessed 21-July-2025].

[19] A. Rozenberga, Table b – tool evaluation results, 2025. URL: https://ej.uz/AppendixB-ARozenberga, [Online; accessed 21-July-2025].

[20] A. Rozenberga, Jira rm tool selector (interactive web app), 2025. URL: https://script.google.com/macros/s/AKfycbzD6VeuRejGDWbOAoBOEkoF8T85QuhdL0tsvHxz3sab/dev, [Online; accessed 21-July-2025].