Computing and Informatics, Vol. 45, 2026, 346–364, doi: 10.31577/cai_2026_2_346

SCENARIO-BASED PRIORITISATION OF USER STORIES GROUNDED IN USAGE SCENARIOS

Maria Eckes-Kondak, Jacek Dajda

Faculty of Computer Science, AGH University of Krakow
al. Adama Mickiewicza 30
Krakow, Poland
e-mail: {meckes, dajda}@agh.edu.pl

Abstract. Prioritising user stories is a fundamental challenge in requirements engineering. It directly impacts project outcomes and is a key factor in agile success. Popular techniques, such as MoSCoW, rely on subjective and static criteria and do not expose dependencies, which can lead to suboptimal sprint and release plans. This article presents a scenario-based prioritisation technique grounded in identified usage scenarios. The technique considers contextual factors, dependencies, and the business impact of delivered user stories. We outline the concept, illustrate its mechanics on a case example, and compare results against classic prioritisation. Preliminary findings indicate that scenario-based prioritisation can serve as a valuable complementary tool for Product Owners when selecting requirements critical to end users. We observed limited overlap between scenario-based priorities and MoSCoW must-haves, with only modest alignment to value-based rankings. A brief survey also indicated a preference for a scenario-oriented MVP over a set of independent high-value features. We discuss implications for planning coherent increments and conclude with directions for further work, including larger-scale evaluations, real-world experiments, and dedicated tool support.

Keywords: Requirements engineering, user stories prioritisation, usage scenarios

1 INTRODUCTION

There is a strong need in the Information Technology (IT) industry to provide customers with valuable software in a short period of time to create a finished product that they are satisfied with. The Product Owner and software developers

are faced with the challenge of deciding which requirements should be included in the first release of the application, that is, to identify the requirements of high and low priority [1]. Prioritising the backlog is critical to the success of any project [2]. It affects the focus on project goals, strategic decision making, resource efficiency, and customer satisfaction.

The Product Owner communicates with the client to gather requirements from them, which most often take the form of user stories (US). A user story is a partially structured requirements specification that most often takes the form of [3, 4]:

- as [WHO],
- I want [WHAT],
- in order to [WHY].

It contains key elements of the requirements, such as the actor, his expectations of the system, and information about why the implementation of these elements is important. User stories describe functionalities that will be of value to the customer [5]. It is also very important to consider the user's requirements when formulating user stories, since it is the user who will ultimately use the software [6].

Prioritisation of user stories depends on several factors [7, 8]. Some of them may belong to the client side, and some of them may belong to the developer side [9]. What is certain is that they are clearly focused on creating business value for their client or users [10]. The literature does not describe a universal solution for estimating the value of user stories [11]. However, it may be thought that the user stories that are selected for the first sprints in the software development process may be considered more valuable to the client.

The MoSCoW (Must-have, Should-have, Could-have, and Would-have) method of prioritisation has gained popularity due to its simplicity [12]. However, during this process, there is often a situation where multiple user stories are equally important, and it is difficult to decide which one to do first.

Another issue is the question of the dependencies that exist between the various stories, which are important from the perspective of the manufacturing process and the usability of the final product. Unfortunately, these dependencies are not specified anywhere, and their inclusion is highly dependent on the awareness of participants in the planning process.

Thus, the introduction of dependencies seems justified, especially since they can result in a synergistic effect which is understood as the interaction of factors that is more beneficial than the sum of the effects of each factor individually [13].

The main objective of our research is to develop an improved user story prioritisation technique and verify its usefulness on a sample analysis by comparing the results obtained with the classic MoSCoW prioritisation method. The method assumes that by focusing on usage scenarios, we can obtain better results during user story prioritisation and faster delivery of valuable product features.

2 RELATED WORK

This section reviews existing research related to requirements engineering, user stories prioritisation, and the use of scenarios in software development. The overview highlights limitations of commonly used prioritisation techniques and motivates the need for approaches that better capture dependencies and end-to-end usage.

2.1 Requirements Engineering and User Stories

Requirements engineering aims to identify, analyse, and document stakeholder needs in a form that supports effective system development. Widely adopted techniques such as vision statements and high-level requirements descriptions help establish a shared understanding of project goals and scope [3, 14, 15].

In agile development, requirements are often expressed as user stories [9, 16, 17]. User stories provide lightweight, user-centred descriptions of functionality that emphasise value and intent rather than detailed specification. Their concise format improves communication between stakeholders and development teams and supports incremental refinement during development [3]. Because of these properties, user stories are frequently used as the primary planning units in agile projects [18].

2.2 User Story Prioritisation

Prioritisation of user stories is a critical activity in agile project planning, as it directly influences the order in which value is delivered to users and stakeholders. Numerous prioritisation techniques have been proposed, including MoSCoW, cumulative voting, and value-based ranking approaches [9, 19, 20].

The primary goal of prioritisation is to maximise business value and return on investment under constraints such as limited time, budget, and development capacity [21]. However, empirical studies indicate that many commonly used techniques rely heavily on subjective judgement and expert opinion [22]. As a result, prioritisation outcomes may vary significantly between stakeholders and lack transparency or repeatability.

Several studies report that value-based prioritisation alone does not always lead to optimal system evolution [21, 23]. Requirements that appear secondary from a business perspective may later prove critical for usability, operational continuity, or system coherence. These findings suggest that traditional prioritisation approaches may insufficiently account for dependencies and contextual relationships between user stories.

2.3 Scenario-Based Approaches in Requirements Engineering

Scenarios have long been used in requirements engineering to describe sequences of interactions between users and systems [24]. Typically associated with use cases,

scenarios capture concrete narratives of system use and help stakeholders reason about behaviour, context, and expectations [25].

Research distinguishes scenarios as a bridge between informal descriptions of real-world activities and more formal system models and specifications [26]. The Carroll taxonomy classifies scenarios according to their role in system development, ranging from early requirements exploration to design and evaluation support [27].

Because scenarios represent ordered sequences of events, they are well-suited to reasoning about process continuity and dependencies. When applied to software development planning, scenarios can reveal relationships between functionalities that are not visible when requirements are analysed in isolation. Organising user stories into scenario-based sequences therefore has the potential to improve coherence of delivered increments and reduce gaps in system functionality [28].

2.4 Research Gap

The reviewed literature demonstrates extensive work on both user story prioritisation and scenario-based requirements analysis. However, existing prioritisation techniques rarely integrate scenarios as a first-class construct for determining backlog order. In practice, prioritisation is often performed at the level of individual user stories without explicitly modelling their role within end-to-end usage flows.

This gap motivates the approach proposed in this study. By combining scenario-based structuring with systematic prioritisation, the method aims to address limitations of traditional techniques and support the delivery of cohesive, user-centred software increments.

3 THE SCENARIO-BASED PRIORITISATION TECHNIQUE

The proposed technique introduces usage scenarios as a primary structuring and prioritisation artifact during release and sprint planning. Instead of ranking user stories solely as independent backlog items, we first form coherent end-to-end flows and then derive user-story priorities from their role within those flows. The overall procedure is summarised in Figure 1.

3.1 Procedure

The planning procedure consists of the following steps:

1. Define high-level requirements, e.g., a vision statement.
2. Derive a comprehensive set of user stories that cover the intended system functionality.
3. Identify a set of usage scenarios from the high-level requirements.
4. Assign user stories to scenarios so that each scenario forms a logical sequence of activities. User stories not used by any scenario are treated as low-relevance candidates for later releases.

5. Prioritise scenarios (scenario set size is typically much smaller than the number of stories, which simplifies decision making around the MVP).
6. Compute a score for each user story based on
   (a) how often it appears across scenarios and
   (b) the priority of the scenarios in which it appears.
7. Sort the backlog according to the resulting user-story scores.

3.2 Rationale and Expected Effects

The procedure links requirements to user-perceived workflows. By placing user stories within end-to-end flows, the method makes dependencies explicit and supports a sensible build order: iterations are more likely to deliver cohesive and testable increments rather than disconnected features.

The technique can also be applied in the reverse direction: teams may begin with draft scenarios describing major system flows and then refine or derive user stories per scenario. In that case, a separate high-level requirements document may be reduced or merged with scenario descriptions, which can shorten the requirements discovery phase while keeping a clear focus on user journeys.

3.3 Story Recurrence Across Scenarios

A key property of the approach is that user stories may recur across multiple scenarios. Such recurrent stories often represent "flow-binding" functionality (e.g., scheduling, status visibility, or notifications) that is required to make several user journeys executable. In the proposed scoring, recurrence naturally increases priority, because omitting these stories would break multiple scenarios. Figure 2 illustrates an example mapping of user stories to scenarios, including recurring items.

In practice, this scenario-first view aligns with common practitioner artefacts such as user-flow diagrams: it captures who performs which actions, in what sequence, and with what transitions. Grouping stories into coherent flows and weighting them accordingly helps surface flow-critical stories early, reduces rework caused by missing dependencies, and supports more effective sprint planning. "Synergy" emerges when multiple stories are implemented together, enabling a complete user journey and providing more value than the same items delivered piecemeal.

4 EVALUATION DESIGN

The evaluation was conducted as a single case study of a kitesurfing school system, involving 112 user stories across three roles. Prioritisation was performed using MoSCoW, the 100 Dollar method and the proposed scenario-based approach. Expert-based (Delphi-style) analysis and a short student survey comparing MVP

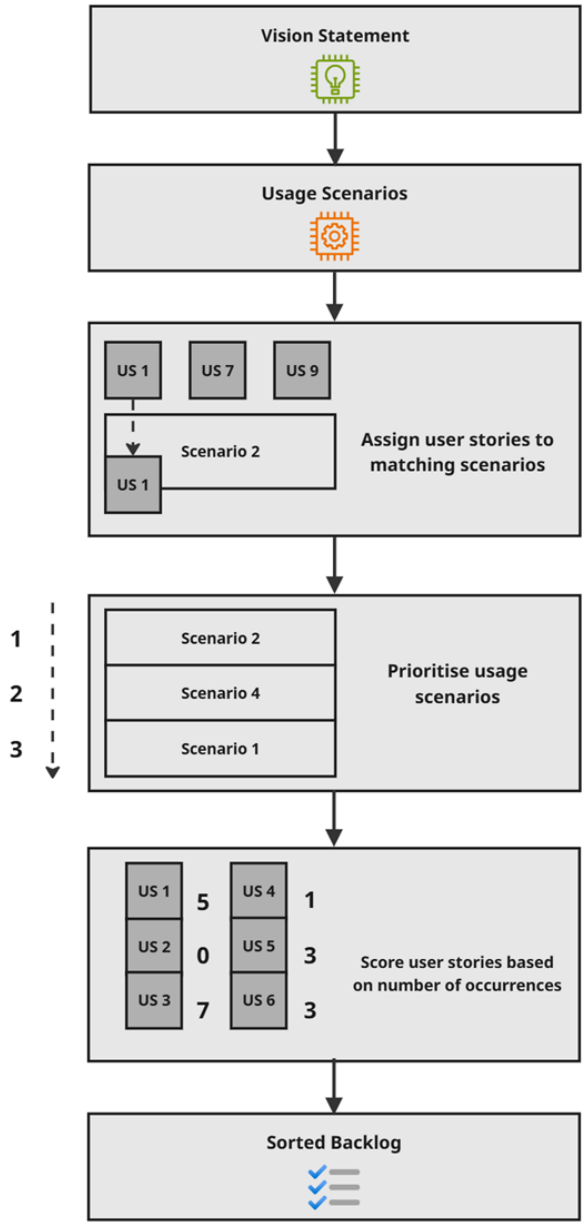

Figure 1. Sequence of the proposed scenario-based prioritisation technique

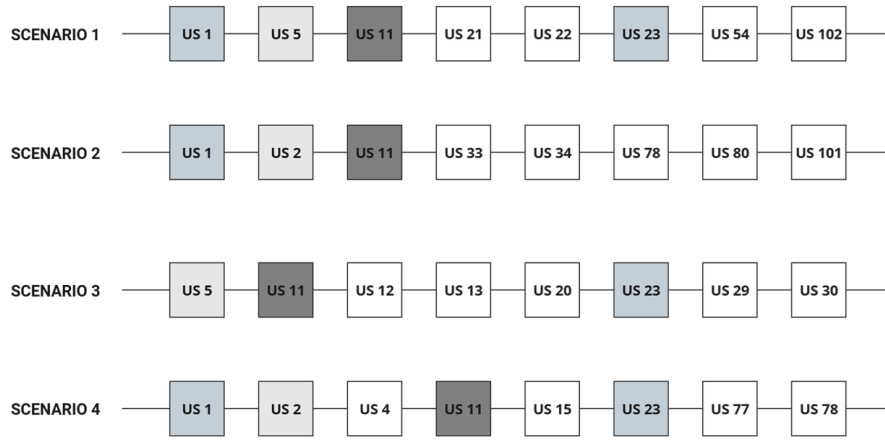

Figure 2. Example mapping of user stories to scenarios; some user stories recur across scenarios

variants were also included [29]. The evaluation was designed to examine whether the proposed scenario-based prioritisation technique offers practical advantages over traditional prioritisation methods in the context of agile software development. The study focuses on differences in requirement selection, ordering, and perceived value delivery when user stories are structured into coherent usage scenarios.

Since no existing benchmark or publicly available dataset currently enables a direct evaluation of scenario-based prioritisation approaches, the authors developed a dedicated evaluation framework that allows for a systematic comparison of the proposed method with a traditional prioritisation technique. The framework is based on a single, well-scoped use case. The evaluation procedure consisted of several consecutive stages, from defining a project vision to analysing prioritisation outcomes obtained using different techniques. An overview of the evaluation process is shown in Figure 3.

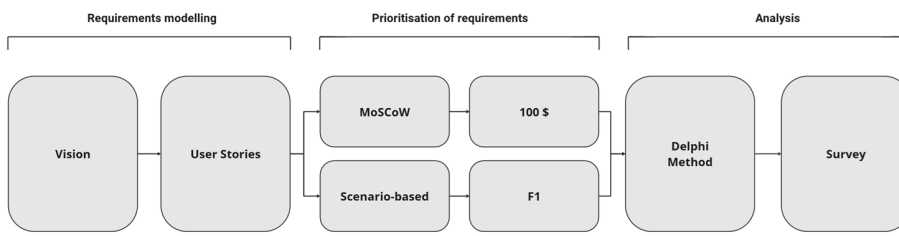

Figure 3. Design of the evaluation process

4.1 Project Vision

The first stage of the evaluation focused on defining a clear project vision for the selected software case. The vision document established the scope and goals of the system, identified key stakeholders, and outlined the main functional and non-functional requirements. It also described the primary system actors and their interactions with the system, providing a shared conceptual foundation for subsequent modelling activities.

The vision served two purposes. First, it ensured that all requirements-related decisions were grounded in explicit business and user goals. Second, it enabled the consistent derivation of user stories and usage scenarios that reflect the intended behaviour of the system.

4.2 Writing User Stories

Based on the project vision, a comprehensive set of user stories was created to represent the functional requirements of the system. The user stories were written from the perspectives of the identified system actors and collectively covered all major system functionalities.

In total, 112 user stories were defined. Some of them were actor-specific, while others were cross-cutting and applied to multiple roles. All user stories followed a standard agile format:

- As a [type of user], I want [a specific feature], so [benefit or value].

This consistent structure ensured clarity, comparability, and traceability between the vision, user stories, and subsequent prioritisation activities. The resulting backlog provided a sufficiently rich basis for evaluating differences between prioritisation techniques.

4.3 Prioritisation Using the MoSCoW Method

As a baseline, the complete set of user stories was prioritised using the MoSCoW method. Each story was classified into one of four priority categories according to its perceived business value:

- must-have,
- should-have,
- could-have,
- would-have.

To enable comparison with quantitative results from the proposed technique, the MoSCoW categories were mapped to numerical scores, with higher values indicating higher priority. This mapping allowed the ranked backlog to be analysed and compared against scenario-based rankings.

4.4 Refinement Using the 100 Dollar Technique

While the MoSCoW method provides a coarse-grained classification of requirements, it does not differentiate between user stories within the same priority group. To obtain a more fine-grained ordering, the 100 Dollar technique was applied as a refinement step.

In this approach, evaluators were given a hypothetical budget of 100 units to distribute among the user stories classified as must-have. The allocated values reflected the perceived relative importance of each story within this group. The resulting distribution enabled a ranked list of high-priority user stories to be produced, forming a value-oriented baseline for comparison.

4.5 Scenario-Based Prioritisation

In parallel, the same set of user stories was prioritised using the proposed scenario-based technique. First, a set of usage scenarios was defined, each representing a coherent end-to-end flow of system interaction. These scenarios were intended to correspond to meaningful functional increments that could be delivered within individual sprints.

User stories were then assigned to one or more scenarios, depending on their role in supporting specific workflows. Scenarios were prioritised according to their importance for delivering an initial Minimum Viable Product (MVP), and user stories inherited priority based on their occurrence and position within these scenarios.

This process resulted in a ranked backlog that explicitly reflects dependencies, process continuity, and functional cohesion rather than isolated business value alone.

4.6 Scoring Scenarios Using the F1-Inspired Method

To operationalise scenario priorities, an F1-inspired scoring scheme was used. Analogous to Formula 1 point allocation, scenarios with higher priority were assigned substantially higher scores than later scenarios. User stories inherited points based on the scenarios in which they appeared.

Stories that occurred in multiple high-priority scenarios accumulated higher scores, reflecting their importance for enabling several end-to-end workflows. User stories that did not appear in any scenario received zero points, indicating low relevance to early system increments.

This scoring mechanism emphasised the delivery of complete and usable workflows in early iterations and supported the rapid assembly of a coherent MVP.

4.7 Data Analysis

To analyse and compare prioritisation outcomes, expert judgement was applied using a Delphi-style consensus approach among the authors. This iterative process helped

reduce individual bias and ensure consistency in assigning priorities and scenarios mappings.

In addition, a short survey was conducted to complement the analytical comparison. The survey contrasted a scenario-oriented MVP with an alternative release composed of independent high-value features and captured participant preferences and perceptions.

Finally, the results obtained from MoSCoW, MoSCoW with the 100 Dollar technique, and the scenario-based method were compared. This analysis focused on overlap between ranked user stories, differences in priority distribution, and implications for planning coherent software increments.

5 RESULTS AND EVALUATION

This section presents the results obtained using the proposed scenario-based prioritisation technique and compares them with outcomes produced by traditional value-oriented approaches. The evaluation is conducted from two complementary perspectives. First, an analytical comparison examines differences in ranked user stories produced by scenario-based prioritisation and by the MoSCoW method refined with the 100 Dollar technique. Second, a short empirical survey explores perceived differences between a scenario-oriented MVP and a set of independent high-value features.

5.1 Scenario-Based Prioritisation Versus MoSCoW

The evaluation was conducted using a predefined vision of a kitesurfing school management system designed to support scheduling, communication, and administrative processes. Three primary system actors were identified: manager, instructor, and student. Based on this vision, 112 user stories were defined to cover the complete system functionality.

User stories were prioritised using two approaches:

- scenario-based prioritisation with F1-inspired scoring, and
- MoSCoW prioritisation refined using the 100 Dollar technique.

Seven usage scenarios were constructed, each representing a coherent end-to-end system flow. The first three scenarios were assumed to form the core of a Minimum Viable Product (MVP). User stories were assigned to one or more scenarios, which resulted in the recurrence of certain stories across multiple flows.

The scenario-based prioritisation produced a ranked backlog that differed substantially from the MoSCoW-based ranking. Although 32 user stories were classified as must-have using MoSCoW, only a subset of these appeared among the highest-ranked items in the scenario-based results.

Table 1 presents the highest-scoring user stories according to the scenario-based prioritisation. These stories primarily support scheduling, notifications, and coordi-

nation between system actors. Their high ranking is a direct result of their recurrence across multiple scenarios and their role in enabling complete and executable system flows.

User Story	Scenario Score	MoSCoW Priority
US 7	94	Must-have
US 10	88	Would-have
US 13	55	Must-have
US 25	52	Should-have
US 28	52	Should-have
US 55	52	Must-have
US 15	50	Could-have
US 20	50	Must-have

Table 1. Top-scoring user stories according to the scenario-based prioritisation

In contrast, MoSCoW prioritisation focuses on the perceived standalone business value of individual user stories. Dependencies and execution order are not explicitly modelled, which explains why certain flow-critical user stories appear in lower priority categories or are distributed unevenly across MoSCoW classes.

5.2 Comparison with MoSCoW Refined by the 100 Dollar Technique

To refine the value-based prioritisation, the 100 Dollar technique was applied to user stories classified as must-have. This method produced a more fine-grained ordering within the highest priority category.

Table 2 shows the ten highest-ranked user stories after applying the 100 Dollar refinement. These stories are predominantly related to access control, booking, and schedule visibility, reflecting stakeholder perception of direct business value.

User Story	Scenario Score	MoSCoW	100 Dollar Score
US 1	25	Must-have	9
US 2	25	Must-have	9
US 21	37	Must-have	8
US 46	37	Must-have	8
US 7	94	Must-have	5
US 12	37	Must-have	5
US 18	30	Must-have	5
US 19	30	Must-have	5
US 20	50	Must-have	5
US 80	40	Must-have	5

Table 2. Top-ranked must-have user stories refined using the 100 Dollar technique

Comparison of Tables 1 and 2 reveals limited overlap between the two approaches. Only two user stories (US 7 and US 20) appear among the top-ranked

items in both methods. These stories represent core functionalities that are simultaneously high in perceived business value and essential for multiple usage scenarios.

The remaining high-ranking scenario-based user stories received relatively lower scores in the 100 Dollar ranking, despite their importance for maintaining workflow continuity. This result indicates that scenario-based prioritisation surfaces a class of flow-binding requirements that are not easily identified through value-oriented assessment alone.

5.3 Summary of Analytical Comparison

Across all rankings, only a minority of user stories consistently appeared at the top of both prioritisation approaches. Approximately one quarter of the highest-ranked scenario-based user stories overlapped with the MoSCoW must-have set. This limited convergence confirms that the two techniques emphasise different notions of importance.

Scenario-based prioritisation highlights the structural role of user stories within end-to-end workflows, while MoSCoW and the 100 Dollar technique focus on stakeholder perceived business value. Rather than replacing traditional approaches, the proposed method provides a complementary prioritisation signal that explicitly accounts for dependencies and process continuity.

5.4 Evaluation Based on Survey Results

To complement the analytical comparison, a short survey was conducted among participants familiar with agile software development. Respondents were introduced to the kitesurfing school management system and presented with two alternative MVP variants:

- a scenario-oriented MVP composed of coherent user workflows, and
- a value-oriented MVP composed of independent high-value features.

The majority of respondents preferred the scenario-oriented variant for an initial release. Participants indicated that complete user flows provided a clearer perception of system readiness and usability than isolated features, even when the latter were perceived as individually valuable.

In addition, respondents rated flow-related user stories, such as scheduling updates and notifications, as more important when considered within scenarios than when evaluated independently. These findings support the assumption that the perceived value of the user is strongly influenced by the functional context and continuity.

5.5 Key Findings

The results of both analytical and empirical evaluation indicate that scenario-based prioritisation leads to a materially different backlog ordering than traditional methods. The main findings can be summarised as follows:

- Scenario-based prioritisation emphasises workflow completeness and dependency awareness.
- Overlap with value-based prioritisation is limited, showing that each method captures different aspects of importance.
- Flow-binding user stories gain priority even when their standalone business value is moderate.
- Survey participants preferred scenario-oriented MVPs in the early stages of development.

Together, these results provide initial evidence that integrating scenario thinking into requirements prioritisation can improve alignment between backlog ordering and user-perceived value.

6 DISCUSSION AND LIMITATIONS

The results presented in the previous section demonstrate that the proposed prioritisation approach based on usage scenarios leads to backlog orderings that differ substantially from those produced by traditional value-oriented methods. This section discusses the implications of these findings, highlights the strengths of the approach, and outlines its limitations.

6.1 Implications for Agile Planning

The main contribution of the proposed technique lies in its explicit focus on end-to-end usage flows. By organising user stories into coherent scenarios, the method shifts prioritisation from isolated functionality toward executable processes. This perspective aligns closely with how users experience software systems and how value is realised in practice.

The results suggest that scenario-based prioritisation is particularly useful during early planning phases, such as MVP definition and release planning. In these contexts, ensuring that core user journeys are complete and testable may be more important than delivering a collection of individually valuable but disconnected features. The approach therefore supports Product Owners in identifying flow-binding requirements that might otherwise be underestimated in value-based prioritisation.

Another important implication concerns dependency awareness. Scenario-based structuring makes dependencies between user stories explicit and actionable. This can reduce the risk of partial implementations, rework, and delayed value realisation,

especially in systems where user interactions span multiple roles and functional areas [30].

6.2 Complementarity with Existing Methods

The findings indicate that scenario-based prioritisation should not be viewed as a replacement for established techniques such as MoSCoW or the 100 Dollar method. Instead, it provides a complementary prioritisation signal that captures aspects not addressed by value-based approaches.

Although MoSCoW and related techniques reflect stakeholder perceptions of business importance, scenario-based prioritisation emphasizes structural importance within system workflows. Combining both perspectives can lead to more informed planning decisions, particularly when trade-offs must be made between business value and functional coherence.

In practice, the proposed technique may be most effective when applied after an initial value-based classification, serving as a refinement step that reorders high-priority items according to scenario relevance and dependency structure.

6.3 Limitations of the Study

Several limitations of the present study must be acknowledged. First, the evaluation was conducted using a single case study, which limits the generalisability of the findings. Although the selected system represents a realistic agile development scenario, the results may differ in other domains or project contexts.

Second, the construction of the scenario and the assignment of the user story relied on the expert judgment of the authors. Although a consensus-based approach was used to mitigate individual bias, some subjectivity remains inherent in the process.

Third, the empirical evaluation was based on a short survey with a limited number of participants. The survey captured perceptions rather than observed behaviour in real project settings. As a result, conclusions regarding the practical impact should be considered preliminary.

Finally, the proposed scoring mechanism assumes that earlier scenarios are inherently more valuable for MVP construction. While this assumption is reasonable in many agile contexts, alternative weighting schemes may be more appropriate in projects with different risk profiles or delivery strategies.

6.4 Threats to Validity

From a construct validity perspective, the study approximates value using MoSCoW and the 100 Dollar technique, and “flow importance” using scenario-based scores. Different operationalisations of these concepts could lead to different results.

Internal validity may be affected by the manual assignment of user stories to scenarios. This risk was mitigated through explicit rules and consensus, but cannot be fully eliminated.

External validity is limited by the use of a single case study and a non-random survey sample. Replication in industrial settings and across multiple domains is required to strengthen generalisability.

Regarding conclusion validity, the reliance on self-reported survey responses may not accurately reflect real planning decisions. Future studies should observe actual backlog prioritisation and development outcomes.

Despite these limitations, the results provide consistent indications that prioritisation based on usage scenarios captures aspects of requirement importance that are not addressed by traditional methods.

7 CONCLUSION AND FUTURE WORK

This paper addressed the problem of user story prioritisation in agile software development, with particular emphasis on limitations of value-oriented techniques that treat requirements as independent backlog items [31]. To overcome these limitations, a scenario-based prioritisation technique was proposed, in which user stories are structured and evaluated within coherent usage scenarios.

The main contribution of this work is the introduction of scenarios as a first-class prioritisation construct. By shifting the focus from isolated features to end-to-end workflows, the proposed approach captures dependencies, process continuity, and functional cohesion that is not explicitly represented in traditional methods such as MoSCoW or the 100 Dollar technique.

The analytical evaluation demonstrated that scenario-based prioritisation produces backlog orderings that differ substantially from value-based rankings. Only a limited overlap was observed between the highest-ranked user stories of the compared approaches, indicating that each method emphasises different dimensions of importance. In particular, the proposed technique consistently highlighted flow-binding user stories related to scheduling, coordination, and notifications.

The empirical survey further supported these findings by showing a clear preference for scenario-oriented MVPs over collections of independent high-value features. Respondents perceived coherent user flows as more valuable and indicative of system readiness in early development stages. Together, these results suggest that scenario-based prioritisation better aligns backlog ordering with user-perceived value and practical usability.

From a practical perspective, the proposed method can support Product Owners during MVP definition and release planning by helping them identify requirements that are critical for delivering complete and testable user journeys. Rather than replacing established prioritisation techniques, it complements them by providing an additional, workflow-oriented decision signal.

Future work should focus on validating the approach in broader industrial contexts and across different application domains. Larger-scale empirical studies involving professional development teams would help assess the impact of scenario-based prioritisation on development efficiency, rework reduction, and user satisfaction.

Another promising direction is the integration of the proposed technique with existing prioritisation frameworks, such as the Kano model, WSJF, or value-versus-complexity analysis, in order to explore hybrid approaches. In addition, alternative scenario weighting schemes could be investigated to adapt the method to projects with varying risk and delivery profiles.

Finally, the development of lightweight tool support, for example, as a plugin for agile project management platforms (e.g., Jira or Azure DevOps), could facilitate adoption in practice. Such integration could enable scenario-based grouping of user stories, automated scoring, and dependency visualization, supporting prioritisation and what-if analysis [32].

In summary, this study demonstrates that incorporating scenario thinking into requirements prioritisation offers a viable and valuable enhancement to agile planning practices, with the potential to improve both technical coherence and user-perceived value of early software increments.

Acknowledgements

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

REFERENCES

[1] Devadas, R.—Cholli, N. G.: Interdependency Aware QUBIT and BROWN-BOOST Rank for Large Scale Requirement Prioritization. International Journal of Computing and Digital Systems, Vol. 11, 2022, No. 1, pp. 625–634, doi: 10.12785/ijcds/110150.

[2] Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional, 2004.

[3] Wautelet, Y.—Heng, S.—Kolp, M.—Mirbel, I.: Unifying and Extending User Story Models. In: Jarke, M., Mylopoulos, J., Quix, C. et al. (Eds.): Advanced Information Systems Engineering (CAiSE 2014). Springer, Cham, Lecture Notes in Computer Science, Vol. 8484, 2014, pp. 211–225, doi: 10.1007/978-3-319-07881-6_15.

[4] Mulligan, P.: Specification of a Capability-Based IT Classification Framework. Information and Management, Vol. 39, 2002, No. 8, pp. 647–658, doi: 10.1016/S0378-7206(01)00117-3.

[5] Raharjana, I. K.—Harris, F.—Justitia, A.: Tool for Generating Behavior-Driven Development Test-Cases. Journal of Information Systems Engineering and Business Intelligence, Vol. 6, 2020, No. 1, pp. 27–36, doi: 10.20473/jisebi.6.1.27-36.

[6] Noel, R.—Riquelme, F.—Lean, R. M.—Merino, E.—Cechinel, C.—Barcelos, T. S.—Villarroel, R.—Munoz, R.: Exploring Collaborative Writing of User Stories with Multimodal Learning Analytics: A Case Study on a Software Engineering Course. IEEE Access, Vol. 6, 2018, pp. 67783–67798, doi: 10.1109/ACCESS.2018.2876801.

[7] Smith, R.—Azar, J.—Cordes, D.: A Value-Oriented Approach to Requirements Prioritization. Technical Report. University of Alabama, 2007.

[8] Wiegers, K. E.: First Things First: Prioritizing Requirements. 1999, http://www.tarrani.net/linda/prioritizing.pdf.

[9] Hudda, S.—Mahajan, R.—Chopra, S.: Prioritization of User Stories in Agile Environment. Indian Journal of Science and Technology, Vol. 9, 2016, No. 45, Art. No. 105069, doi: 10.17485/ijst/2016/v9i45/105069.

[10] Racheva, Z.—Daneva, M.—Sikkel, K.: Value Creation by Agile Projects: Methodology or Mystery? In: Bomarius, F., Oivo, M., Jaring, P., Abrahamsson, P. (Eds.): Product-Focused Software Process Improvement (PROFES 2009). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 32, 2009, pp. 141–155, doi: 10.1007/978-3-642-02152-7_12.

[11] Bass, J. M.—Haxby, A.: Tailoring Product Ownership in Large-Scale Agile. CoRR, 2018, doi: 10.48550/arXiv.1812.06524.

[12] Kravchenko, T.—Bogdanova, T.—Shevgunov, T.: Ranking Requirements Using MoSCoW Methodology in Practice. In: Silhavy, R. (Ed.): Cybernetics Perspectives in Systems (CSOC 2022). Springer, Cham, Lecture Notes in Networks and Systems, Vol. 503, 2022, pp. 188–199, doi: 10.1007/978-3-031-09073-8_18.

[13] Geary, N.: Understanding Synergy. American Journal of Physiology – Endocrinology and Metabolism, Vol. 304, 2013, No. 3, pp. E237–E253, doi: 10.1152/ajpendo.00308.2012.

[14] Bajraktari, A.—Binder, M.—Vogelsang, A.: Requirements Engineering for Research Software: A Vision. 2024 IEEE 32nd International Requirements Engineering Conference (RE), 2024, pp. 423–431, doi: 10.1109/RE59067.2024.00050.

[15] Wiegers, K. E.: Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Microsoft Press, 2003.

[16] Al-Msie'deen, R.—Blasi, A. H.—Alsuwaiket, M. A.: Constructing a Software Requirements Specification and Design for Electronic IT News Magazine System. International Journal of Advanced and Applied Sciences, Vol. 8, 2021, No. 11, pp. 104–118, doi: 10.21833/ijaas.2021.11.014.

[17] Blincoe, K.—Dehghan, A.—Salaou, A. D.—Neal, A.—Linaker, J.—Damian, D.: High-Level Software Requirements and Iteration Changes: A Predictive Model. Empirical Software Engineering, Vol. 24, 2019, No. 3, pp. 1610–1648, doi: 10.1007/s10664-018-9656-z.

[18] Kannan, V.—Basit, M. A.—Bajaj, P.—Carrington, A. R.—Donahue, I. B. et al.: User Stories as Lightweight Requirements for Agile Clinical Decision Support Development. Journal of the American Medical Informatics Association, Vol. 26, 2019, No. 11, pp. 1344–1354, doi: 10.1093/jamia/ocz123.

[19] Sibal, R.—Kaur, P.—Sharma, C.: Prioritization of User Story Acceptance Tests in Agile Software Development Using Meta-Heuristic Techniques and Comparative Analysis. In: Chakraverty, S., Goel, A., Misra, S. (Eds.): Towards Extensible and Adaptable Methods in Computing. Springer, Singapore, 2018, pp. 43–55, doi: 10.1007/978-981-13-2348-5_4.

[20] Saravana, K. M.—Basavaraj, G. N.—Rajkumar—Kovalan, A.: Case Study on Agile User Stories Prioritization Using Imaginative Standard. International Journal of Engineering Research and Applications, Vol. 2, 2012, No. 5, pp. 472–480, https://www.ijera.com/papers/Vol2-issue5/CD25472480.pdf.

[21] Ngo-The, A.—Ruhe, G.: Decision Support in Requirements Engineering. In: Aurum, A., Wohlin, C. (Eds.): Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg, 2005, pp. 267–286, doi: 10.1007/3-540-28244-0_12.

[22] Ali, A.—Hafeez, Y.—Hussain, S.—Yang, S.: Role of Requirement Prioritization Technique to Improve the Quality of Highly-Configurable Systems. IEEE Access, Vol. 8, 2020, pp. 27549–27561, doi: 10.1109/ACCESS.2020.2971382.

[23] Barney, S.—Aurum, A.—Wohlin, C.: A Product Management Challenge: Creating Software Product Value Through Requirements Selection. Journal of Systems Architecture, Vol. 54, 2008, No. 6, pp. 576–593, doi: 10.1016/j.sysarc.2007.12.004.

[24] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2001.

[25] Sutcliffe, A. G.—Maiden, N. A. M.—Minocha, S.—Manuel, D.: Supporting Scenario-Based Requirements Engineering. IEEE Transactions on Software Engineering, Vol. 24, 1998, No. 12, pp. 1072–1088, doi: 10.1109/32.738340.

[26] Sutcliffe, A. G.: Scenario-Based Requirements Analysis. Requirements Engineering, Vol. 3, 1998, No. 1, pp. 48–65, doi: 10.1007/BF02802920.

[27] Carroll, J. M.: Scenario-Based Design: Envisioning Work and Technology in System Development. Wiley, 1995.

[28] Sobiech, F.—Eilermann, B.—Rausch, A.: Using Synergies Between User Stories in Scrum. Lecture Notes on Software Engineering, Vol. 4, 2016, No. 2, pp. 91–94.

[29] Okoli, C.—Pawlowski, S. D.: The Delphi Method as a Research Tool: An Example, Design Considerations and Applications. Information and Management, Vol. 42, 2004, No. 1, pp. 15–29, doi: 10.1016/j.im.2003.11.002.

[30] Schmidt, R.—Lyytinen, K.—Keil, M.—Cule, P.: Identifying Software Project Risks: An International Delphi Study. Journal of Management Information Systems, Vol. 17, 2001, No. 4, pp. 5–36, doi: 10.1080/07421222.2001.11045662.

[31] Burkin, V.: Mitigating Risks in Software Development Through Effective Requirements Engineering. CoRR, 2023, doi: 10.48550/arXiv.2305.05800.

[32] Savić, D.—Vlajić, S.—Lazarević, S.—Antović, I.—Stanojević, V.—Milić, M.—da Silva, A. R.: Use Case Specification Using the SILABREQ Domain Specific Language. Computing and Informatics, Vol. 34, 2015, No. 4, pp. 877–910.

Maria Eckes-Kondak is Assistant Professor at the Faculty of Computer Science at the AGH University of Krakow, Poland. She holds her Ph.D. in management and her research focuses on project management, particularly in IT and research and development (R & D) projects. Her work emphasizes value-driven project management, stakeholder analysis, agile methodologies (such as Scrum), project risk management, organizational development, requirements engineering, UX, and AI.

Jacek Dajda is a research and teaching staff member at the Faculty of Computer Science at the AGH University of Krakow, Poland, specializing in software engineering, lightweight development methods (Agile/Scrum), and database and analytical system architectures. In his research activities, he combines academic knowledge with practical applications.