Al-imad Journal of Humanities and Applied Sciences (AJHAS)
مجلة العماد للعلوم الإنسانية والتطبيقية
ISSN: 3106-0323
Volume 2 – Issue 2 – 2026
Pages 228-241

A Framework for Improving Software Requirement Prioritization Using Hierarchical Cumulative Voting (HCV) and Best-Worst Method (BWM) in Agile Project

Salma A. Huwedi[1] *, Salwa M. Elakeili [2]
[1,2] Department of Software Engineering, Faculty of Information Technology, University of Benghazi, Libya
*Corresponding: Salma.huwedi@gmail.com

إطار عمل لتحسين ترتيب أولويات متطلبات البرمجيات باستخدام التصويت التراكمي الهرمي (*HCV*) وطريقة الأفضل والأسوأ (*BWM*) في المشاريع الرشيقة

سالمة أشرف هويدي [1] *، سلوى محمد العقيلي [2]
[2،1] قسم هندسة البرمجيات، كلية تقنية المعلومات، جامعة بنغازي، ليبيا

Received: 29-05-2026; Accepted: 19-06-2026; Published: 20-07-2026

Abstract:
Requirements prioritization is a fundamental activity in requirements engineering, as it directly impacts release planning, resource allocation, and project success. In agile software development (ASD), this process becomes more challenging due to evolving requirements, multiple stakeholders, limited scalability, and the high cognitive burden of traditional techniques. This study proposes a hybrid prioritization framework that combines the hierarchical cumulative voting (HCV), used to structure requirements and simplify stakeholder input, with the best-worst method (BWM), used to determine consistent expert weights with fewer pairwise comparisons. The stakeholder and expert rankings were integrated using reciprocal rank fusion (RRF) to produce a unified ranking. The framework was evaluated through a case study of an Organizational Quality Management System (OQMS), which included 17 requirements, 43 stakeholders, and 10 experts. The results showed that the approach was effective and reliable, demonstrating that the proposed framework reduced the effort required for requirements prioritization while maintaining a high level of judgment consistency and producing a reliable prioritization result. Overall, the framework demonstrated effectiveness and scalability for ASD while preserving the perspectives of stakeholders and experts.

Keywords: requirements prioritization, hierarchical cumulative voting, best-worst method, rank aggregation.

الملخص
تعتبر عملية ترتيب أولويات المتطلبات نشاطاً أساسياً في هندسة المتطلبات، نظراً لتأثيرها المباشر على تخطيط الإصدارات، وتخصيص الموارد، ونجاح المشاريع. في تطوير البرمجيات الرشيقة(*ASD*) ، تصبح هذه العملية أكثر تحدياً بسبب تطور المتطلبات، وتعدد أصحاب المصلحة، وقابلية التوسع المحدودة، والعبء المعرفي العالي للتقنيات التقليدية. يقترح هذا البحث إطار عمل هجين لترتيب الأولويات يجمع بين التصويت التراكمي الهرمي(*HCV*) ، المستخدم لهيكلة المتطلبات وتبسيط مدخلات أصحاب المصلحة، وطريقة الأفضل والأسوأ(*BWM*) ، المستخدمة لتحديد أوزان الخبراء المتسقة بعدد أقل من المقارنات الزوجية. تم دمج ترتيبات أصحاب المصلحة والخبراء باستخدام دمج الرتب التبادلية (*RRF*) لإنتاج ترتيب موحد. تم

تقييم الإطار من خلال دراسة حالة لنظام إدارة الجودة التنظيمية(*OQMS*) ، والذي تضمن 17 متطلباً، و43 صاحب مصلحة، و10 خبراء. أظهرت النتائج أن النهج كان فعالاً وموثوقاً، مما يدل على أن الإطار المقترح قلل الجهد اللازم لترتيب الأولويات مع الحفاظ على مستوى عالٍ من الاتساق في الأحكام وإنتاج نتائج موثوقة. بشكل عام، أظهر الإطار فعالية وقابلية للتوسع في المشاريع الرشيقة مع الحفاظ على وجهات نظر أصحاب المصلحة والخبراء.

الكلمات المفتاحية: ترتيب أولويات المتطلبات، التصويت التراكمي الهرمي، طريقة الأفضل والأسوأ، تجميع الرتب.

Introduction

Requirements engineering involves requirements elicitation, analysis, documentation, validation, and continuous management throughout the development lifecycle (Tasneem et al., 2025; Jawale & Bhole ,2015). Requirements prioritization is one of the most important of these activities, as it directly supports decision-making about which requirements should be implemented first under time, cost, and resource constraints. As systems become more complex, the number of requirements increases and stakeholders diversify, making prioritization more difficult, particularly in Agile development environments characterized by constantly changing user and business requirements. The most prominent challenges pointed out in research are limited scalability, the significant effort required to conduct pairwise comparisons, human bias in evaluation, and the difficulty of reprioritizing when requirements are added or modified (Jawale & Bhole, 2015; Eldrandaly, 2023). Although well-known methods such as the Analytic Hierarchy Process (AHP), Cumulative Voting (CV), MoSCoW, and the Kano model address different aspects of requirements prioritization, they generally suffer from high computational complexity, limited flexibility, and limited accuracy and scalability when applied to large projects (Tasneem et al., 2025). Therefore, there is a need for a balanced approach that combines ease of application, result accuracy, and the ability to support requirements prioritization efficiently.

Furthermore, as the number of requirements grows, reaching consensus on priorities becomes increasingly complex (Eldrandaly, 2023). However, most existing techniques treat prioritization as a one-time activity, which limits their ability to support ongoing changes in requirements and often forces organizations to repeat the entire prioritization process whenever modifications occur (Gupta et al., 2015). Consequently, agile development projects require a lightweight, practical, and reliable framework that supports continuous requirements prioritization.

Hierarchical Cumulative Voting (HCV) is an extension of the Cumulative Voting (CV) method, in which requirements are organized into a hierarchical structure; stakeholders first allocate a fixed number of points to groups of High-Level Requirements (HLRs) and then allocate a second fixed set of points to the Low-Level Requirements (LLRs) within each group (Riņķevičs and Torkar, 2013; Berander et al., 2009). This hierarchical decomposition improves scalability compared with the traditional flat CV structure, while maintaining its quantitative nature based on stakeholder assessments, making HCV suitable for large sets of requirements and for ASD environments (Riņķevičs and Torkar, 2013).

The Best–Worst Method (BWM) is one of the modern methods in Multi-Criteria Decision Making (MCDM), in which element weights are derived by conducting pairwise comparisons against the best and worst elements, rather than performing all pairwise comparisons as in AHP (Rezaei, 2015). This method reduces the number of required comparisons from $n(n-1)/2$ to $2n-3$, reduces cognitive burden, avoids fractional judgments, and yields a built-in consistency measure. As a result, BWM was reported to provide higher reliability than AHP with less effort (Rezaei, 2015).

This study proposes a hybrid framework that combines HCV and BWM to improve

requirements prioritization in agile projects. This proposed framework aims to reduce the complexity of the prioritization process, improve judgment consistency and prioritization accuracy through a structured weighting procedure for determining weights, and better represent stakeholder preferences in large-scale projects.
The remainder of this paper is organized as follows: Section II presents related work. Section III describes the proposed methodology. Section IV explains the case study. Section V gives results and discussion. Finally, Section VI concludes the paper.

Literature Review

As indicated by Alhenawi et al. (2024), requirements prioritization techniques can be classified into relative techniques, for example, MoSCoW, the Planning Game, and Numerical Assignment (NA), which are straightforward to implement and involve low implementation effort, but generally provide lower prioritization accuracy. On the other hand, exact techniques, such as the Analytic Hierarchy Process (AHP), Cumulative Voting (CV), Hierarchical Cumulative Voting (HCV), and the Wiegers method, are commonly used approaches that produce quantitative results supported by mathematical models but involve greater computational and assessment efforts and may provide limited flexibility in rapidly changing environments.
Furthermore, previous comparative studies have shown that hybrid and fuzzy approaches generally outperform individual traditional techniques in terms of accuracy, handling conflicts, and flexibility, while a key challenge is maintaining a balance between mathematical rigor and applicability in practice (Ais et al., 2026; Bibi et al., 2025).

A. Application of HCV for requirements prioritization

HCV was developed as an extension of the traditional CV technique to address the scalability issues in requirements prioritization by structuring requirements hierarchically.
(Mohamed et al., 2008) proposed a Value-Oriented by Hierarchical Cumulative Voting (VOHCV) framework through integration of HCV and Value-Oriented prioritization (VOP) to support software product management in market-driven environments. The framework used stakeholder value assessments to select appropriate requirements for incremental software releases. The study demonstrated that the method based on HCV to help release planning decisions; nevertheless, it mainly focused on evaluating the proposed technique and did not address challenges associated with changing requirements or determine weights for different stakeholders.
(Riņķevičs and Torkar, 2013) reviewed the literature systematically to examine the empirical use of CV and HCV techniques in software engineering. The study presented the Equality of Cumulative Votes (ECVs) framework, based on Compositional Data Analysis (CoDA), to enhance the interpretation of cumulative voting results. The review indicated the effectiveness of hierarchical voting methods while also pointed out several limitations in previous research, including limited evaluation on large sets of requirements and a lack of publicly available datasets from empirical studies.
(Awais, 2016) proposed an improved framework that integrated HCV into the task phase of the Entry Task Validation and Exit (ETVX) model to establish a structured process for prioritizing requirements. The framework organized requirements into a hierarchical structure, enabling stakeholder to distribute their votes to be distributed across different levels thereby reduced the effort needed for prioritization compared with traditional prioritization techniques. However, the framework did not use advanced mathematical techniques to estimate consistent weights for stakeholders or experts.
(Jawale et al., 2017) extended the Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV) methodology and experimentally evaluated its ability to support requirements prioritization

under changing requirements. The mechanism supported changes in requirements by allowing requirements to be added, requirements to be reorganized, and priorities to be revised when needed. Although the approach improved flexibility, the evaluation relied primarily on simulation and did not address quantitative determination of expert's weights or measurement of consistency.

B. Application of BWM for Weight Determination

Since studies applying BWM directly to software requirements prioritization remain scarce, related MCDM studies from other domains were also considered.

(Norouzi & Namin, 2019) proposed a hybrid model that combines Fuzzy BWM and Fuzzy TOPSIS to prioritize risk factors in large-scale projects. The model used Fuzzy BWM to derive reliable criterion weights for project success criteria and demonstrated the effectiveness in handling uncertainty while reducing the complexity of pairwise comparisons. However, the evaluation was conducted only in the infrastructure domain, making it difficult to generalize the results to other domains.

(Olawore et al., 2023) proposed a framework that integrated Fuzzy BWM and Fuzzy AHP to prioritize success factors for software technology marketing in higher education institutions. The findings indicated that Fuzzy BWM involved fewer pairwise comparisons and provided more consistent judgments when compared with Fuzzy AHP. However, the framework focused on the marketing of software technologies rather than on prioritizing software requirements, and it relied on a limited number of experts.

(Trivedi et al., 2024) developed a hybrid multi-criteria decision-making framework that combined BWM, TOPSIS, and Simple Additive Weighting (SAW) to prioritize traffic safety improvements. BWM was used to derive criterion weights based on expert judgments, while TOPSIS and SAW were used to rank alternatives. The study showed that integrating BWM with other MCDM techniques improved the reliability of the results while maintaining judgment consistency of the prioritization results. However, the framework assumed equal weights for all experts, and it was evaluated outside the software engineering domain.

Aljohani et al. (2025) proposed a hybrid framework that integrates large language models (LLMs) and Fuzzy BWM to support the prioritization of success factors for agile requirements change management (ARCM) in global software development (GSD) environments. The framework demonstrated a high degree of agreement between the ranking results produced by LLMs and expert evaluations. However, the study was limited to a small number of experts and a single language model, which limited the applicability of the findings. Additionally, the framework focused on prioritizing success factors rather than software requirements.

The review showed that existing techniques in software requirements prioritization still face several challenges. While HCV-based methods have improved scalability and supported the hierarchical organization of requirements, they generally lack a rigorous analytical mechanism for deriving consistent weights for stakeholders or experts. In contrast, BWM reduced pairwise comparison effort and enhanced the consistency among expert judgments in different decision-making areas; although the adoption of these techniques for requirements prioritization in software engineering is still limited. Furthermore, no previous studies had combined HCV and the BWM to provide a unified framework that simultaneously addresses scalability, consistency of judgments, and analytical weight determination in ASD.

Based on the limitations found in previous studies, this paper proposes an HCV–BWM hybrid framework for software requirements prioritization in ASD. The proposed framework used HCV to organize and prioritize requirements with stakeholder involvement, while BWM was applied to obtain weights from experts' judgments. Through integration of both techniques, the framework was designed to enhance the consistency of requirements prioritization decisions and reduce the effort required for pairwise comparisons.

Research Methodology

The proposed framework was extended from a published case study (Al-Rawashdeh et al., 2025) and it is based on a quantitative, design-based methodology, which integrated HCV and BWM within a hierarchical requirements structure, followed by the Reciprocal Rank Fusion to integrate the resulting rankings and an incremental reprioritization mechanism (Fig. 1). HCV was used to efficiently collect stakeholder input, while BWM was used to assign weights based on expert assessments that are characterized by a higher degree of accuracy and mathematical analysis. The two methods were combined to balance the ease of data collection with prioritization accuracy.

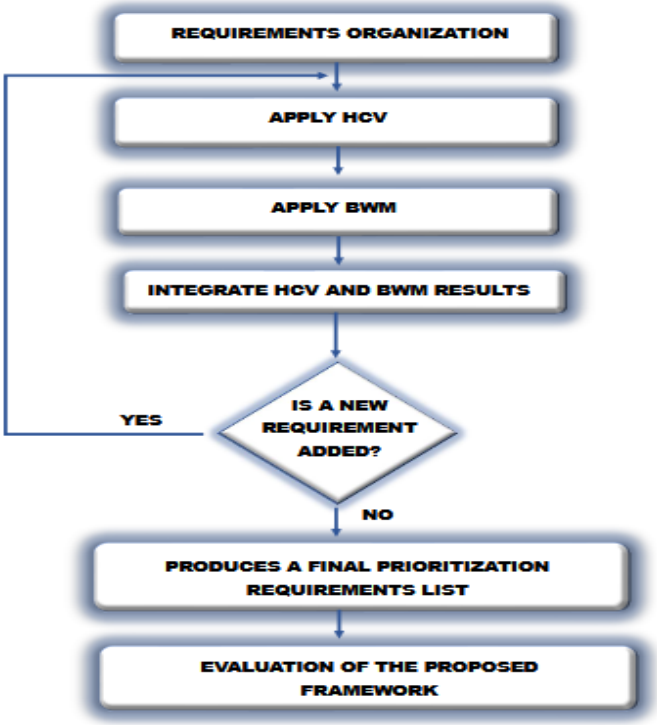

Fig. 1 The proposed framework

A. Requirements Organization

The requirements were developed, evaluated and reorganized into groups based on functional similarity. The resulting structure consisted of two levels: High-Level Requirements, (HLRs) which were functional groups, and Low-Level Requirements (LLRs) which were individual requirements and based on the principle of functional decomposition (Santos et al., 2016). The hierarchical structure was illustrated in Fig. 2 according to HCV technique, which organizes decision-making elements within multiple hierarchical levels to facilitate systematic evaluation and prioritization across different levels of the hierarchical structure. This hierarchical organization helped clarify the relationships between requirements and reduced the cognitive burden associated with subsequent evaluation processes.

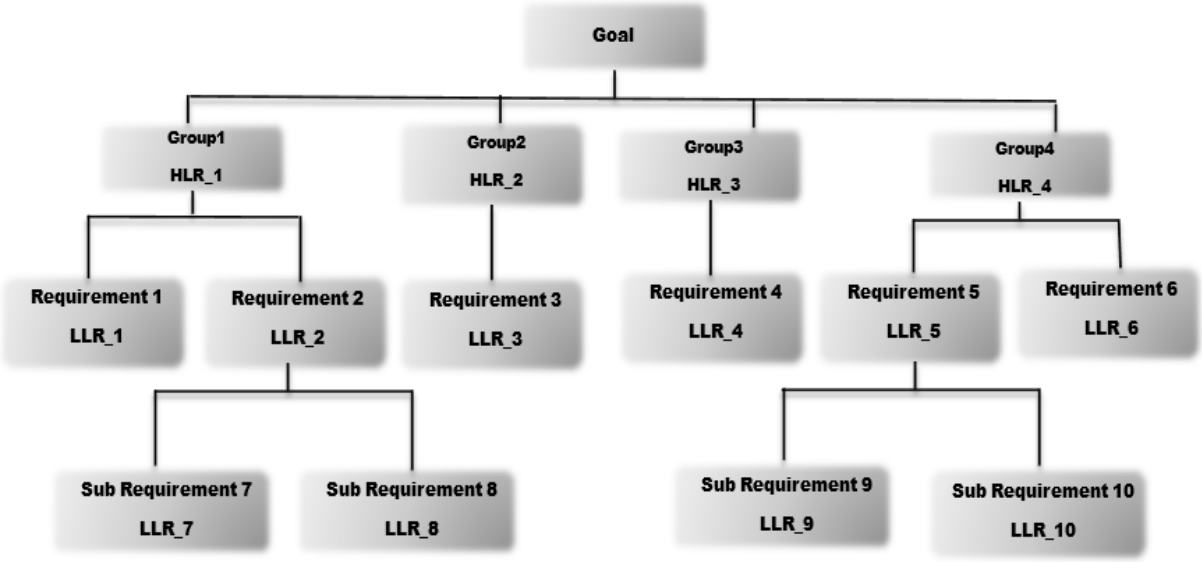

Fig. 2 Hierarchical structure of requirements

B. Application of HCV

The voting process consisted of two stages: participants first distributed 100 points among the HLRs according to perceived importance, and then distributed 100 points among the LLRs within each HLR independently (Akbar et al., 2021). Participants were not assigned the same weights; each participant was assigned a weight based on their expertise, which was then normalized so that the sum of the weights equals one. Following the procedure described in (Santos et al., 2016), priorities were calculated in three steps. For each LLR x, the intermediate priority was calculated as shown in (1):

$$I_x = A_x . H_x . Q_x \qquad (1)$$

where A_x is the score assigned to requirement x, H_x is the score of the HLR to which it belongs, and Q_x is the number of LLRs within that HLR. Intermediate priority values were normalized as illustrated in (2):

$$F_x = \frac{I_x}{\sum_{i=1}^{n} I_i} \qquad (2)$$

The final priority for requirement x, calculated according to HCV was determined by aggregating the weighted contributions of all n participants as indicated in (3):

$$P_x = \sum_{i=1}^{n} W_i \cdot F_{x,i} \qquad (3)$$

This procedure produced the final HCV priorities on a ratio scale.

C. Application of BWM

BWM was applied within the hierarchical structure resulting from the HCV phase to determine the relative importance of software requirements. Unlike the traditional single-level application, BWM was applied at two hierarchical levels. At the first level, the weights of the HLR groups were calculated; then, the method was applied independently within each group to determine the local weights of LLRs. This hierarchical approach improved scalability while maintaining a structured and consistent prioritization process.

At each decision-making level, each expert identified the best and worst elements, then constructed the Best-to-Others (B2O) A_B=($a_{B1}, a_{B2}, \ldots\ldots, a_{Bn}$) and Others-to-Worst (O2W)

A_W=($a_{1W}, a_{W2}, \ldots\ldots, a_{nW}$) comparison vectors using a scale from 1 to 9 in accordance with the standard BWM procedure (Rezaei, 2015). Next, the optimal weights were calculated by solving the following optimization model:

$$min\ \ \xi$$
$$\left| \frac{W_B}{W_j} - a_{Bj} \right| \leq \xi\ , \left| \frac{W_j}{W_w} - a_{jw} \right| \leq \xi$$
$$\sum_{i=1}^{n} w_i = 1\ , \qquad w_j \geq 0\ , \forall j$$

Where $\boldsymbol{W_B}$, $\boldsymbol{W_w}$, and $\boldsymbol{W_j}$ represented the optimal weights of the best element, worst element, and element $\boldsymbol{j}$, respectively. The parameters $\boldsymbol{a_{Bj}}$ and $\boldsymbol{a_{jW}}$ represented the preference of the best element over element $\boldsymbol{j}$ and the preference of element $\boldsymbol{j}$ over the worst element. The variable $\boldsymbol{\xi}$ represents the maximum deviation, which was used as an indicator of consistency.
The optimization model was solved using Excel Solver to obtain the normalized weights at each hierarchical level.
After calculating the HLRs weights and LLRs weights, the final weight for each requirement was calculated by combining its local weight with the weight of its corresponding HLR group, as shown in (4):

$$W_{BWM}(r_j) = W_{HLR}(G_i) \times W_{LLR}(r_{ij}) \qquad (4)$$

Where $\boldsymbol{W_{BWM}(r_j)}$ is the final weight of requirement $\boldsymbol{j}$; $\boldsymbol{W_{HLR}(G_i)}$ is the weight of the group to which requirement r_{ij} belongs and $\boldsymbol{W_{LLR}(r_{ij})}$ is the local weight of the requirement within its corresponding group
Finally, all requirements were sorted in descending order according to their final weights to produce a BWM-based priority ranking.

D. Integration of HCV and BWM

The rankings generated by the HCV and BWM methods were combined to obtain a unified ranking of software requirement priorities. This aggregation aimed to combine the stakeholder perspectives derived from HCV with the expert judgments resulting from BWM. The Reciprocal Rank Fusion (RRF) algorithm was adopted to aggregate the ranking lists. The algorithm also assigned higher scores to top-ranked requirements while reducing the influence of lower-ranked requirements (Cormack et al., 2009; Wang et al., 2024).

The final RRF score for each requirement was calculated as shown in (5):

$$S_{RRF}(u_i) = \sum_{R^t \in R} \frac{1}{R^t(u_i) + K} \qquad (5)$$

where $S_{RRF}(u_i)$ represents the final RRF score of requirement u_i obtained from the RRF algorithm; R represented the set of ranking lists used in the aggregation process; $R^t(u_i)$ represented the rank obtained by requirement u_i in the ranking list $R^t$and K represented a constant value used to control the influence of higher-ranked elements. Based on previous studies (Cormack et al., 2009; Wang et al., 2024), the standard value K= 60 was adopted to ensure the stability of the ranking aggregation process.
To ensure that the selection of K does not introduce bias into the results, given the relatively

small number of requirements in this case study, the RRF scores were recomputed using K = 17 (approximately equal to the number of requirements) and then compared with the results produced with the standard value K = 60. The rankings obtained were identical, which verifies that the final integrated ranking was robust with respect to the choice of K in this setting.
Finally, the requirements were sorted in descending order according to their RRF scores to obtain the final combined ranking of requirement priorities.

E. Incremental Addition of New Requirements

When a new LLR is added, domain experts assign it to the most appropriate HLR group. Next, only the affected group of requirements was reevaluated using HCV and BWM and the updated ranking of requirements within that group is re-aggregated again using Equation (5). The rankings of unaffected groups are preserved, eliminating the need to repeat the entire prioritization process and reducing evaluation effort.

F. Evaluation Metrics

1. Comparison Effort Reduction

Framework efficiency was evaluated based on the reduction in the number of required pairwise comparisons relative to the baseline AHP-based approach and was calculated as follows (Rezaei, 2015):

For AHP:

$$C = \frac{n(n-1)}{2} \qquad (6)$$

For BWM:

$$C = 2n - 3 \qquad (7)$$

Where C represents the total number of comparisons required by each method, and n represents the number of requirements at the considered level.

2.Consistency assessment

The consistency of the experts' judgments was assessed using the input-based consistency ratio. The input-based consistency ratio CR^I proposed by (Liang et al., 2020) was calculated based on the input inconsistency indicator ξ^{input}

$$\xi^{input} = max_j \left| \frac{a_{Bj} \times a_{jw} - a_{BW}}{a_{BW}} \right|$$

where a_{Bj} denoted the degree of preference of the best requirement over requirement j, a_{jw} represented the degree of preference of requirement j over the worst requirement, and a_{BW} denoted the direct comparison between the best and worst requirement

The calculated CR^I value was compared to the reference thresholds given in (Liang et al., 2020), which depend jointly on the number of elements n and the direct comparison between the best and worst a_{BW} . Judgments were considered consistent when $CR^I \leq$ the threshold; otherwise, the judgments were revised before calculating the weights. This process was performed automatically using the BWM Excel Framework (v5).

3. Stakeholder and Expert Participation

Stakeholder and expert participation were also evaluated by comparing the number of participants and their distribution across the two participant groups (HCV stakeholders and BWM experts) with the baseline study (Al-Rawashdeh et al., 2025).

4. Rank Correlation Analysis

The degree of agreement between the proposed framework ranking and the rankings based on AHP was measured using Spearman's rank correlation coefficient ρ, which was calculated using Pearson's formula indicated in (8) and applied to the rank data because tied ranks existed (Burgund et al., 2023):

$$\rho = \frac{\sum_{i=1}^{n}(R_{xi} - \overline{R_x})(R_{yi} - \overline{R_y})}{\sqrt{\sum_{i=1}^{n}(R_{xi} - \overline{R_x})^2}\sqrt{\sum_{i=1}^{n}(R_{yi} - \overline{R_y})^2}} \quad (8)$$

$\boldsymbol{R_{xi}}$ represented the rank of requirement *i* in the first ranking; $\boldsymbol{R_{yi}}$ represented the rank of requirement *i* in the second ranking; $\overline{\boldsymbol{R_x}}$ represents the mean rank in the first ranking; $\overline{\boldsymbol{R_y}}$ represented the mean rank in the second ranking and **n** represented the number of ranked requirements.

The coefficient ranges from -1 to 1. Values closer to 1 indicated a strong positive correlation between the two rankings, while values close to 0 indicated a weak or no correlation, and values close to -1 indicated a strong negative correlation.

This analysis was used to determine whether the proposed hybrid framework generated prioritization results consistent with the baseline AHP-based approach while benefiting from the advantages of both HCV and BWM.

Case Study

To demonstrate the applicability of the proposed HCV-BWM framework, the Organizational Quality Management System (OQMS) was selected as the case study. The requirements used in this case study were adopted from (Al-Rawashdeh et al., 2025), who elicited and identified the requirements using the Goal-Oriented Requirements Elicitation Process (GOREP). The case study was conducted using 17 software requirements of which 13 were functional requirements (FRs) and four were non-functional requirements (NFRs).

Although the original study sorted these requirements into functional and non-functional categories, that categorization does not capture a hierarchical or thematic structure appropriate for HCV implementation. Consequently, the requirements were reorganized into five functionally coherent groups according to their domain characteristics, independent of their functional/non-functional classification, so as to support meaningful hierarchical decomposition in line with the HCV technique.

To support the hierarchical prioritization process, the requirements were reorganized into five requirement groups based on their characteristics without modifying the original requirements: System Access and Data Management, Governance and Strategic Management, Academic and Institutional Operations, Performance Monitoring and Reporting, and Quality Attributes. This hierarchical decomposition simplified the assessment process and allowed HCV and BWM to be applied at different levels of the hierarchy.

The resulting three-level hierarchy comprised the overall prioritization goal, five requirement groups, and their associated requirements as illustrated in Fig. 3.

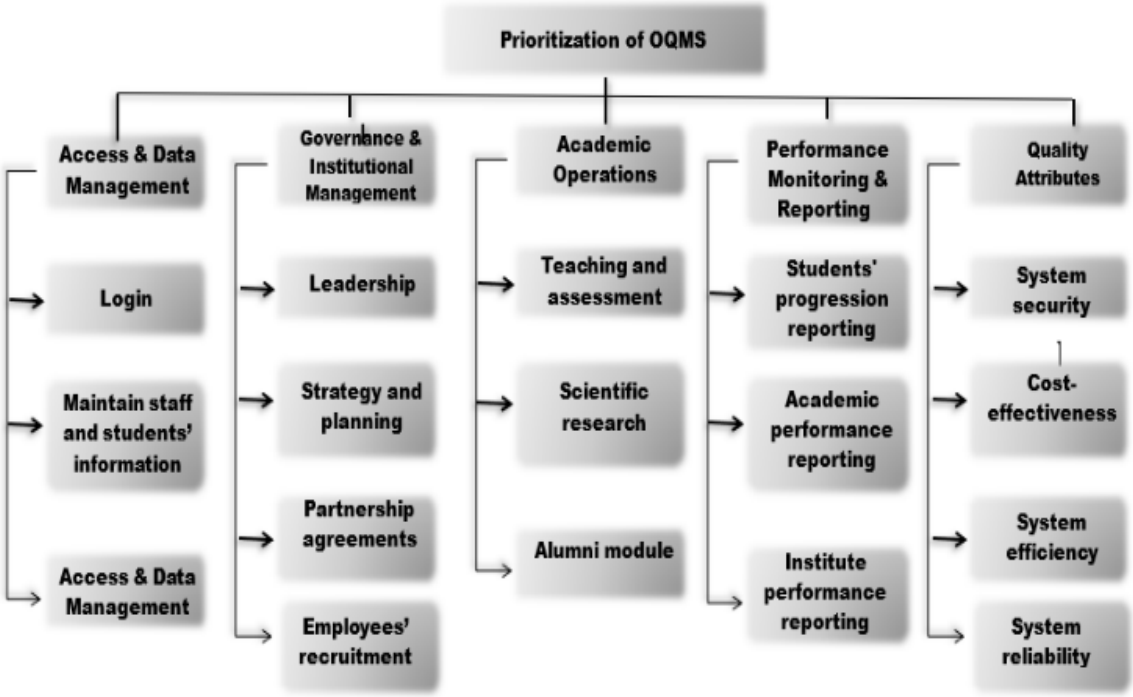

Fig. 3 Hierarchical decomposition of OQMS requirements

The HCV phase involved 43 stakeholders, including educators, developers, quality professionals, graduates and postgraduate students, who distributed a total of 100 points (through online or paper-based questionnaires) hierarchically across requirement groups and LLRs. The stakeholder responses were processed using the weighting and aggregation procedures described in Section III-B resulting in the final stakeholder-based priorities.

The BWM phase involved 10 experts with relevant domain expertise (selected from 15 initial experts after excluding experts with inconsistent judgments). All experts had more than three years of professional experience and conducted pairwise comparisons at both hierarchical levels using the 1-9 preference scale defined in the BWM technique. Subsequently, the individual weight vectors of the experts were aggregated using the geometric mean to obtain the aggregated HLR and LLR weights. The final weight of each requirement was calculated by combing LLR weight with corresponding HLR weight using Equation (4). The resulting weights were subsequently normalized to obtain the final BWM-based requirement priorities. Finally, the rankings generated by the HCV and BWM were combined using the RRF algorithm, as defined in Equation (5), to produce the final requirement priority ranking. The quantitative evaluation and comparative analysis of the results are presented in the following section.

Results And Discussion

The proposed framework was evaluated using the OQMS case study to assess its effectiveness in prioritizing software requirements. The final ranking resulted from the integration of stakeholder preferences collected using HCV with expert judgments obtained through BWM. Table (1) summarized the highest-ranked requirements obtained after applying the RRF algorithm.

Table (1). Top Ten Requirements Based on The Final Integrated Ranking

Requirement ID	Requirements	HCV Rank	BWM Rank	RRF	Final Rank
FR8	Teaching and assessment module	1	1	0.0328	1
FR9	Scientific research module	3	3	0.03175	2
NFR1	System security	4	4	0.0313	3
NFR4	System reliability	8	2	0.03083	4
FR4	Leadership	5	6	0.03054	5.5
NFR3	System efficiency	6	5	0.0305	5.5
FR1	Login module	2	12	0.0300	7
FR3	Maintain faculties and program information	7	7	0.0299	8
FR5	Strategy and planning module	10	9	0.0288	9
FR2	Maintain staff and students' information	9	11	0.0286	10

The integrated ranking showed a moderate level of agreement among stakeholders and experts regarding the highest-priority requirements, while balancing conflicting priorities. Requirement FR8 (Teaching and Assessment) remained the highest priority in both HCV and BWM, followed by FR9 (Scientific Research) and NFR1 (System Security), reflecting the importance of core educational processes, research functions and system protection.

Requirements that showed notable differences, such as FR1 (Login), were assigned intermediate ranks after the integration process, demonstrating the RRF algorithm's ability to reconcile different perspectives without favoring either stakeholders or experts.

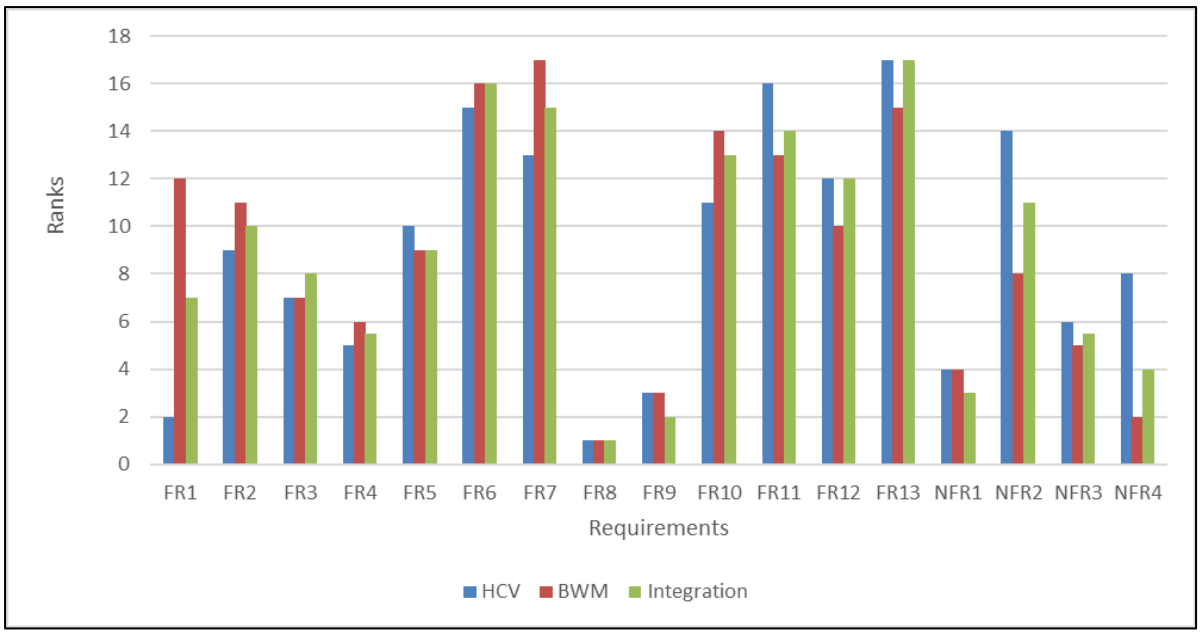

Fig. 4 Comparison of requirement ranking using HCV, BWM, and the proposed integrated framework.

Fig. 4 illustrated that the integrated framework-maintained agreement between the rankings produced by the two techniques for high-priority requirements, while reducing ranking discrepancies for requirements that were received different rankings. This indicated that the RRF algorithm provided an effective aggregation mechanism capable of integrating stakeholder preferences with expert judgments into a unified and balanced ranking.

In addition to the ranking results, the proposed framework demonstrated significant improvements in efficiency compared to the AHP-based approach reported by (Al-Rawashdeh et al. 2025). The hierarchical decomposition of requirements, combined with the use of BWM, contributed to a significant reduction in the evaluation effort while preserving the consistency of prioritization decisions.

Table (2). Framework Evaluation

Evaluation Metrics	**Proposed Framework**	**Baseline (AHP)**
Participants	43 stakeholders + 10 experts	55 participants
Number of Comparisons	26	85
Consistency	0.0617	0.089
Rank Correlation	0.4745	

As shown in Table (2), the assessment results indicated that the proposed framework enhanced the efficiency of the prioritization process and consistency of software requirements prioritization. The framework reduced the comparison effort by reducing the number of comparisons from 85 to 26, realizing a 69.41% reduction compared with the AHP-based approach. In addition, the consistency assessment demonstrated that the experts' judgments maintained a high level of

consistency, with an average CR^I value of 0.0617, indicating satisfactory consistency throughout the BWM evaluation process.
In addition, the participation of 43 stakeholders in the HCV stage and 10 in the BWM phase enabled the framework to incorporate stakeholder preferences with expert judgments, leading to a more comprehensive prioritization process.
Furthermore, Spearman's rank correlation coefficient (ρ = 0.4745) between the ranking produced by the proposed framework and the AHP-based ranking suggested a moderate positive relationship. However, this figure does not mean that the two rankings coincide exactly, and an exact match was not anticipated because the two methods were built on different grouping structures. In the AHP-based approach, requirements were arranged into two categories—functional and non-functional—then requirements were compared within each category and the two categories were compared with each other. By contrast, the proposed framework divided requirements into five functionally cohesive groups and applies HCV and BWM within a three-level hierarchy (goal, groups, requirements), followed by RRF-based integration. Owing to these differences in the basis for categorization, the depth of the hierarchy, and the weighting strategy, the observed moderate correlation indicated instead that the two approaches broadly align on the overall direction of prioritization: requirements ranked highly under AHP are likewise likely to receive high priority under the proposed framework, even though identical rank positions were not required at every level. This, in turn, lends support to the validity of the proposed framework for generating directionally consistent results despite the structural differences between the two prioritization approaches.
In general, the findings showed that the proposed framework enabled a structurally scalable design and also offered a dependable, integrated approach for software requirements prioritization in Agile software development environments.

Conclusion

This study proposed an integrated framework for software requirements prioritization in agile development environments. The framework integrated stakeholder-based hierarchical cumulative voting with expert judgments obtained through the best-worst method, combined through reciprocal rank fusion. The framework was validated using an OQMS case study with 17 requirements, involving 43 stakeholders and 10 experts. The findings indicated that the framework reduced the pairwise comparisons effort by 69.41% compared with the AHP-based approach, while maintaining high judgment consistency (mean CR^I = 0.0617) and achieving a moderate positive correlation (ρ = 0.4745) with the rankings reported in a previous AHP-based study. These results provided empirical evidence supporting the practical applicability of the proposed framework.
Future work may assess the framework using larger and more diverse case studies, developing a software tool for its practical implementation, incorporating fuzzy representations of expert judgments to address uncertainty, exploring the use of LLMs in requirements classification, and simulating stakeholders using multi-agent systems.

References

[1] Tasneem, N., Zulzalil, H. B., & Hassan, S. A. (2025). Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques. IEEE Access.
[2] Jawale, B., & Bhole, A. T. (2015). Adaptive fuzzy hierarchical cumulative voting: a novel approach toward requirement prioritization. International Journal of Research in Engineering and Technology, 4(5), 365-370.
[3] Eldrandaly, B. K. (2023). A Framework of Type-2 Neutrosophic for Requirements Prioritization. Neutrosophic Sets and Systems, 53, 421-432.

[4] Gupta, V., Chauhan, D. S., & Dutta, K. (2015). Exploring reprioritization through systematic literature surveys and case studies. SpringerPlus, 4(1), 539.
[5] Alhenawi, E., Awawdeh, S., Khurma, R. A., García-Arenas, M., Castillo, P. A., & Hudaib, A. (2024). Choosing a suitable requirement prioritization method: A survey. arXiv preprint arXiv:2402.13149.
[6] Bibi et al. (2025). Comparative Study of Requirements Prioritization Techniques. Contemporary Journal of Social Science Review, 3(3), 1301-1314.
[7] Ais, R. C., De Souza, É. F., & Souza, A. C. C. (2026). Requirement prioritization in software engineering: A systematic literature review update. Journal on Interactive Systems, 17(1), 48–68. https://doi.org/10.5753/jis.2026.5371.
[8] Riņķevičs, K., & Torkar, R. (2013). Equality in cumulative voting: A systematic review with an improvement proposal. Information and Software Technology, 55(2), 267-287.
[9] Berander, P., & Svahnberg, M. (2009). Evaluating two ways of calculating priorities in requirements hierarchies–An experiment on hierarchical cumulative voting. Journal of Systems and Software, 82(5), 836-850.
[10] Rezaei, J. (2015). Best-worst multi-criteria decision-making method. Omega, 53, 49-57
[11] Mohamed, S. I., El-Maddah, I. A., & Wahba, A. M. (2008, January). Criteria-based requirements prioritization for software product management. In Software Engineering Research and Practice (pp. 587-593).
[12] Jawale, B. B., Patnaik, G. K., & Bhole, A. T. (2017, January). Requirement prioritization using adaptive fuzzy hierarchical cumulative voting. In 2017 IEEE 7th International Advance Computing Conference (IACC) (pp. 95-102). IEEE
[13] Awais, M. A. (2016). Requirements prioritization: challenges and techniques for quality software development. Adv Comput Sci Int J, 5(2), 14-21.
[14] Olawore, A. S., Wong, K. Y., Ma'aram, A., & Sutopo, W. (2023). Prioritization of technology commercialization success factors using fuzzy best worst method. Journal of Open Innovation: Technology, Market, and Complexity, 9(3), 100096.
[15] Norouzi, A., & Namin, H. G. (2019). A hybrid fuzzy TOPSIS–best worst method for risk prioritization in megaprojects. Civil Engineering Journal, 5(6), 1257-1272.
[16] Al-Rawashdeh, T. A., Al'Azzeh, F., & Al-Tarawneh, F. (2025). An analytical hierarchy Process-based technique for software requirements prioritization. IEEE Access, 13, 50603-50610.
[17] Trivedi, P., Shah, J., Čep, R., Abualigah, L., & Kalita, K. (2024). A hybrid best-worst method (BWM)—technique for order of preference by similarity to ideal solution (TOPSIS) approach for prioritizing road safety improvements. IEEe Access, 12, 30054-30065.

