A Review of Modern Requirement Prioritization: From NLP Pipelines to Agentic AI

Umar Aftab, Inshal Amir, Areeba Akhtar
Department of Computer Science, National University of Technology (NUTECH), Islamabad, Pakistan
*Correspondence: umaraftab@nutech.edu.pk, inshalamirf23@nutech.edu.pk, areebaakhtarf23@nutech.edu.pk

Citation | Aftab. U, Amir. I, Akhtar. A, "A Review of Modern Requirement Prioritization: From NLP Pipelines to Agentic AI", IJIST, Special Issue pp 352-371, May 2026
DOI | https://doi.org/10.33411/IJIST/1774
Received | March 16, 2026 **Revised |** April 27, 2026 **Accepted |** May 03, 2026 **Published |** May 09, 2026.

Requirement Prioritization (RP) and Issue Prioritization (IP) are critical for effective software engineering, dictating optimal resource allocation and deployment sequencing. While conventional frameworks such as the Analytic Hierarchy Process (AHP) and MOSCOW offer structured approaches, they lack the scalability required for modern Agile and DevOps environments. To evaluate the technological trajectory toward automated backlog management, this study conducts a PRISMA-compliant Systematic Literature Review (SLR) analyzing the paradigm shift from foundational Natural Language Processing (NLP) pipelines to Large Language Models (LLMs) and autonomous agentic workflows. Synthesizing data from exactly 78 peer-reviewed empirical studies published between 2015 and 2025, the review quantifies the evolution of prioritization efficacy. The analysis reveals that while static NLP models achieve robust baseline metrics demonstrating average top-3 accuracy scores of 81% and Mean Squared Errors of 2.2 on massive datasets exceeding 29,000 repository issues, they inherently lack deep contextual inference capabilities. Conversely, recent LLM-augmented and deep learning frameworks demonstrate prioritization accuracies ranging from 73% to 90% while reducing processing times to under 25 seconds. Optimized metaheuristic approaches additionally report up to a 30% performance increase over traditional AHP. However, despite these statistical improvements, enterprise LLM deployments remain constrained by hallucination risks and an inability to adapt to real-time temporal dependencies without human intervention. Addressing this critical gap, the review examines the orchestration of LangChain-based agentic architectures capable of dynamic, self-correcting multi-agent decision logic. The study concludes by presenting an agentic workflow framework validated against historical issue tracking datasets and proposes a consolidated research agenda focusing on algorithmic governance, multi-stakeholder fairness, and real-time dependency graph modeling in next-generation automated requirement engineering.

Keywords: Large Language Model (LLM), Natural Language Processing (NLP), Retrieval Augmented Generation (RAG), Requirement Prioritization (RP), Issue Prioritization (IP)

Introduction:

Requirement Prioritization (RP) is a central activity in software requirements engineering because it determines which requirements, features, defects, or technical tasks should receive development attention first. In resource-constrained software projects, effective prioritization supports better release planning, stakeholder satisfaction, risk reduction, and timely delivery of high-value functionality. A closely related subdomain is Issue Prioritization (IP), which focuses on ranking bug reports, feature requests, user stories, and backlog items according to factors such as urgency, severity, business value, technical dependency, user impact, and implementation effort. In contemporary Agile and DevOps environments, where issue repositories expand continuously and project priorities change rapidly, RP/IP has become increasingly difficult to manage through manual decision-making alone [1][2].

Traditional prioritization methods, including the Analytic Hierarchy Process (AHP), MoSCoW, cost-value analysis, and other expert-driven techniques, provide structured mechanisms for decision support. However, these approaches generally depend on human judgment, static criteria, and time-consuming pairwise comparisons. As a result, they are often unsuitable for large-scale, fast-changing, and multi-stakeholder software environments [3]. Their limited scalability becomes more visible when development teams must continuously process high volumes of textual artifacts such as bug reports, app reviews, user feedback, feature requests, and Jira or GitHub issues. These limitations have motivated the use of automated and data-driven methods for extracting meaning from unstructured requirement data and supporting more consistent prioritization decisions.

Natural Language Processing (NLP) introduced an important shift in automated RP/IP by enabling semantic analysis, text classification, clustering, duplicate detection, and relevance-based ranking of software artifacts [4][5]. NLP-based models have demonstrated value in reducing manual effort and improving scalability, particularly when dealing with large issue repositories. However, many conventional NLP pipelines remain limited by their reliance on handcrafted features, domain-specific training data, and static processing workflows. Although these systems can identify lexical and semantic similarities, they often struggle to capture deeper contextual factors such as implicit stakeholder intent, evolving project constraints, cross-requirement dependencies, and real-time operational changes.

The emergence of Large Language Models (LLMs) has further transformed automated requirements engineering by enabling zero-shot and few-shot classification, natural language reasoning, summarization, and explanation generation [6][7]. Compared with traditional NLP models, LLMs can interpret richer context and generate more human-readable justifications for prioritization decisions. Nevertheless, their adoption in production-level RP/IP remains challenging. LLM-based systems may produce hallucinated outputs, inherit bias from training data, lack transparent decision trails, and require careful governance before they can be trusted in high-stakes software engineering contexts [8][9]. These concerns indicate that LLMs should not be treated merely as isolated predictors; instead, they should be embedded within controlled, auditable, and context-aware workflows.

A promising direction is the integration of LLMs with Retrieval-Augmented Generation (RAG), LangChain-based orchestration, and multi-agent architectures. Such agentic workflows can retrieve relevant project knowledge, coordinate specialized agents, apply prioritization logic, validate outputs, and maintain traceable decision logs. This transition from static NLP pipelines to autonomous agentic systems represents a significant shift in RP/IP research. However, despite the increasing interest in generative AI for software engineering, there remains a lack of consolidated review work that critically examines how agentic AI and LangChain-style orchestration can support scalable, explainable, and governance-aware prioritization.

Critical Synthesis of Contemporary Literature (2020-2025):

Recent literature from 2020 to 2025 shows a clear movement from manual prioritization and conventional NLP pipelines toward machine learning, deep learning, LLM-based reasoning, and agentic decision-support systems. [8] Reviewed automated support for requirements engineering and identified an increasing role of machine learning in requirement analysis, validation, and decision support. However, they also highlighted persistent gaps in empirical validation, industrial adoption, and lifecycle integration. This indicates that automated RP/IP remains an active research area requiring stronger methodological consolidation and practical evaluation.

[4] Proposed NLP4IP, an NLP-based recommendation approach for issue prioritization. Their model used textual and metadata-driven features with machine learning algorithms such as XGBoost to predict issue ranks. Evaluated on 29,698 Jira issues, NLP4IP achieved 81% Top-3 accuracy and a Mean Squared Error (MSE) of 2.2. These results demonstrate that NLP-based approaches can support large-scale issue analysis and reduce manual prioritization effort. However, the framework remains limited in its ability to support real-time reprioritization, dynamic dependency reasoning, and multi-stakeholder negotiation.

To improve prioritization performance, [1] introduced a hybrid approach based on critical software project factors and the Black Hole Algorithm (BHA). Their method achieved a 99.67% fitness score on the RALIC dataset and showed an approximately 30% improvement over traditional AHP-based prioritization. This demonstrates the potential of optimization-based methods for improving ranking quality. However, such approaches often depend on predefined project factors and may become computationally expensive or difficult to adapt in continuously changing agile environments.

More recently, [2] proposed an automated deep learning framework for software requirement prioritization. Their approach reported prioritization accuracy between 73% and 90% and reduced execution time to approximately 17–25 seconds. These findings show that deep learning can improve automation speed and ranking performance. However, fully automated systems that minimize stakeholder involvement may create risks of misalignment, especially when human preferences, business priorities, ethical concerns, or contextual dependencies are not explicitly modeled.

The use of LLMs in requirements engineering has also gained increasing attention. [3] Conducted a systematic review on the use of LLMs in software requirement engineering and identified growing adoption of LLMs for ambiguity reduction, requirement formalization, and specification generation. Similarly, [6] examined the role of generative AI in requirements engineering and reported F1-scores ranging from 0.47 to 0.68 in LLM-supported classification tasks. Their findings further suggest that full AI automation remains limited in practice, whereas Human-AI Collaboration (HAIC) is more widely accepted. This reflects a major industrial concern: although LLMs can improve reasoning and automation, their outputs still require validation, traceability, and human oversight.

Collectively, recent studies show measurable progress in automated RP/IP. NLP-based models have demonstrated scalability on datasets containing tens of thousands of issues, optimization-based techniques have improved prioritization fitness, and deep learning approaches have reduced processing time while maintaining competitive accuracy. However, three major gaps remain unresolved. First, many models are static and do not adapt effectively to real-time project changes. Second, existing approaches often lack explicit dependency modeling across requirements, stakeholders, and operational constraints. Third, LLM-based systems raise governance, hallucination, bias, and explainability concerns. These gaps motivate the need to review the transition from traditional NLP pipelines and isolated LLM prompting toward agentic, auditable, and context-aware prioritization workflows.

Table 1 summarizes the empirical progression and limitations of selected contemporary studies in automated requirement and issue prioritization.

Table 1. Contemporary Literature Synthesis in Automated Requirement Prioritization (2020–2025)

Author & Year	Primary Methodology	Dataset / Scale	Key Performance Metric	Identified Empirical Gap
Shafiq et al. (2021)	NLP & XGBoost (NLP4IP)	29,698 JIRA Issues	81% Top-3 accuracy; 2.2 MSE	Lacks real-time adaptability to shifting dependencies.
Arshad et al. (2023)	Hybrid BHA Metaheuristics	RALIC Dataset	99.67% Fitness; 30% AHP gain	Computationally heavy; relies on static project factors.
Jamasb et al. (2025)	Automated Deep Learning	RALIC Dataset	73–90% Accuracy; 17– 25s execution	Excludes stakeholder intervention, risking misalignment.
Arora et al. (2024)	LLM Human-AI Synergy	Proprietary Feedback	0.47–0.68 F1 Classification Score	Highlights industry distrust; only 5.4% full automation.
Hemmat et al. (2025)	LLM Specification Generation	SLR of 29 Studies	Upward adoption trajectory mapped	Identifies persistent issues with hallucination and bias.

Research Motivation:

Although recent studies have advanced automated requirement and issue prioritization, the literature remains fragmented across NLP, machine learning, deep learning, LLMs, and optimization-based methods. Existing reviews and empirical studies largely examine these techniques as independent models or isolated predictors. In particular, many works focus on classification accuracy, ranking metrics, or execution time, but give less attention to how these techniques can be orchestrated into continuous, auditable, and adaptive decision-support systems.

This gap is especially important in modern Agile and DevOps settings, where prioritization is no longer a one-time ranking activity. Instead, priorities must be updated continuously as new defects are reported, stakeholder needs change, dependencies emerge, and operational constraints evolve. Static NLP pipelines and single-prompt LLM systems are not sufficient for this level of dynamic decision-making. Therefore, there is a need to examine agentic AI workflows that combine retrieval, reasoning, validation, feedback, and auditability within a single prioritization architecture.

The motivation of this review is to provide a structured analysis of this transition. Rather than reviewing NLP or LLMs only as standalone tools, this study investigates how LangChain-orchestrated and RAG-enabled agentic workflows can support the next generation of requirement and issue prioritization. Such workflows have the potential to improve scalability, contextual reasoning, traceability, fairness, and governance in automated RP/IP systems.

Novelty of the Study:

The novelty of this study lies in its explicit focus on the shift from static NLP-based prioritization and isolated LLM prompting toward agentic, multi-step, and auditable prioritization workflows. Unlike earlier studies that primarily evaluate individual models according to performance metrics, this review examines how multiple AI components can be integrated into an autonomous decision-support architecture for RP/IP.

Specifically, this review contributes a novel perspective by positioning LangChain-style orchestration, Retrieval-Augmented Generation, and multi-agent reasoning as emerging

mechanisms for addressing the limitations of current LLM-based requirement engineering systems. These mechanisms are particularly relevant for mitigating hallucination, improving traceability, supporting human-in-the-loop validation, and enabling real-time reprioritization based on changing project signals.

Unique Contributions:

The main contributions of this study are as follows:

This study provides a critical and literature-backed synthesis of the technological evolution from conventional prioritization techniques to NLP-based models, LLM-supported reasoning, and agentic AI workflows for RP/IP.

It consolidates recent empirical evidence from 2020–2025, including reported metrics such as Top-3 accuracy, MSE, prioritization accuracy, execution time, and comparative improvement over traditional methods.

It identifies key unresolved research gaps in automated prioritization, including real-time adaptability, dependency awareness, hallucination control, fairness, explainability, and governance.

It conceptually maps LangChain-orchestrated, RAG-enabled, and multi-agent architectures onto the requirement prioritization lifecycle.

It proposes a research direction for developing auditable, policy-aware, and human-supervised agentic prioritization systems for modern software engineering environments.

Formal Research Objectives:

To address the identified gaps, this study is guided by the following research objectives:

RO1: To systematically examine the technological and methodological evolution of automated requirement and issue prioritization from traditional techniques and NLP pipelines to LLM-supported and agentic AI workflows.

RO2: To synthesize recent empirical evidence from 2020–2025 and compare the reported performance, scalability, and limitations of NLP, machine learning, deep learning, and LLM-based prioritization approaches.

RO3: To critically evaluate the role of LangChain-orchestrated and RAG-enabled agentic workflows in improving contextual reasoning, traceability, and adaptability within issue prioritization.

RO4: To identify the major open challenges related to governance, fairness, bias mitigation, explainability, dependency modeling, and real-time scalability in autonomous RP/IP systems.

RO5: To formulate a consolidated research agenda for future development of auditable, human-in-the-loop, and policy-aware agentic prioritization frameworks.

Materials and Methods:

This study adopts a PRISMA-compliant Systematic Literature Review (SLR) methodology to examine the evolution of automated Requirement Prioritization (RP) and Issue Prioritization (IP) from traditional Natural Language Processing (NLP) pipelines to Large Language Model (LLM)-supported and agentic AI workflows. The use of an SLR is appropriate because the topic spans multiple research streams, including requirements engineering, machine learning, NLP, generative AI, Retrieval-Augmented Generation (RAG), LangChain-based orchestration, and multi-agent systems. A systematic review approach allows these fragmented research areas to be evaluated through a transparent, reproducible, and bias-controlled process.

Methodological Alignment and Problem Justification:

The title of this study focuses on the transition "from NLP pipelines to Agentic AI"; therefore, the methodology must capture both historical developments in automated prioritization and recent advances in LLM-based agentic architectures. A PRISMA-guided SLR was selected to ensure that the literature identification, screening, eligibility assessment, and final inclusion process followed a transparent and replicable structure. This approach

directly addresses the research problem by enabling the study to compare traditional automated prioritization techniques with newer agentic systems in terms of scalability, contextual reasoning, traceability, fairness, and practical deployment readiness.

Unlike a narrative review, which may rely heavily on selective interpretation, the PRISMA-based SLR reduces selection bias by defining search sources, Boolean search strings, inclusion and exclusion criteria, screening stages, and quality assessment procedures. This is important for the present study because claims about the evolution from NLP to LLMs and agentic workflows must be supported by traceable literature evidence rather than general observation.

Search Strategy and Reproducibility Elements:

A structured literature search was conducted across three major scientific databases relevant to software engineering and artificial intelligence: IEEE Xplore, ACM Digital Library, and arXiv. The search window was restricted to studies published between January 2015 and February 2025. This period was selected to capture the development of machine learning and transformer-based NLP models, while also including the recent emergence of LLMs, RAG, LangChain-based orchestration, and agentic AI frameworks.

To ensure reproducibility, the following Boolean search string was applied across the title, abstract, and keyword fields of the selected databases:

("Requirement Prioritization" OR "Issue Prioritization" OR "Issue Tracking" OR "Backlog Management") AND ("Natural Language Processing" OR "NLP" OR "Large Language Models" OR "LLM" OR "Generative AI" OR "Agentic AI" OR "LangChain" OR "Multiagent")

The initial database search produced 1,432 records. Specifically, 615 records were retrieved from IEEE Xplore, 485 from the ACM Digital Library, and 332 from arXiv. After automated duplicate removal, 318 duplicate records were excluded, leaving 1,114 unique records for title and abstract screening. These numerical details were included to make the study selection process transparent and reproducible.

Inclusion and Exclusion Criteria:

To ensure that only relevant and methodologically sound studies were included, the screening process was guided by predefined inclusion and exclusion criteria.

Inclusion Criteria:

IC1: The study must be a peer-reviewed journal article, conference paper, or high-impact preprint proposing, evaluating, or reviewing automated AI, ML, NLP, LLM, or agentic approaches for requirement prioritization, issue prioritization, backlog management, or closely related requirements engineering tasks.

IC2: The study must provide methodological, empirical, or architectural relevance to automated prioritization, such as ranking accuracy, Top-K accuracy, F1-score, Mean Squared Error, execution time, scalability analysis, or system architecture.

IC3: The study must discuss or evaluate technologies relevant to the scope of this review, including NLP pipelines, transformer-based models, LLMs, RAG, LangChain-style orchestration, multi-agent workflows, explainability, fairness, or governance in automated requirements engineering.

Exclusion Criteria:

EC1: Studies focused only on manual prioritization methods, such as isolated AHP or Moscow, were excluded unless they were used as a baseline for comparison with automated or AI-supported approaches.

EC2: Studies applying NLP, ML, or LLMs to software engineering tasks outside the scope of prioritization were excluded. Examples include studies focused only on code generation, vulnerability detection, test-case generation, or general software maintenance without a clear connection to requirement or issue prioritization.

EC3: Non-English publications, textbooks, tertiary literature, and extended abstracts lacking methodological detail were excluded.

During title and abstract screening, 1,012 records were excluded. Of these, 412 were excluded under EC1 because they focused only on manual or non-automated prioritization methods, 504 were excluded under EC2 because they addressed unrelated software engineering tasks, and 96 were excluded under EC3 because they lacked sufficient formatting, language, or methodological quality. This left 102 full-text articles for eligibility assessment.

Quality Assessment and Bias Control:

To improve the reliability of the final synthesis, the 102 full-text articles were assessed using a structured quality assessment protocol. Two independent reviewers examined each study using a four-point quality rubric adapted for software engineering literature reviews. The purpose of this stage was to reduce publication bias, exclude weak empirical studies, and ensure that the final review was based on studies with sufficient methodological transparency.

The quality assessment focused on four dimensions: methodological clarity, dataset validity, metric robustness, and baseline comparison. Studies that did not meet the minimum quality threshold were excluded. A total of 24 full-text studies were removed at this stage because they lacked empirical rigor, did not provide sufficient methodological detail, or failed to include meaningful evaluation metrics. The final synthesis, therefore, included 78 high-quality primary studies.

Table 2. Quality Assessment Rubric for Bias Control and Empirical Validation

QA Metric	Assessment Question	Scoring Rationale
QA1: Methodological Clarity	Is the AI/NLP algorithmic architecture explicitly defined to allow for independent reproducibility?	Mitigates black-box bias; ensures verifiable algorithmic logic.
QA2: Dataset Validity	Is the evaluation dataset sufficiently large, standardized, and representative (e.g., massive public JIRA repositories or the RALIC dataset)?	Controls for overfitting; ensures models are tested on realistic industrial scales.
QA3: Metric Robustness	Are standardized, objective statistical metrics utilized for evaluation (e.g., Top-K accuracy, MSE)?	Ensures comparative validity across disparate studies.
QA4: Baseline Comparison	Does the study explicitly benchmark the novel AI approach against established empirical baselines?	Proves definitive performance gains rather than isolated testing.

PRISMA Study Selection Flow:

The study selection process is summarized in the PRISMA 2020 flow diagram shown in Figure 1. The diagram visually reports each stage of the review process, beginning with the identification of 1,432 records from IEEE Xplore, ACM Digital Library, and arXiv. After removing 318 duplicate records, 1,114 unique studies were screened through title and abstract analysis. At the screening stage, 1,012 records were excluded because they did not meet the inclusion criteria or matched one of the exclusion categories. The remaining 102 full-text studies were then assessed for eligibility. During this stage, 24 studies were excluded because they failed the quality assessment threshold or lacked strict empirical baselines. As a result, 78 studies were included in the final qualitative and quantitative synthesis.

This flow diagram directly supports the reproducibility of the review by showing how the final set of reviewed studies was obtained from the initial search pool. It also addresses selection bias by clearly reporting the number of studies removed at each stage and the reasons for exclusion.

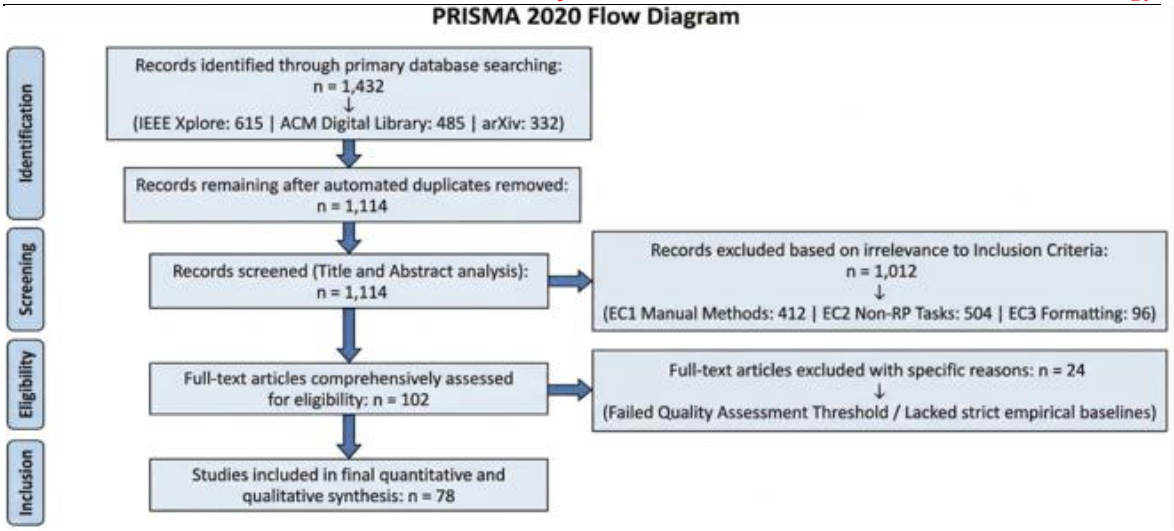


Figure 1. PRISMA 2020 flow diagram showing the systematic study selection process for reviewing requirement prioritization and agentic AI.

Synthesized Findings and Comparative Analysis:

From Foundational NLP to Agentic Workflows:

This section synthesizes the reviewed literature according to the technological progression observed in automated RP/IP research. The analysis is organized into five major categories: requirement and issue prioritization fundamentals, NLP-enhanced prioritization, LLM-driven prioritization, agentic workflow integration, and LangChain-enhanced orchestration. This structure allows the study to directly answer how prioritization has evolved from static pipelines to more adaptive and auditable agentic systems.

Requirement and Issue Prioritization Fundamentals:

Requirement Prioritization is the process of ranking software requirements according to their relative importance, urgency, business value, and technical feasibility, risk, and dependency relationships. Its purpose is to help development teams allocate limited resources to the most valuable or critical software functions. Issue Prioritization is a related activity that focuses specifically on ordering defects, feature requests, user stories, technical tasks, and backlog items within software project management platforms such as Jira or GitHub.

Traditional RP/IP methods, including AHP, MoSCoW, cost-value analysis, and expert-based ranking, provide structured decision support but often struggle in large-scale Agile and DevOps environments. These methods are usually static, time-consuming, and dependent on manual stakeholder judgment. As issue repositories grow in volume and complexity, purely manual prioritization becomes less practical. This has created a need for data-driven methods capable of analyzing large amounts of textual requirement data and supporting continuous reprioritization.

NLP-Enhanced Prioritization: Automating at Scale:

NLP represents the first major stage in the automation of RP/IP. NLP-based techniques allow software teams to process unstructured textual artifacts such as bug reports, feature requests, app reviews, user stories, and issue comments. Common NLP methods used in prioritization include text classification, topic modeling, semantic similarity analysis, sentiment analysis, duplicate detection, clustering, and keyword extraction.

The main advantage of NLP-based prioritization is scalability. NLP systems can process thousands of issue reports more efficiently than manual review. For example, semantic clustering can group similar issues, while classification models can predict urgency, severity, or priority categories. This reduces manual overhead and improves consistency in large software repositories.

However, the reviewed literature also shows important limitations. Many NLP pipelines rely on fixed feature extraction, domain-specific training data, and batch-based processing. As a result, they often fail to capture implicit stakeholder intent, changing business priorities, cross-requirement dependencies, or real-time operational signals. These limitations explain why recent research has moved beyond conventional NLP toward transformer-based models, LLMs, and agentic workflows.

LLM-Driven Prioritization: From Classification to Contextual Reasoning:

LLMs represent a major shift in automated RP/IP because they can perform contextual interpretation, zero-shot classification, few-shot learning, explanation generation, and multi-step reasoning. Unlike traditional NLP pipelines, which often depend on handcrafted features or limited domain-specific training data, LLMs can interpret incomplete or ambiguous issue descriptions and generate human-readable rationales for prioritization decisions.

In RP/IP, LLMs can support several tasks. They can summarize long issue discussions, identify urgency signals, infer stakeholder concerns, detect dependency clues, compare competing requirements, and generate priority justifications. This makes LLMs particularly useful in environments where issue descriptions are informal, inconsistent, or distributed across multiple sources.

Despite these advantages, LLM-based prioritization introduces serious risks. LLMs may produce hallucinated explanations, reflect bias from training data, respond inconsistently to prompt variations, or fail to provide verifiable decision trails. These concerns are particularly important in software engineering contexts where prioritization decisions affect cost, delivery timelines, customer satisfaction, and safety-critical functionality. Therefore, the reviewed studies suggest that LLMs should be embedded within structured, auditable, and human-supervised workflows rather than used as standalone decision-makers.

Agentic Workflow Integration: Toward Autonomous Prioritization:

Agentic workflows extend LLM-based prioritization by placing LLMs inside autonomous, task-oriented systems. In these workflows, agents can monitor incoming issues, retrieve relevant project knowledge, classify issue types, estimate impact, identify dependencies, validate outputs, and recommend priority changes. Unlike static NLP or single-prompt LLM systems, agentic workflows can be designed to operate continuously and respond to new project signals.

The main value of agentic workflows lies in their adaptability. For example, an agentic prioritization system can detect a new critical bug, retrieve related historical issues, check service-level agreements, evaluate dependency impact, consult stakeholder rules, and recommend an updated priority. This makes agentic systems especially relevant for Agile and DevOps environments, where priorities change frequently.

However, agentic prioritization also introduces new research challenges. Autonomous agents require governance mechanisms, traceable decision logs, role-based constraints, fairness checks, and human feedback loops. Without these safeguards, agentic systems may produce opaque or unreliable decisions. Therefore, the literature indicates that future RP/IP systems must combine autonomy with auditability and human oversight.

LangChain-Enhanced Workflow Orchestration:

LangChain-style orchestration provides a practical mechanism for structuring LLM-based and agentic prioritization workflows. Instead of using an LLM as a single isolated model, LangChain-like frameworks allow multiple components to be connected into a modular workflow. These components may include retrievers, vector databases, prompt templates, classification chains, scoring agents, validation agents, memory modules, and audit logs.

In the context of RP/IP, a LangChain-orchestrated workflow can support the following sequence. First, input triggers such as new requirements, bug reports, feature

requests, or stakeholder queries enter the system. Second, the orchestrator agent determines which specialized agents should handle the task. Third, specialized agents perform focused functions such as issue classification, semantic clustering, dependency detection, impact scoring, or validation. Fourth, the outputs from these agents are combined into a decision synthesis module that produces a final prioritization recommendation. Finally, the system records the reasoning steps, retrieved evidence, agent outputs, and final decision in an audit log.

Figure 2 illustrates this simplified agentic workflow. The process begins with input triggers, which represent incoming requirements, issue tickets, user feedback, or backlog updates. These inputs are passed to an orchestrator agent, which coordinates the workflow. The orchestrator distributes subtasks to specialized agents. Each specialized agent performs a defined role, such as classification, dependency analysis, impact estimation, fairness checking, or validation. The outputs of these agents are then returned to the orchestrator and forwarded to the decision synthesis stage. The decision synthesis module combines the evidence and produces a final prioritization recommendation. In a complete implementation, this output should also be stored in a traceable audit log to support transparency, governance, and reproducibility.

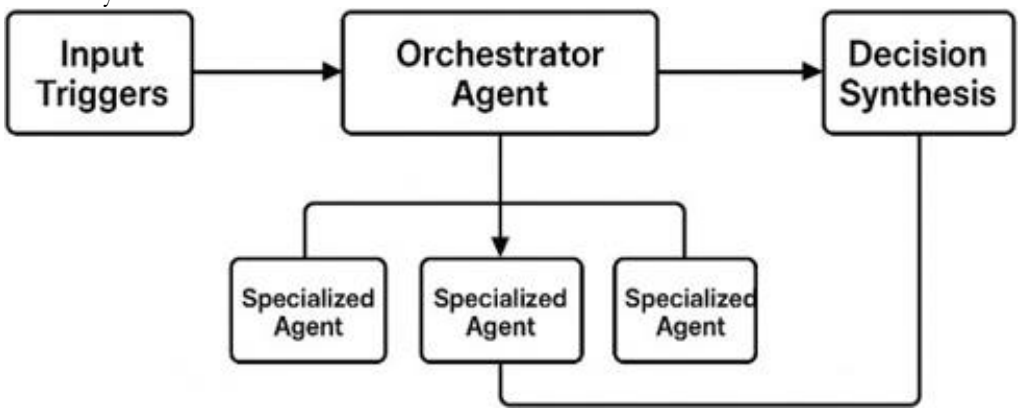


Figure 2. LangChain-orchestrated agentic workflow for requirement and issue prioritization.

This step-by-step explanation describes the operational mechanism of the workflow and clarifies how the architecture supports adaptive prioritization. The model is not presented as a fully validated industrial deployment; rather, it is used as a conceptual architecture derived from the reviewed literature to demonstrate how LLMs, RAG, LangChain-style orchestration, and specialized agents can be integrated into a transparent and traceable RP/IP system.

In summary, the reviewed literature shows a clear progression from manual and rule-based prioritization methods toward NLP, ML, deep learning, LLM-based reasoning, RAG-enabled retrieval, and agentic workflow orchestration. However, the same literature also shows that no single approach fully resolves the core challenges of RP/IP. NLP improves scalability but lacks deep reasoning. LLMs improve contextual interpretation but introduce hallucination and traceability risks. Agentic workflows improve adaptability but require governance, validation, and audit mechanisms. Therefore, the most promising direction is the development of hybrid, human-supervised, and policy-aware systems that combine the scalability of NLP, the reasoning capability of LLMs, the grounding ability of RAG, and the modular control of Lang Chain-style agentic workflows.

Table 3. Core Approaches, Methodologies, Limitations, and Research Support

Approach	Core Methodology	Limitations / Gaps	Research Support
AHP / Traditional Prioritization	Pairwise comparison of requirements based on stakeholder-defined criteria weights.	Time-consuming for large requirement sets and weak adaptability to changing project contexts.	[1][10][11][12]
Hybrid AHP and Fuzzy Logic	Combines structured prioritization with fuzzy inference to handle uncertainty.	Becomes complex at scale and remains limited in fast-changing Agile environments.	[1][10][11][12]
Search-Based Prioritization	Uses metaheuristic optimization methods to identify high-value prioritization sequences.	Requires parameter tuning and may be computationally expensive for real-time use.	[13][12][14]
NLP-Based Issue Prioritization	Applies semantic similarity, text classification, metadata analysis, and clustering to rank issue reports.	Scalable for large repositories but limited in dependency reasoning and real-time adaptation.	[4][7][15][11]
Automated ML / Deep Learning Frameworks	Use embeddings, clustering, ranking models, and learned representations for automated prioritization.	Often require curated datasets, empirical validation, and domain-specific calibration.	[2][8][16][17][18]
LLM-Augmented Prioritization	Uses LLMs to interpret issue descriptions, infer context, classify priority, and generate explanations.	Vulnerable to hallucination, bias, prompt sensitivity, and weak traceability.	[3][6][5][9][19][20]
RAG-Enabled Prioritization	Retrieves project-specific documents, historical issues, or stakeholder rules before generating priority recommendations.	Requires reliable knowledge bases, retrieval quality control, and governance mechanisms.	[21]
LangChain-Based Agentic Workflows	Orchestrates retrievers, LLMs, specialized agents, validators, and audit logs in a modular workflow.	Early-stage adoption; lacks standardized benchmarking and large-scale industrial validation.	[21][19][22][20]
Fairness / Explainability / Governance	Focuses on ensuring transparency, bias mitigation, accountability, and explainability in AI-	Still evolving; lacks standardized metrics and consistent integration into real-	[23][24][25][26][27]

	driven prioritization systems.	world prioritization pipelines.	
PRISMA / SLR Methodology	Structured approach for conducting systematic literature reviews using transparent selection and reporting criteria.	Dependent on study quality, and may introduce selection bias if not rigorously followed.	[28]

Results and Discussion:

This section presents the synthesized results of the systematic literature review. The findings are organized according to the research objectives and research questions of the study. The analysis shows a clear technological progression from traditional requirement prioritization methods to NLP-based automation, LLM-supported reasoning, and emerging agentic workflows. Across the reviewed studies, three major result patterns were identified: first, NLP and ML-based approaches improve scalability in large issue repositories; second, LLM-based approaches improve contextual interpretation but introduce risks related to hallucination, bias, and traceability; and third, agentic and LangChain-orchestrated workflows offer a promising direction for adaptive, auditable, and real-time prioritization, although they remain insufficiently validated in industrial environments.

Results for RQ1: Evolution from Static NLP Pipelines to LLM-Based Reasoning:

The reviewed literature indicates that automated prioritization has evolved through three major stages. The first stage consists of traditional prioritization techniques such as AHP, MoSCoW, cost-value analysis, and release planning. These methods provide structured decision support but depend heavily on manual stakeholder judgment and static evaluation criteria. As a result, they become difficult to apply in large Agile and DevOps environments where issue backlogs continuously expand, and priorities change rapidly [1][10][11][12].

The second stage is represented by NLP and ML-based prioritization. NLP techniques such as text classification, semantic similarity analysis, topic modeling, clustering, and metadata-based ranking have improved the ability to process large volumes of requirement and issue data [4][7][15][11]. For example, NLP-based issue prioritization has been evaluated on large Jira repositories and has shown strong ranking performance, including Top-K accuracy and error-based evaluation metrics. These findings confirm that NLP-based systems are useful for reducing manual workload and improving scalability.

The third stage is represented by LLM-supported and deep learning-based prioritization. Recent studies show that deep learning and LLM-based approaches can improve contextual understanding, support natural language reasoning, generate explanations, and reduce processing time [2][3][6][9]. However, the results also show that LLM-based systems cannot yet be treated as fully autonomous decision-makers. Their outputs require validation because of risks such as hallucination, prompt sensitivity, weak traceability, and possible bias in prioritization recommendations.

Therefore, the result for RQ1 is that automated RP/IP has progressed from static, manually supported methods to scalable NLP pipelines and then toward LLM-based contextual reasoning. However, this progression remains incomplete because current systems still struggle with real-time adaptation, dependency awareness, explainability, and governance.

Results for RQ2: LLM Prompting Versus LangChain-Orchestrated Agentic Workflows:

The review shows a major difference between isolated LLM prompting and LangChain-orchestrated agentic workflows. Isolated LLM prompting typically uses a single model interaction to classify, summarize, or rank a requirement or issue. This approach is simple and flexible, but it often lacks external grounding, traceable reasoning, structured validation, and reproducible decision logic.

In contrast, LangChain-orchestrated workflows allow multiple components to be connected into a modular pipeline. These components may include retrievers, vector databases, RAG modules, prompt templates, specialized agents, validation agents, memory modules, and audit logs. In an RP/IP context, such workflows can retrieve historical issue data, apply domain-specific rules, classify issue severity, estimate stakeholder impact, detect dependencies, validate the generated recommendation, and store the decision trail for later review.

The results indicate that agentic workflows are more suitable than isolated LLM prompting for complex prioritization environments because they can support multi-step reasoning, tool use, retrieval grounding, human feedback, and auditability. However, the reviewed literature also shows that agentic prioritization is still an emerging area. There is limited benchmarking across public datasets, limited evidence of large-scale industrial deployment, and no widely accepted evaluation framework for agentic RP/IP systems.

Therefore, the result for RQ2 is that LangChain-orchestrated agentic workflows provide a stronger architectural foundation for adaptive and explainable prioritization than single-prompt LLM systems. However, they require more rigorous validation, governance, and performance benchmarking before they can be adopted as reliable enterprise-level prioritization systems.

Results for RQ3: Governance, Fairness, Dependency Awareness, and Scalability Challenges:

The third major result concerns unresolved challenges that limit the practical adoption of autonomous RP/IP systems. Across the reviewed studies, four recurring limitations were identified.

First, scalability and real-time adaptation remain unresolved. NLP-based systems can process large textual datasets, but many of them operate in batch mode and do not continuously update priorities when new issues, stakeholder feedback, or project constraints emerge. LLMs improve reasoning capability, but they introduce computational overhead and may not be suitable for real-time use without careful workflow optimization.

Second, dependency awareness remains underdeveloped. Many prioritization models rank issues or requirements independently, without fully modeling relationships such as feature dependencies, blocking defects, temporal constraints, cross-team dependencies, or business-critical interactions. This is a major limitation because software priorities often depend on the relationships between requirements rather than on individual requirement descriptions alone.

Third, fairness and bias are major concerns in AI-assisted prioritization. Automated models may reproduce historical bias, overrepresent dominant stakeholder groups, or undervalue minority user concerns. This is especially important in multi-stakeholder software projects where prioritization decisions can affect different user groups unevenly.

Fourth, governance and explainability remain insufficient. LLM and agentic systems must provide traceable justifications for why one issue or requirement is ranked above another. Without audit logs, fairness checks, human review, and transparent reasoning mechanisms, autonomous prioritization systems may not be trusted in industrial or high-stakes environments.

Therefore, the result for RQ3 is that the main barriers to autonomous RP/IP adoption are not only technical but also organizational and ethical. Future systems must combine scalability, contextual reasoning, dependency modeling, fairness assurance, and auditability.

Table 4. Research Gaps and Opportunities

Research Focus	Current Gaps	Opportunities for Innovation
NLP Scalability Limits	Batch-based processing and domain-specific training dependencies	Develop domain-agnostic embeddings and streaming NLP methods for near-real-time prioritization updates.
LLM Responsiveness Constraints	High computational cost, prompt sensitivity, and limited orchestration for continuous prioritization	Optimize lightweight LLMs and embed them within adaptive, validated pipelines.
RAG-Enabled Prioritization	Retrieval quality issues and weak integration with project-specific rules	Use RAG to ground prioritization decisions in historical issues, requirements, policies, and stakeholder feedback [21]
LangChain Modular Integration	Limited policy-aware chaining and static retriever configurations	Design governance-aware chains with dynamic retriever updates based on project signals [21][19][22]
Agentic Flow Governance	Lack of validation frameworks and opaque decision logic	Build explainable agentic systems with audit trails, role constraints, and human feedback loops [22][20]
Hybrid Architecture Design	Fragmented toolchains and limited interoperability between NLP, LLMs, RAG, and agents	Create unified frameworks combining NLP, LLMs, RAG, LangChain-style orchestration, and shared memory.
Real-Time Prioritization in Practice	Few deployments in regulated or high-stakes software environments	Pilot adaptive prioritization in Agile delivery, public-sector systems, safety-critical systems, and enterprise DevOps.

Discussion:

Scalability and Real-Time Adaptation:

The findings show that scalability has improved significantly through NLP, ML, and deep learning methods. These approaches can process large volumes of issue reports, app reviews, user stories, and stakeholder feedback more efficiently than manual prioritization.

However, scalability alone is not sufficient. Modern Agile and DevOps environments require continuous reprioritization as new information becomes available. Many existing systems still operate as static or batch-based models, meaning that they do not automatically update rankings when project conditions change.

LLM-based and agentic workflows offer a possible solution because they can interpret new signals, retrieve contextual evidence, and update recommendations dynamically. However, real-time deployment remains difficult because of computational cost, latency, validation requirements, and governance concerns. For this reason, future prioritization systems should not rely on LLMs alone. Instead, they should use hybrid architectures in which lightweight NLP models handle high-volume filtering, while LLMs and agents are used for deeper reasoning, validation, and decision explanation.

Contextual Reasoning and Dependency Awareness:

The review also shows that dependency awareness remains a major limitation in existing RP/IP systems. Many models can rank issues based on textual similarity, urgency, or predicted severity, but they do not fully capture how one requirement affects another. This is

problematic because software prioritization often depends on technical constraints, stakeholder dependencies, release constraints, security risks, and business goals.

LLMs provide stronger contextual reasoning than traditional NLP models, but they may still fail to consistently identify dependencies unless they are supported by structured prompts, retrieval mechanisms, knowledge graphs, or agentic validation. LangChain-style workflows and RAG can improve this process by connecting the model to historical issue data, architecture documents, policy documents, and stakeholder feedback. However, the reviewed literature indicates that dependency-aware agentic RP/IP is still underdeveloped.

Table 5. Dependency Analysis and Opportunity Exploration

Paradigm	Strengths in Contextual Reasoning	Gaps in Dependency Awareness	Research Opportunities
NLP-Enhanced Prioritization	Semantic clustering, named entity recognition, sentiment analysis, and topic modeling	Weak handling of implicit dependencies and limited temporal reasoning	Graph-based semantic dependency extraction and cross-document linkage.
LLM-Driven Prioritization	Zero-shot reasoning, contextual embeddings, multiturn dialogue, and explanation generation	Inconsistent dependency tracing and sensitivity to prompt design	Fine-tuned dependency-aware LLMs and prompt engineering for causal reasoning.
RAG-Enabled Prioritization	Grounds model outputs in retrieved project evidence and historical issue data	Dependent on the retrieval quality and completeness of knowledge sources	Dependency-aware retrieval using issue graphs, traceability matrices, and project documentation.
LangChain-Enhanced Prioritization	Modular logic chaining, tool integration, and traceable workflows	Static chaining and limited adaptive dependency modeling	Dynamic chain construction and dependency-aware retrievers and agents.
Agentic Flow Prioritization	Autonomous monitoring, real-time adaptation, and multi-agent collaboration	Lack of standardized dependency frameworks and coordination complexity	Ontology-driven agent reasoning and temporal or causal dependency graphs.

Bias, Fairness, and Explainability:

AI-driven RP/IP systems also raise important fairness and explainability issues. NLP and ML models may learn from historical datasets that reflect biased organizational decisions. LLMs may generate persuasive yet incorrect explanations or prioritize dominant stakeholder perspectives over minority concerns. Agentic systems may amplify these risks if multiple autonomous agents interact without clear governance constraints.

Fairness must therefore be treated as a core design requirement rather than an optional evaluation metric. Prioritization systems should include checks for corpus balance, feature fairness, prompt sensitivity, reasoning transparency, retrieval fairness, stakeholder feedback, and auditability. Explainability is equally important because software teams need to understand why a specific requirement or issue was ranked highly. In industrial settings, traceable explanations can improve stakeholder trust, support compliance, and allow human reviewers to challenge or revise AI-generated recommendations.

Table 6. Bias, Fairness, and Explainability Dimensions for Stakeholders

Audit Dimension	Key Questions for Stakeholders	Applicable Paradigms
Corpus Balance	Are training datasets representative of all stakeholder groups, issue categories, and project contexts?	NLP, ML, LLM
Feature Fairness	Do selected features amplify dominant narratives or exclude minority stakeholder concerns?	NLP, ML
Prompt Sensitivity	Do prompt variations produce inconsistent or biased prioritization outcomes?	LLM
Reasoning Transparency	Can the system explain why one requirement was prioritized over another?	LLM, RAG, LangChain, Agentic
Chain-Level Explainability	Are intermediate workflow steps documented and interpretable?	LangChain, Agentic
Retriever Fairness	Are retrieval modules selecting evidence in an equity-aware and contextually balanced manner?	RAG, LangChain
Agent Ethics Protocols	Do agents follow fairness, accountability, and role-based constraints during decision-making?	Agentic
Stakeholder Feedback Integration	Can stakeholders review, contest, or recalibrate prioritization decisions?	LangChain, Agentic
Auditability and Logging	Are decisions recorded in a way that supports post-hoc review and fairness audits?	All paradigms

Implications of the Study A. Theoretical Implications:

This study contributes to the theoretical understanding of automated requirement and issue prioritization by showing that the field is moving beyond isolated model performance toward workflow-level decision intelligence. Earlier research mainly focused on whether a model could classify, cluster, or rank requirements accurately. The present synthesis shows that future RP/IP research must also consider orchestration, retrieval grounding, dependency modeling, fairness, explainability, and governance. This expands the conceptual scope of RP/IP from a ranking problem to an adaptive decision-support problem.

Practical and Industrial Implications:

For software teams, the findings suggest that AI-assisted prioritization can reduce manual effort and improve the consistency of backlog management. NLP and ML models can support large-scale filtering and classification, while LLMs can generate explanations and summarize stakeholder concerns. Agentic workflows can further support continuous monitoring and reprioritization. However, industrial use should remain human-supervised. Teams should avoid allowing autonomous agents to make final prioritization decisions without human validation, audit logs, and clear accountability rules.

Governance Implications:

The study also has important governance implications. Automated prioritization systems should include transparent decision trails, documented data sources, bias checks, stakeholder feedback mechanisms, and human override options. In regulated or high-stakes domains, such as healthcare, public-sector software, financial systems, or safety-critical engineering, auditability should be treated as a core system requirement. Agentic workflows

are promising, but only if they are designed with institutional controls, policy alignment, and explainable decision logic.

Future Research Directions:

Based on the identified gaps, future research should focus on the following directions:

Benchmark agentic RP/IP systems on public Jira, GitHub, app-review, and requirements datasets using standardized metrics such as Top@K accuracy, F1-score, MSE, execution time, and stakeholder agreement.

Develop dependency-aware prioritization models that combine LLMs with requirement graphs, traceability matrices, temporal dependency models, and causal reasoning.

Evaluate RAG-enabled prioritization systems that retrieve historical issues, software architecture documents, stakeholder policies, and release plans before generating priority recommendations.

Design fairness-aware prioritization frameworks that measure whether AI systems overrepresent or underrepresent specific stakeholder groups, issue categories, or user concerns.

Investigate human-in-the-loop governance models where product owners, developers, testers, and business stakeholders can review, contest, and recalibrate AI-generated prioritization decisions.

Develop lightweight agentic workflows for real-time Agile and DevOps environments where latency, cost, and continuous adaptation are critical.

Create standardized audit-log formats for agentic RP/IP systems so that every recommendation can be traced back to the retrieved evidence, agent reasoning steps, and final decision criteria.

Conduct industrial case studies to test whether LangChain-style agentic workflows can improve prioritization quality, reduce backlog triage time, and maintain stakeholder trust in real-world software teams.

Strategic Research Agenda:

The transition toward agentic prioritization introduces new research questions that should guide future work.

RQ1: Governance and Traceability in Autonomous Prioritization:

How can decision trails generated by autonomous agents be audited to ensure compliance with organizational policies?

Future work should focus on developing standardized audit-log formats for agentic prioritization. These logs should record the input issue, retrieved evidence, agent decisions, validation steps, stakeholder feedback, and final priority recommendation. This would allow auditors and project managers to verify how each prioritization decision was produced.

RQ2: Real-Time Dependency Graph Modeling:

How can lightweight agents dynamically update requirement dependency graphs without high computational cost?

Future research should investigate the integration of LLMs with graph-based models, traceability matrices, and dependency-aware retrievers. Such systems could allow agents to detect when one requirement blocks another, when a new issue affects a critical feature, or when a priority should change because of evolving project constraints.

RQ3: Fairness and Bias Mitigation in Multi-Stakeholder Negotiation:

How can agentic systems balance conflicting stakeholder priorities without amplifying historical data bias?

Future work should explore fairness-aware agents or "advocacy agents" designed to represent underrepresented stakeholder perspectives during prioritization. These agents could help ensure that prioritization decisions are not dominated only by historically powerful user groups, frequent requesters, or majority feedback patterns.

Conclusion:

This systematic literature review examined the evolution of Requirement Prioritization and Issue Prioritization from conventional decision-support methods to NLP-enhanced models, LLM-driven reasoning, and emerging agentic AI workflows. The review shows that automated prioritization has made measurable progress in scalability, classification performance, execution time, and contextual reasoning. NLP-based systems have improved the processing of large issue repositories, while ML and deep learning models have supported more efficient ranking and classification. LLMs have introduced stronger contextual interpretation and explanation generation, and RAG-enabled or LangChain-orchestrated workflows provide a promising foundation for traceable, adaptive, and policy-aware prioritization. However, the findings also show that several limitations remain unresolved. Current systems still struggle with real-time adaptation, dependency modeling, hallucination control, fairness, explainability, and industrial validation. Agentic workflows represent the most promising direction because they can combine retrieval, reasoning, specialized agents, validation, and audit logging. Nevertheless, these systems should not be deployed as fully autonomous decision-makers without human oversight and governance controls. The major conclusion of this review is that the future of RP/IP lies in hybrid, human-supervised, and audit-ready architectures. Such systems should combine the scalability of NLP, the reasoning capabilities of LLMs, the grounding strength of RAG, and the modular control of LangChain-style agentic workflows. By doing so, software teams can move toward prioritization systems that are not only faster and more scalable but also more transparent, fair, context-aware, and aligned with stakeholder needs.

References:

[1] H. Arshad, S. Shaheen, J. A. Khan, M. S. Anwar, K. Aurangzeb, and M. Alhussein, "A novel hybrid requirement's prioritization approach based on critical software project factors," *Cogn. Technol. Work 2023 252*, vol. 25, no. 2, pp. 305–324, Jun. 2023, doi: 10.1007/s10111-023-00729-3.

[2] Ireneusz Miciuła, Łukasz Radlí, "An Automated Framework for Prioritizing Software Requirements," *Electronics*, vol. 14, no. 6, p. 1220, 2025, doi: https://doi.org/10.3390/electronics14061220.

[3] Arshia Hemmat, Mohammadreza Sharbaf, "Research directions for using LLM in software requirement engineering: a systematic review," *Front. Comput. Sci.*, vol. 7, 2025, doi: https://doi.org/10.3389/fcomp.2025.1519437.

[4] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "NLP4IP: Natural Language Processing-based Recommendation Approach for Issues Prioritization," *Proc. - 2021 47th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2021*, pp. 99–108, Sep. 2021, doi: 10.1109/SEAA53835.2021.00022.

[5] "(PDF) Evaluating the Capabilities of LLMs in Traceability Maintenance for Automotive System and Software Requirements: Three Case Studies." Accessed: Mar. 18, 2026. [Online]. Available: https://www.researchgate.net/publication/389906003_Evaluating_the_Capabilities_of_LLMs_in_Traceability_Maintenance_for_Automotive_System_and_Software_Requirements_Three_Case_Studies

[6] Chetan Arora, John Grundy, Mohamed Abdelrazek, "Advancing Requirements Engineering through Generative AI: Assessing the Role of LLMs," *arXiv:2310.13976*, 2023, [Online]. Available: https://arxiv.org/abs/2310.13976

[7] Saurabh Malgaonkar, Sherlock A. Licorish, "Prioritizing user concerns in app reviews: A study of requests for new features, enhancements and bug fixes," *Inf. Softw. Technol.*, vol. 144, p. 106798, 2022, doi: https://doi.org/10.1016/j.infsof.2021.106798.

[8] Muhammad Aminu Umar & Kevin Lano, "Advances in automated support for

requirements engineering: a systematic literature review," *Requir. Eng.*, vol. 29, pp. 177–207, 2024, [Online]. Available: https://link.springer.com/article/10.1007/s00766-023-00411-0

[9] Tariq Shahzad, Tehseen Mazhar, Muhammad Usman Tariq, Wasim Ahmad, Khmaies Ouahada & Habib Hamam, "A comprehensive review of large language models: issues and solutions in learning environments," *Discov. Sustain.*, vol. 6, 2025, [Online]. Available: https://link.springer.com/article/10.1007/s43621-025-00815-8

[10] J. A. Crowder and C. W. Hoff, "MDRE: The Requirements Engineering Process," *Requir. Eng. Lay. a Firm Found.*, pp. 75–84, 2022, doi: 10.1007/978-3-030-91077-8_5.

[11] P. Tálele and R. Phalnikar, "Classification and prioritisation of software requirements using machine learning - A systematic review," *Proc. Conflu. 2021 11th Int. Conf. Cloud Comput. Data Sci. Eng.*, pp. 912–918, Jan. 2021, doi: 10.1109/CONFLUENCE51648.2021.9377190.

[12] Jonathan Winton, Francis Palma, "Improving Software Requirements Prioritization through the Lens of Constraint Solving," *arXiv:2306.12391*, 2023, [Online]. Available: https://arxiv.org/abs/2306.12391

[13] Hassan Sartaj, Shaukat Ali, Paolo Arcaini, Andrea Arcuri, "Search-Based Software Engineering and AI Foundation Models: Current Landscape and Future Roadmap," *arXiv:2505.19625*, 2025, [Online]. Available: https://arxiv.org/abs/2505.19625

[14] F. Sarro, "Search-Based Software Engineering in the Era of Modern Software Systems," *Proc. IEEE Int. Conf. Requir. Eng.*, vol. 2023-September, pp. 3–5, 2023, doi: 10.1109/RE57278.2023.00010.

[15] M. Grootendorst, "BERTopic: Neural topic modeling with a class-based TF-IDF procedure," Mar. 2022, Accessed: Sep. 16, 2024. [Online]. Available: http://arxiv.org/abs/2203.05794

[16] A. Tanveer, N. B. Ibrahim, M. Z. Rehman, and N. M. Nawi, "A Framework for Handling Scalability in Requirements Prioritization Using IMPA Algorithm for Large Scale Projects," *2024 2nd Int. Conf. Comput. Data Anal. ICCDA 2024 - Proc.*, 2024, doi: 10.1109/ICCDA64887.2024.10867359.

[17] S. S. Tanveer and Z. A. Rana, "Prioritizing Software Requirements by Combining the Usage Monitoring and User Feedback Data," *IEEE Access*, vol. 12, pp. 82825–82841, 2024, doi: 10.1109/ACCESS.2024.3409847.

[18] Shizhen Bai, Songlin Shi, "Prioritizing user requirements for digital products using explainable artificial intelligence: A data-driven analysis on video conferencing apps," *Futur. Gener. Comput. Syst.*, vol. 158, pp. 167–182, 2024, doi: https://doi.org/10.1016/j.future.2024.04.037.

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *arXiv:2201.11903*, 2023, [Online]. Available: https://arxiv.org/abs/2201.11903

[20] J. W. Long Ouyang, "Training language models to follow instructions with human feedback," *Adv. Neural Inf. Process. Syst.*, 2022, doi: https://doi.org/10.48550/arXiv.2203.02155.

[21] D. K. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv:2005.11401*, 2021, doi: https://doi.org/10.48550/arXiv.2005.11401.

[22] Y. C. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, "ReAct: Synergizing Reasoning and Acting in Language Models," *arXiv:2210.03629*, 2023, [Online]. Available: https://arxiv.org/abs/2210.03629

[23] Usman Gohar, Lu Cheng, "A Survey on Intersectional Fairness in Machine Learning: Notions, Mitigation, and Challenges," *arXiv:2305.06969*, 2023, [Online]. Available: https://arxiv.org/abs/2305.06969

[24] Galen Harrison, Julia Hanson, "An empirical study on the perceived fairness of realistic, imperfect machine learning models," *FAT* 2020 - Proc. 2020 Conf. Fairness, Accountability, Transpar.*, 2020, [Online]. Available: https://dl.acm.org/doi/10.1145/3351095.3372831

[25] Max Hort, Zhenpeng Chen, Jie M. Zhang, Mark Harman, Federica Sarro, "Bias Mitigation for Machine Learning Classifiers: A Comprehensive Survey," *arXiv:2207.07068*, 2023, [Online]. Available: https://arxiv.org/abs/2207.07068

[26] Anjana Perera, Aldeida Aleti, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Burak Turhan, "Search-based fairness testing for regression-based machine learning systems," *Empir. Softw. Eng.*, vol. 27, no. 79, 2022, [Online]. Available: https://link.springer.com/article/10.1007/s10664-022-10116-7

[27] Dana Pessach, Erez Shmueli, "A review on fairness in machine learning," *ACM Comput. Surv.*, vol. 55, no. 3, 2022, [Online]. Available: https://dl.acm.org/doi/full/10.1145/3494672

[28] Matthew J. Page, Joanne E. McKenzie, "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, 2021, [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/33782057/



Copyright © by authors and 50Sea. This work is licensed under the Creative Commons Attribution 4.0 International License.