From Manual Review to Intelligent Insight: An AI Roadmap for Requirements Document Review

Raha AlAssaf
Abu Dhabi School of Management
Abu Dhabi, United Arab Emirates
raha_assaf@yahoo.com

https://doi.org/10.63962/GHNA4772

Abstract—Reviewing software requirements is one of the critical activities in software development projects, yet it is often done manually, which is time-consuming and prone to inconsistency. Ineffective review practices may lead to vague, incomplete, or low-quality requirements. This increases the risk of rework and project failure. This paper proposes an AI roadmap for enhancing the review of requirements documents by using natural language processing and machine learning techniques. The proposed roadmap aims to improve the quality of requirements while reducing the manual review effort and supporting more effective decision-making during the requirements engineering process.

Keywords—Artificial Intelligence, Requirements Engineering, Requirements Review, Natural Language Processing

I. INTRODUCTION AND BACKGROUND

The success of software development projects depends on many internal activities and processes. One such process is software requirements review, which has an important role in ensuring the quality and clarity of requirements [1]. When requirements are clearly reviewed, they help map system functionality to stakeholder needs and project goals [2].

Requirements review is usually performed manually. This manual process is time-consuming and may lead to inconsistent results when requirements documents are long or complex [3]. Different reviewers may interpret the same requirement in different ways, and problems such as ambiguous words, missing details, or conflicting requirements may not be identified early in the project [4]. These problems can increase the risk of rework and affect project outcomes negatively [1].

Recent advances in artificial intelligence (AI) have created new possibilities to support the requirements review process [3]. AI techniques can help reviewers by analysing requirements documents and highlighting quality issues [2]. This paper proposes an AI roadmap for enhancing software requirements document review using natural language processing (NLP) and machine learning techniques.

Requirements review is still mainly performed manually and depends on reviewer experience, which may lead to inconsistent quality assessments and overlooked issues [1][4] [5]. Although previous studies highlight the importance of early review activities in improving software quality, there is still limited integration of AI across different review stages within an organizational workflow [3][6]. This paper proposes an AI roadmap that embeds machine learning techniques into the requirements review lifecycle to improve consistency, efficiency, and decision support.

This paper proposes an organizational roadmap for integrating AI into the software requirements review process. The contribution focuses on using machine learning across multiple review stages rather than limiting the scope to model development, presenting AI adoption as a phased process embedded within existing organizational workflows.

II. REQUIREMENTS REVIEW CHALLENGES

A. *Manual Requirements Review Process*

The software requirements review process is one of the important stages within software development projects [1][2]. Throughout this process, business analysts, developers and testers manually examine requirements documents to make sure they are clear, consistent, complete and aligned with the needs of the client [2][7]. Table 1 shows the main process steps and stakeholders. This process takes a lot of time, is subjective and can be easily affected by human mistakes, especially when the requirements are complex [1].

Process Step	Description	Stakeholders Involved
1. Gather Requirements	Collect business needs from clients by conducting interviews, meetings, and reviewing existing documents.	Business Analyst, Product Owner, Business Users
2. Draft Requirements Document	Create Software Requirement Specification (SRS), user stories, and functional and non-functional requirements documents.	Business Analyst
3. Initial Review	Check requirements clarity, completeness and alignment with project objectives and client needs.	Business Analyst
4. Technical Review	Check whether the requirements can be built, identify technical limitations or constraints and understand how they impact the system.	Developers, Technical Lead, System Architect
5. Quality and Test Review	Check if the requirement can be tested, identify risks, and note any missing acceptance criteria.	Quality Assurance (QA) and Test Engineers
6. Refine Requirements	Update requirements based on feedback from all review groups.	Business Analyst, Developers, QA/ Test Team, Product Owner
7. Final Approval and Sign-off	Approve the finalized requirements so development can start.	Project Manager, Product Owner, Business Users

TABLE I. SOFTWARE REQUIREMENTS REVIEW STEPS

Process Step	Input Data	AI Algorithm	AI Output	Updated Activity (AI-Oriented)
		(clear vs. unclear)	requirements	flagged items instead of reading every line in detail.
4. Technical Review	Requirements with technical details.	Logistic Regression classifier (low risk vs. high risk)	High risk requirement list.	Developers focus their efforts on predicted requirements as high risk.
5. Test Review	Acceptance criteria sections.	Logistic Regression classifier (adequate vs. missing acceptance criteria)	List of requirements with missing acceptance criteria.	QA team strengthens test conditions and acceptance criteria earlier in the project.
6. Refinement	Updated requirements and model predictions.	Logistic Regression (reclassification)	Updated clarity and risk scores.	Faster refinement cycles, guided by updated predictions instead of full manual rereads.
7. Final Approval	Final SRS and AI risk prediction summary.	None.	Clarity and risk summary report.	The Project Manager and Business Users review the AI-generated report to support their approval decision.

B. *Quality Issues in Requirements Documents*

Problems such as using vague words, missing details, or conflicts between statements frequently go unnoticed, which cause defects, rework, and delays in the project [8]. Manual reviews vary depending on the skills each reviewer has, which may introduce quality risks in developing the required software [5]. A study in requirements engineering finds that having unclear requirements early in the project is the major cause of defects and higher costs [9].

C. *Motivation for AI-Supported Review*

AI techniques can support requirements review by analyzing textual patterns, identifying ambiguity, and highlighting potential risk indicators based on historical data [8][9]. Instead of replacing human reviewers, AI can assist them by flagging requirements that need further attention, which helps improve efficiency and reduce the risk of overlooked defects [2].

III. AI IN REQUIREMENTS REVIEW

The integration of AI into the requirements review workflow involves embedding supervised learning models within key review stages. After the business analyst gathers and drafts the requirements, the textual content is preprocessed and transformed into numerical features using TF-IDF representation. These features enable the classification of requirements according to predefined clarity and risk categories [6].

As shown in Table 2, during the initial review, the Logistic Regression model classifies each requirement as “clear” or “unclear”, while in the technical review, the model predicts high-risk requirements based on patterns that the model learns from past data examples [8]. Any missing acceptance criteria are pointed out by the model during the quality review step. These predictions direct the business analysts, developers and QA teams to focus on a list of flagged sections instead of manual line by line review.

TABLE II. AI-ENHANCED SOFTWARE REQUIREMENTS REVIEW PROCESS

Process Step	Input Data	AI Algorithm	AI Output	Updated Activity (AI-Oriented)
1. Gather Requirements	Raw requirement text from meetings, client emails and notes.	None.	None	Business analyst gathers requirements manually. AI starts in step 2.
2. Draft Requirements Document	Draft SRS, user stories, and existing documents.	TF-IDF Vectorizer	Clean structured text ready for classification	Documents are written in a clearer structure to support AI prediction model.
3. Initial Review	Requirement text.	Logistic Regression classifier	List of unclear or incomplete	Business analyst reviews only

Once the business analyst refines the requirements, the machine learning model classifies the revised text again and updates the clarity and risk scores. The team can iterate the refinement step faster [2]. In the final step, the model shows its predictions in a report that is presented to the project manager and business users to support their final approval decision. AI implementation improves the workflow and creates a review process that is more consistent, faster, and less prone to human error [3]. Figure 1 illustrates the AI-enhanced review process across seven steps, including the iterative reclassification feedback loop during refinement.

Although the proposed solution reduces the effort needed to review every requirement manually, human reviewers remain central to the process. Business analysts and technical reviewers examine the AI-generated predictions, refine requirement wording when needed, and decide on necessary corrective actions. The model supports decision-making rather than replacing human judgment, and final approval remains under the responsibility of the project team.

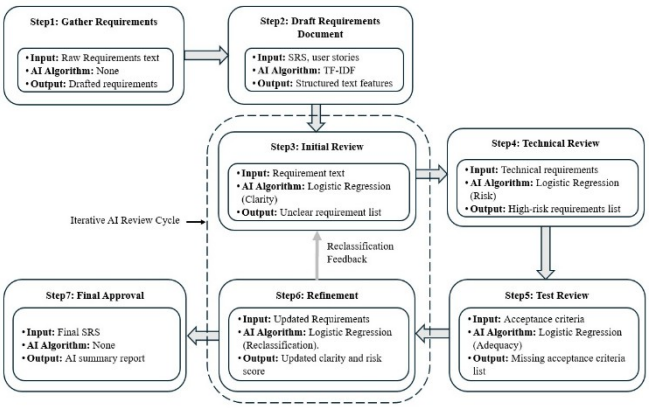


Fig. 1 AI Enhanced Software Requirements Review Process

IV. AI ROADMAP FOR REQUIREMENTS REVIEW

An AI roadmap is proposed to support the Software Development Department in improving the software requirements review process and aligning the revised process with organizational goals. The AI roadmap shows how data, technology and skills support the proposed solution.

A. *Key Use Case*

The main use case is AI-enhanced requirements document review. The TF-IDF and Logistic Regression model are used to find unclear or high-risk requirements early in the development cycle. This supports the aim of the organization to improve the clarity of the requirements and reduce delays resulting from incomplete documentation [1].

B. *Roadmap Phases*

1) *Phase 1: Readiness and Data Preparation*

1) This phase concentrates on preparing the department for adopting AI. The team gathers requirement documents, defines standards for data, sets up version control, and prepares numerical features using TF-IDF under Scikit-learn. This builds a clean database that helps train the model.

2) *Phase 2: Model Development and Pilot Deployment*

This phase focuses on building and testing Logistic Regression classifiers by using Scikit-learn pipelines and labeled requirements. During model training, requirement statements are labeled based on predefined criteria. A requirement is classified as clear if it is specific and testable, and unclear if it contains vague or incomplete information. Requirements are categorized as high-risk when they involve technical uncertainty or dependency conflicts. To ensure consistency, two reviewers independently annotate a subset of requirements and resolve disagreements through discussion. The model outputs clarity and risk scores and highlights statements that require attention. The pilot release of the model is tested within the review workflow before full implementation. To reduce potential bias from historical data patterns or reviewer subjectivity, standardized labeling guidelines are applied and classification results are periodically reviewed.

3) Phase 3: Scaling and Improvement

The third phase concentrates on improving and preparing the model for long-term use. The teams review the model's predictions and share feedback to update the wording, fix any issues and improve the model's accuracy. Also, the department puts in place simple model monitoring and retraining schedules and chooses whether to run the solution on internal servers or to move it to cloud if more resources are needed for capacity. Figure 2 illustrates the overall AI-enhanced requirements review roadmap across the three phases. Table 3 shows how the AI roadmap phases align with AI-enabled process improvements and organizational objectives.

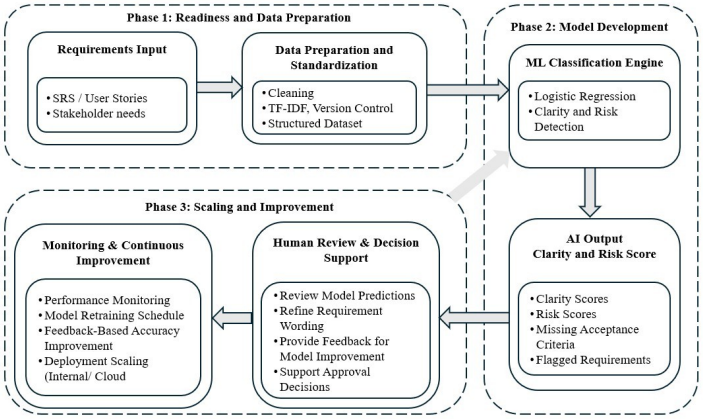


Fig. 1 AI Roadmap Phases for Requirements Review

TABLE III. AI ROADMAP PHASES AND ORGANIZATIONAL OBJECTIVES

Roadmap Phase	AI-Enabled Improvement (AI BP)	Organizational Objective
Phase 1: Readiness and Data Preparation	Standardized and clean requirements data enabling automated analysis	Improve requirements quality, reduce rework
Phase 2: Model Development and Pilot Deployment	Early detection of unclear and high-risk requirements using AI models	Faster requirements analysis, on-time delivery
Phase 3: Scaling and Improvement	Continuous refinement of AI predictions and review consistency	Cost reduction, sustainable quality improvement

C. *Validation*

To provide practical evidence for the proposed approach, a simulated evaluation was conducted using a publicly available software requirements dataset from Kaggle containing 977 labeled requirements [10]. Categories related to performance, availability, security, scalability, and operational concerns were treated as higher-risk requirements.

A TF-IDF representation and Logistic Regression model with class balancing were applied using five-fold cross-validation. The model achieved a mean accuracy of 88.8% and an F1-score of 0.75. These results show that lightweight machine learning techniques can reasonably identify higher-

risk requirements and support the AI-enhanced review process. Although the evaluation is simulated, it provides initial evidence of the feasibility of the proposed roadmap.

D. Strategy Alignment

The AI roadmap follows key elements: data, ethical factors, technology, skills, implementation and change management. The data strategy helps keep the requirement files clean, well structured, safely saved and their versions are controlled for tracking any changes [3][11]. Ethical factors involve reducing bias by using one scoring method for all requirements and making results easy to understand.

The strategy employs Python and Scikit-learn to run the TF-IDF and Logistic Regression models on machines with a standard CPU, with the option to use either internal servers or cloud services. Implementation includes the pilot model integration into the workflow and performance monitoring. Change management focuses more on clear communication, required training and support to help teams in adopting the new process.

E. Expected Benefits

The AI roadmap lowers the manual work in the review process, resulting in a faster process, clearer requirements and better planning [1][3]. When it comes to organizational level, the roadmap lowers the risks that the project may face, helps deliver the project on time and with high quality [2][9].

V. FRAMEWORK EVALUATION AND DISCUSSION

For the organization to shift to an AI-enhanced Software Requirements Review Process, there is a need to evaluate the existing AI implementation frameworks that are widely used in the industry. Frameworks, such as Scikit-Learn, Keras and TensorFlow provide production-ready environments for developing machine learning models that classify text and predict risk scores [12].

A. TensorFlow

TensorFlow is a widely adopted deep learning framework designed for scalable and computationally intensive machine learning tasks [13]. It supports neural network development, GPU acceleration, and distributed training, making it suitable for large and complex datasets [12][14]. However, for the proposed solution, its architectural complexity and higher computational requirements may not be necessary for moderate-sized requirements datasets typically found in organizational environments.

B. Keras

Keras is a high-level neural network framework built on top of TensorFlow that provides a simple interface for developing and testing deep learning models [13]. It supports rapid prototyping and reduces coding complexity, making it suitable for teams seeking to experiment with neural architectures [16]. However, for the proposed roadmap, its deep learning capabilities may not be necessary for lightweight text classification tasks such as requirements clarity and risk detection.

C. Scikit-learn

Scikit-learn is a common open-source framework for data science and machine learning that is built on top of Python libraries [12]. It is simple, lightweight and gives the needed tools for data preparation, feature extraction, classifier training and evaluation by the use of pipelines [16]. The main advantages are its tools that are easy to use, quick training for most models, and it has TF-IDF and machine learning algorithms built in [13]. The implementation of Scikit-learn needs the team to convert requirement text into TF-IDF features, train a classifier, check its performance and integrate this pipeline into the review process.

D. Comparison and Recommendations

Table 4 compares the three frameworks based on complexity, features, cost, and compatibility, highlighting their differences.

Deep learning approaches are well suited for complex natural language tasks, particularly when substantial computational resources and specialized model management are available [15]. However, for organizational requirements review processes that prioritize interpretability, ease of maintenance, and practical deployment within existing workflows, lightweight classical machine learning models provide a more feasible solution [12].

TABLE IV. AI IMPLEMENTATION FRAMEWORKS COMPARISON (adapted from [12], [13], [15], [16])

Framework	Category	Complexity	Key Feature	Cost	Compatibility
TensorFlow	Deep learning framework	High	Deep learning, scalable model training, GPU support.	Free (open source)	Works on cloud, on premises, Python APIs.
Keras	High level deep learning framework	Medium	Simple API for fast neural network building, supports text layers.	Free (open source)	Runs on TensorFlow, cross platform.
Scikit-learn	Classical ML implementation environment	Low	Classical ML pipeline with TF-IDF and classifiers.	Free (open source)	Python-based, easy local or cloud deployment.

Based on the evaluation and comparison presented in Table 4, Scikit-learn is recommended for enhancing the software requirements review process, as it aligns directly with the proposed TF-IDF and Logistic Regression solution [12][16]. In addition, Scikit-learn is easy to implement and maintain, and it produces interpretable results that support decision-making during the review process [12][13]. It provides the necessary tools for text preprocessing and classification without introducing the additional architectural complexity associated with TensorFlow and Keras [12][15].

VI. CONCLUSION

This paper evaluates the software requirements review process and finds where AI can make the process faster and more accurate. The use of TF-IDF and Logistic Regression implemented under the Scikit-learn framework provides a simple and effective way for the team to find risky and unclear requirements and reduce any manual work.

Future work may focus on expanding the proposed AI roadmap by evaluating machine learning techniques and testing the framework on larger and more diverse requirements datasets. In addition, secondary data such as published case studies or industry reports may be used to validate the effectiveness of the proposed roadmap and evaluate its impact on requirements quality and review efficiency in real-world software development environments.

REFERENCES

[1] Nazurin, J., Alya, A. K., & Nordin, A. (2024). Collaborative Requirements Review Tool (Collaborev) to Support Requirements Validation. *International Journal on Perceptive and Cognitive Computing, 10*(1), 105–112. https://doi.org/10.31436/ijpcc.v10i1.456

[1] Kotagi, V., & Yadav, S. K. (2023). Analysis for Requirements Testing Phase of Software Development. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4520340

[2] Akter, T., Abid, M. H., Tanna, R. N., Masud, J. H., & Noyon, M. R. (2025). Automating Requirements Engineering using Machine Learning. *International Journal of Computer Applications, 187*(38), 47–55. https://doi.org/10.5120/ijca2025925661

[3] Abualhaija, S., Aydemir, F. B., Dalpiaz, F., Dell'Anna, D., Ferrari, A., Franch, X., & Fucci, D. (2024). Replication in Requirements Engineering: The NLP for RE Case. *ACM Transactions on Software Engineering and Methodology*, *33*(6), 1–33. https://doi.org/10.1145/3658669

[4] Zhi, Q., Gong, L., Ren, J., Liu, M., Zhou, Z., & Yamamoto, S. (2023). Element quality indicator: A quality assessment and defect detection method for software requirement specification. *Heliyon, 9*(5), e16469. https://doi.org/10.1016/j.heliyon.2023.e16469

[5] Guleria, P., Frnda, J., & Srinivasu, P. N. (2025). NLP based text classification using TF-IDF enabled fine-tuned long short-term memory: An empirical analysis. *Array, 27*, 100467. https://doi.org/10.1016/j.array.2025.100467

[6] Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet*, *16*(6), 180. https://doi.org/10.3390/fi16060180

[7] Frattini, J. (2024). Identifying relevant factors of requirements quality: An industrial case study. In D. Méndez & A. Moreira (Eds.), Requirements engineering: Foundation for software quality (pp. 20–36). Springer. https://doi.org/10.1007/978-3-031-57327-9_2

[8] Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2025). Generative AI for Requirements Engineering: A Systematic Literature Review. *Software: Practice and Experience*, spe.70029. https://doi.org/10.1002/spe.70029

[9] Software Requirements Dataset. (2023). Kaggle. Available at: https://www.kaggle.com/datasets/iamvaibhav100/software-requirements-dataset

[2] Alzayed, A. (2024). Evaluating the Role of Requirements Engineering Practices in the Sustainability of Electronic Government Solutions. *Sustainability, 16*(1), 433. https://doi.org/10.3390/su16010433

[10] Rane, N. L., Mallick, S. K., Kaya, O., & Rane, J. (2024). Tools and frameworks for machine learning and deep learning: A review. *Applied machine learning and deep learning: architectures and techniques*, 80-95. Deep Science Publishing. https://doi.org/10.70593/978-81-981271-4-3_4

[11] Kanungo, P. (2023). Machine learning implementation in Python: Performance analysis of different libraries. *World Journal of Advanced Research and Reviews, 20*(1), 1390–1398. https://doi.org/10.30574/wjarr.2023.20.1.2144

[12] Ramchandani, M., Khandare, H., Singh, P., Rajak, P., Suryawanshi, N., Jangde, A. S., Arya, L., Kumar, P., & Sahu, M. (2022). Survey: Tensorflow in Machine Learning. *Journal of Physics: Conference Series*, *2273*(1), 012008. https://doi.org/10.1088/1742-6596/2273/1/012008

[13] Lafta, N. A. (2023). A Comprehensive Analysis of Keras: Enhancing Deep Learning Applications in Network Engineering. *Babylonian Journal of Networking, 2023*, 94–100. https://doi.org/10.58496/BJN/2023/012

[14] Chary, D., & Singh, R. P. (2020). Review on advanced machine learning model: Scikit-learn. *International Journal of Scientific Research and Engineering Development, 3*(4), 526–529. https://ssrn.com/abstract=3694350