J-INTECH (Journal of Information and Technology)
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 10/C/C3/DT.05.00/2025
E-ISSN: 2580-720X || P-ISSN: 2303-1425

Clarifying Organizational Requirements through Rapid Application Development: Evidence from a Performance Appraisal Information System

Handy Setiawan[1*], Koko Wahyu Prasetyo[2]

[1,2]*Universitas Bhinneka Nusantara, Informatika, Informatika, Jl. Raya Tidar No.100, Sukun, Malang, Indonesia,*

Keywords
Iterative prototyping,performance appraisal,rapid application development,requierement evolution,software engineering process

Corresponding Author:
201111033@mhs.stiki.ac.id

Abstract
This study is positioned as a software engineering case study which analyzes the iterative application of Rapid Application Development (RAD) in the engineering of an employee performance appraisal information system within a higher education context. The system was developed to replace a spreadsheet-based process that involved manual score aggregation and fragmented data storage. Using a software engineering case study approach, this research examines how system requirements evolved across development iterations. Design artifacts, such as dashboard interfaces, module configurations, and workflow structures, produced during successive RAD cycles were analyzed as evidence of requirement refinement. The findings show that early development focused on functional centralization, while later iterations introduced configurable modules for evaluation criteria, evaluator assignment, and score normalization. These interface and module revisions reflect progressive clarification of organizational needs and stabilization of business logic. The study demonstrates that RAD supports requirement refinement through iterative prototyping, especially in systems involving multiple roles and configurable structures. The results provide process-oriented insight into requirement evolution in internal organizational information systems.

1. Introduction

Employee performance appraisal in higher education institutions is a complex process that often involves multiple evaluators, leading to challenges in score aggregation, role coordination, and documentation. This multi-evaluator structure increases the complexity of score aggregation and documentation. When the process is managed using separate spreadsheet files, several operational problems may occur, such as manual calculation errors, inconsistent data storage, and difficulty in retrieving historical records. Similar challenges, especially in spreadsheet dependency, multi-role evaluation, and score aggregation, have motivated web-based appraisal and performance measurement systems in Indonesian organizations and education-related settings [1]-[2].

From a software engineering perspective, the challenge is not only to build a functional system, but also to ensure that organizational rules and evaluation structures are correctly translated into system requirements. In many internal organizational systems, requirements are initially described in a general and operational manner, often reflecting informal stakeholder requests and undocumented procedures rather than formally structured system logic. However, as development progresses, hidden complexity may emerge. This situation

often leads to requirement refinement during the development process. Therefore, understanding how requirements evolve across iterations becomes an important aspect of system engineering. Structured requirement engineering practices and documentation have been proposed in Indonesian government settings to reduce ambiguity and improve alignment [3]. Requirement volatility and refinement challenges are widely reported in agile and iterative environments [4], including Indonesian healthcare contexts [5].

Rapid Application Development (RAD) emphasizes short iterative cycles, prototyping, and continuous stakeholder involvement. RAD is often selected when development time is limited and user feedback is essential. While many studies report the successful implementation of systems using RAD, fewer studies examine how iterative prototyping supports requirement clarification in practice. In particular, there is limited discussion on how interface changes and configuration modules reflect the maturation of system requirements in organizational contexts. Empirical RAD work also highlights gaps between method prescriptions and practice, supporting the value of process-oriented analysis [6]. Rapid prototyping has long been used in requirements validation and system evolution studies [7], [8].

This study addresses that gap by analyzing the iterative development of an employee performance appraisal information system. The system was designed to replace a spreadsheet-based process with a centralized web-based solution. Rather than evaluating technical performance metrics, this research focuses on the evolution of requirements as reflected in successive design artifacts. RAD has also been applied in Indonesian healthcare and public-sector contexts under practical constraints [9], [10]. Iterative prototyping is also reported in complex engineering and control domains, reinforcing its relevance for studying system evolution [11], [12].

The study aims to answer the following research questions:

RQ1: How does iterative development in RAD influence the refinement of system requirements?

RQ2: How do interface and module revisions reflect clarification of organizational needs?

By examining the progression of design artifacts across development iterations, this study provides process-oriented insight into how RAD supports requirement stabilization in systems involving multiple roles and configurable business rules. Agile and iterative projects often under-document quality requirements; guidance and empirical findings are available in recent studies [13]-[14]. Recent reviews and conceptual frameworks also summarize common agile requirements engineering challenges and solution directions [15]. Automation and structured artifact generation have also been proposed to improve rapid requirements validation [16].

2. Research Method

This study applies a software engineering case study approach to analyze the iterative implementation of Rapid Application Development (RAD) in the development of an employee performance appraisal information system. Figure 1 illustrates the three main stages of the development process, namely planning, iterative design workshops, and implementation, each of which corresponds to the stages described in this section. The objective is not to evaluate the technical performance of the final system, but to examine how system requirements evolved during the development process. Therefore, the main unit of analysis in this research is the set of design artifacts produced in each iteration. Process-oriented case studies are commonly used to understand requirement refinement under iterative development [5], and have also been applied in safety-critical agile project contexts [17].

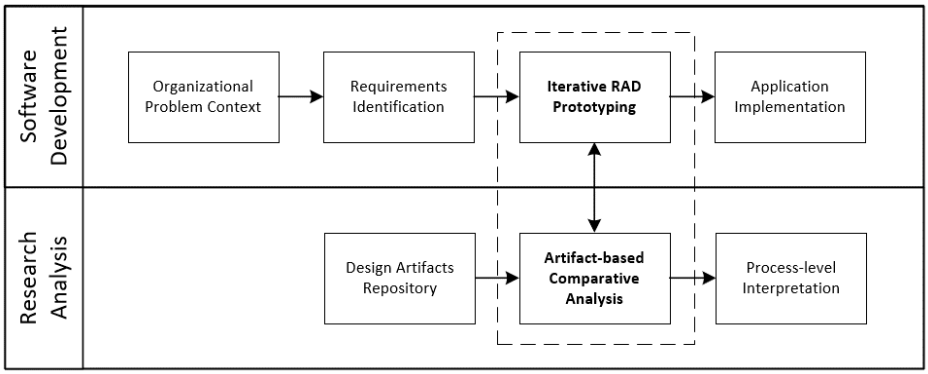

Figure 1. Research methodology overview illustrating the stages of planning, iterative design workshops, and implementation in the RAD process.

The development context involved a multi-evaluator performance assessment process that was previously managed using separate spreadsheet files. This approach created several operational difficulties, including manual score aggregation, inconsistent data storage, and limited access to historical evaluation records. These problems defined the initial scope of system requirements and motivated the adoption of a web-based solution. Similar spreadsheet-to-web transitions are reported in Indonesian employee performance assessment systems [1], [18].

The development process followed the general stages of RAD, which emphasize short development cycles and continuous interaction between developers and stakeholders. In the planning phase, the core requirements were identified through informal discussions with administrative stakeholders, observation of existing spreadsheet workflows, and internal team meetings. These initial requirements formed the basis for the first prototype. During the design workshop phase, the prototype was revised iteratively. Several changes were introduced, including additional configuration modules for defining evaluation criteria, assigning evaluators, and standardizing score interpretation, based on stakeholder feedback, observed workflow limitations, and evaluation of the earlier prototype. These revisions indicate that the understanding of organizational needs became clearer over time. In the implementation phase, the refined modules were integrated into a stable system with automated aggregation and database storage. Comparable RAD-driven development has also been reported in Indonesian case studies in other domains [19], [20], and RAD has been used together with architectural approaches in employee appraisal systems [21].

The analysis in this study uses an artifact-based interpretive approach. Changes across iterations were compared by examining differences in interface structure, configuration capabilities, and workflow representation. Requirement refinement was inferred based on the introduction of new system functionalities, increased configurability, and the formalization of previously implicit organizational rules. Changes in dashboard structure, menu composition, and system configuration features were analyzed as indicators of requirement clarification and refinement. Instead of using quantitative measurements or formal interview transcripts, this study reconstructs the development process through systematic examination of prototype evolution. This approach allows the identification of patterns in requirement stabilization and provides insight into how RAD supports iterative refinement in software engineering practice. This artifact-based logic is consistent with prior work on rapid prototyping for requirements validation and system evolution [7], [8].

3. Result and Discussions

This section presents the findings in relation to RQ1 and RQ2. The analysis examines how iterative RAD cycles influenced requirement refinement (RQ1) and how interface and module revisions reflected clarification of organizational needs (RQ2).

3.1 Initial Problem Identification

The preliminary examination of the existing performance appraisal process revealed three primary issues. First, score aggregation from multiple evaluators was conducted manually using spreadsheet files. This process increased the risk of calculation errors and required additional administrative effort. Second, historical performance data were stored in separate files, making retrieval inefficient and inconsistent. Third, evaluator roles and hierarchical relationships were not formally represented within a structured system. Role clarity and requirements documentation are commonly emphasized in iterative requirements engineering guidance [13] and empirical agile quality requirements studies [8].

These limitations defined the initial scope of system development. At this stage, the main objective was to automate final score calculation and centralize data storage. The initial understanding of requirements focused primarily on operational efficiency rather than structural flexibility. Requirement refinement and change are repeatedly reported as central challenges in iterative development settings [4].

3.2 Iteration 1: Baseline Prototype

The initial prototype developed during the early RAD phase is shown in Figure 2. This dashboard provided basic navigation and access to essential features. The design addressed the need for centralized access and simplified workflow. However, the system structure at this stage assumed that evaluation criteria and evaluator assignments were static. Configuration options were limited, and the underlying business rules were not yet abstracted.

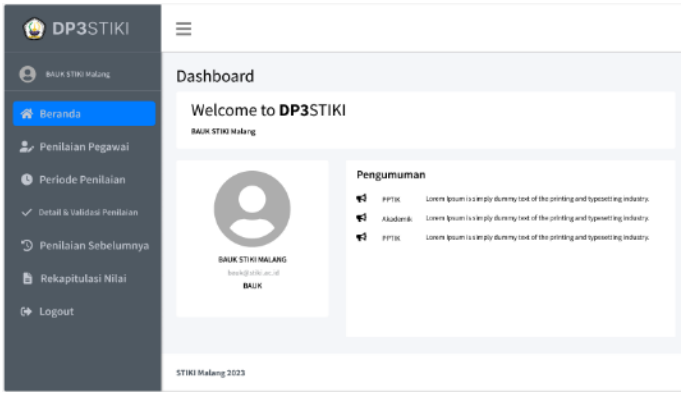

Figure 2. Baseline dashboard prototype representing initial functional requirements focused on centralized access, score input, and basic recap generation..

This dashboard provided basic navigation and access to essential features. The design addressed the need for centralized access and simplified workflow. However, the system structure at this stage assumed that evaluation criteria and evaluator assignments were static. Configuration options were limited, and the underlying business rules were not yet abstracted. Early prototypes often reflect incomplete domain understanding and act as baselines for subsequent refinement [7], [8].

The baseline prototype therefore reflects explicit operational requirements, such as score input functionality, centralized access to the system, and basic recap generation, but it does not fully capture the organizational complexity of the performance evaluation process.

3.3 Iteration 2: Requirement Refinement through Configurability

After conducting design workshops and reviewing the initial prototype, several revisions were introduced. The updated dashboard is presented in Figure 3.

Compared to Figure 2, the revised interface includes additional configuration menus. This modification indicates that stakeholders required greater flexibility in defining evaluation parameters, as inferred from stakeholder feedback and the evolution of the prototype during workshop discussions. The need for

configurability emerged as a response to the recognition that organizational structures and assessment policies may change over time. This pattern is consistent with reported agile and iterative requirements engineering challenges and responses through incremental refinement [4], [8].

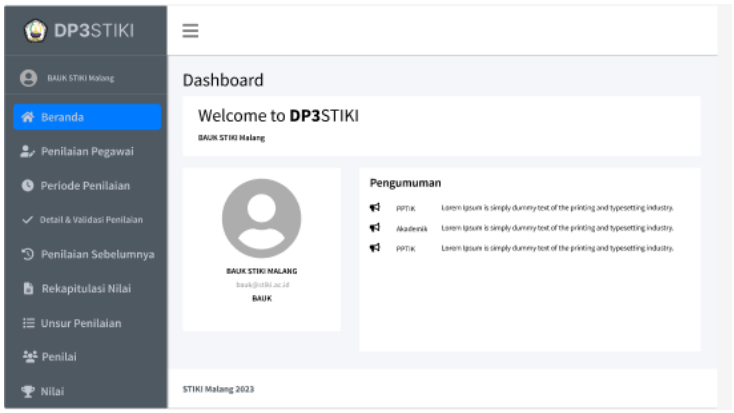

Figure 3. Revised dashboard after iteration showing the introduction of configuration features to support flexible evaluation parameters.

One significant addition during this iteration is the evaluation criteria configuration module, shown in Figure 4. The introduction of this module reflects a shift from fixed assessment elements toward configurable criteria. Initially, evaluation components were implicitly assumed to be constant. However, iterative clarification revealed the necessity to adjust criteria according to institutional needs. From a software engineering perspective, this change represents abstraction of domain rules into configurable system components. Standard operating procedures for requirements engineering can support this abstraction and alignment process [3].

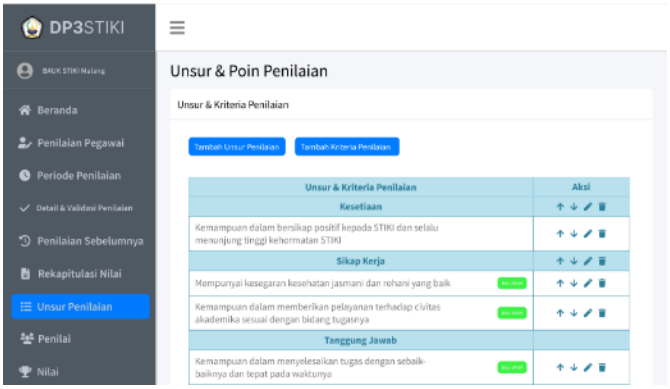

Figure 4. Evaluation criteria configuration module illustrating the transition from fixed assessment elements to a configurable evaluation structure.

Another important refinement is the evaluator assignment module presented in Figure 5. This module formalizes hierarchical evaluator relationships within the system. Previously, such relationships were managed outside the system and depended on administrative conventions. By embedding these relationships into a structured module, the system transforms implicit organizational practices into explicit software logic. This change indicates a deeper understanding of role-based requirements. In relation to RQ2, this revision demonstrates how interface changes made implicit organizational structures explicit in system design. Interface evolution as evidence of requirement maturation is consistent with rapid prototyping studies on system evolution [8].

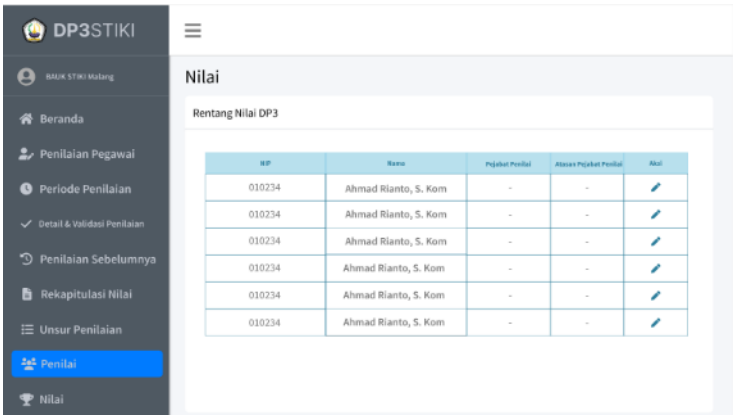

Figure 5. Evaluator configuration module showing the formalization of multi-role evaluator relationships within the system.

The third major addition in this iteration is the score normalization module illustrated in Figure 6.

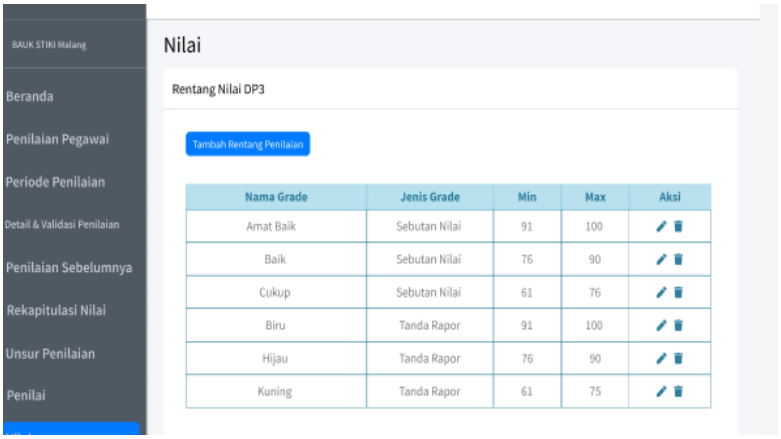

Figure 6. Score normalization module supporting consistent interpretation of evaluation results across different organizational units.

This module standardizes the interpretation of evaluation scores across different units. Without such normalization, aggregated results could lead to inconsistent interpretations. The system therefore evolved to include mechanisms that ensure semantic consistency in score representation. This refinement demonstrates the progressive stabilization of business logic. Quality requirement documentation guidance highlights the importance of consistent definitions and representations during iterative development [13]. These iterative refinements provide direct evidence for RQ1, showing that RAD cycles enabled progressive identification and restructuring of system requirements.

3.4 Iteration 3: Aggregation Stabilization and Reporting

The final major artifact analyzed in this study is the performance recap module shown in Figure 7. This module integrates automated averaging of evaluator scores and provides structured access to historical data stored in a centralized database. At this stage, development focused less on adding new configuration options and more on consolidating aggregation mechanisms. The logic for calculating final scores becomes explicitly defined and consistently implemented. Stabilization phases are commonly observed after shared understanding of domain rules is achieved in iterative development [5].

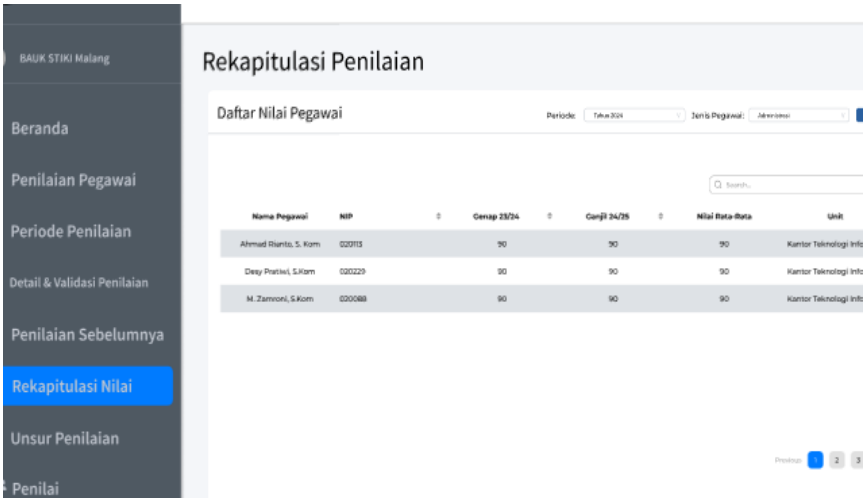

Figure 7. Aggregated performance recap module illustrating the stabilization of score aggregation logic and centralized reporting.

The transition from configurability expansion to aggregation stabilization indicates that requirement clarification had reached a mature stage. Iterations in this phase aimed to ensure consistency and reliability rather than to redefine structural components.

3.5 Discussions

The progression from Figure 2 to Figure 7 demonstrates how RAD supported iterative requirement refinement as illustrated in Table 1. The development process can be interpreted as consisting of three stages: initial functional centralization, introduction of configurability, and stabilization of aggregation logic. Each stage reflects an increased understanding of organizational needs.

Table 1. Mapping between artifact changes and requirement clarifications

Iteration	Artifact Change	Requirement Clarification
Iteration 1	Basic dashboard	Centralized access and score input
Iteration 2	Configuration modules	Flexible criteria and role definition
Iteration 3	Aggregation module	Standardized scoring and reporting

In relation to RQ2, the transition from a static dashboard to configurable modules demonstrates how interface revisions made implicit organizational structures more explicit within the system. The addition of configuration modules suggests that early requirements underestimated the variability and structural complexity of the evaluation process. Through iterative prototyping, stakeholders were able to recognize the importance of flexible criteria, formalized evaluator roles, and standardized scoring schemes. These refinements emerged during workshop interactions and were captured in subsequent design artifacts.

The findings suggest that interface evolution can serve as observable evidence of requirement maturation, within the context of this case study and with appropriate caution regarding generalization. Rather than relying solely on formal requirement documents, the progression of design artifacts reveals how problem understanding improved across iterations. In this case, RAD functioned not only as a development model but also as a structured mechanism for negotiating and stabilizing requirements.

Some requirements only became visible after stakeholders interacted with early prototypes, suggesting that initial requirements were incomplete representations of actual workflow complexity. This indicates that, in internal organizational systems, requirement discovery is not fully achievable through upfront specification

but requires iterative refinement through system use and feedback. The findings imply that systems involving multiple roles and evaluation structures particularly benefit from iterative prototyping, as organizational rules are often implicit and only become explicit when operationalized in system interfaces. In relation to broader agile and RAD practices, this case reinforces the view that prototyping is not only a development tool but also a mechanism for requirement clarification and negotiation.

Therefore, the findings collectively answer RQ1 and RQ2 by demonstrating that iterative RAD development both refines requirements and makes organizational needs visible through interface evolution. Comparable iterative prototyping benefits have also been reported in other complex domains [11], [12]. Automated generation of prototype data has been proposed to support faster and more systematic requirements validation [18].

This study is limited to artifact-based interpretation and does not include quantitative evaluation of development efficiency or user satisfaction. However, the results indicate that RAD can effectively support iterative clarification in systems involving multiple roles and configurable business rules. The evolution of the dashboard and related modules illustrates how prototyping contributes to requirement stabilization within practical software engineering contexts. Similar system development efforts in Indonesia often complement artifact-based reporting with broader evaluation components [9], [19].

4. Conclusions and Future Works

This study analyzed the iterative use of Rapid Application Development (RAD) in developing an employee performance appraisal information system. The focus was not on technical performance, but on how system requirements evolved during the development process. The findings show that RAD supported gradual clarification of organizational needs through successive prototype revisions.

The development process moved through three main stages. The first stage focused on centralizing access and automating score calculation. In the second stage, additional configuration modules were introduced, including evaluation criteria, evaluator assignment, and score normalization. These changes indicate that the initial understanding of requirements was still incomplete. Through iteration, implicit organizational rules were translated into explicit system structures. In the final stage, the system logic for aggregation and reporting became more stable and consistent.

The results suggest that interface changes can reflect requirement maturation. Design artifacts provide visible evidence of how understanding improved across iterations. In this case, RAD functioned not only as a fast development approach, but also as a structured way to refine and stabilize requirements in a multi-role organizational setting. The results suggest that interface changes can reflect requirement maturation; however, this conclusion is derived from artifact-based analysis rather than direct user evaluation.

This study has limitations. The analysis is based on design artifacts and does not include quantitative measures such as usability testing or time efficiency evaluation. The absence of interviews or direct user feedback means that the study captures visible design evolution rather than the full stakeholder reasoning behind each requirement change.

Future research may expand this work by including structured stakeholder interviews or usability assessment. Comparative studies with other development approaches may also provide deeper insight into methodological strengths and weaknesses. In addition, quantitative evaluation of system impact could strengthen empirical evidence. A longer-term study may also examine how requirement stability changes after deployment.

Overall, this study highlights the importance of iterative refinement in software engineering practice, especially for systems that require flexible configuration and clear role definition.

5. References

[1] D. Bambang, T. Wijaya, T. Wahyono, A. Nugrahesthy, and S. Hapsari, "TOPSIS Method Implementation for Employee Performance Information System," *International Journal of Information Technology and Business*, 2019, doi: 10.24246/ijiteb.422022.21-26.

[2] R. Apriani, F. Dwita, Sumardiono, and M. R. Sabil, "Using The SMART Method For Web-Based Employee Performance Measurement," *Gema Wiralodra*, 2024, doi: 10.31943/gw.v15i2.726.

[3] A. R. Albareta and P. Mursanto, "Design of Standard Operating Procedure for Requirement Engineering in Software Development," *J. Phys. Conf. Ser.*, 2019, doi: 10.1088/1742-6596/1175/1/012081.

[4] A. Rasheed, "Requirement Engineering Challenges in Agile Software Development," *Math. Probl. Eng.*, 2021, doi: 10.1155/2021/6696695.

[5] U. N. Mukharomah, T. Raharjo, and N. W. Trisnawaty, "Challenges and Solutions of Agile Software Development Implementation," *International Journal of Advanced Computer Science and Applications*, 2024, doi: 10.14569/IJACSA.2024.0150475.

[6] J. M. Verner, O. P. Brereton, B. A. Kitchenham, M. Turner, and M. Niazi, "Risks and risk mitigation in global software development," *Inf. Softw. Technol.*, 2014, doi: 10.1016/j.infsof.2013.06.005.

[7] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *IEEE Softw.*, 2008, doi: 10.1109/MS.2008.1.

[8] W. Alsaqaf, M. Daneva, and R. Wieringa, "Agile Quality Requirements Engineering Challenges: First Results from a Case Study," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, Nov. 2017, pp. 454–459. doi: 10.1109/ESEM.2017.61.

[9] I. M. Subrata, "Designing a rabies control mobile application," *Vet. World*, 2022, doi: 10.14202/vetworld.2022.1237-1245.

[10] H. U. Mahendra, "Tata Kelola Rekam Medis Berbasis Elektronik," *Jurnal Teknologi Sistem Informasi dan Aplikasi*, 2023, doi: 10.32493/jtsi.v6i4.32562.

[11] A. Young, "Parameterisation of domain knowledge for rapid and iterative prototyping," *Expert Syst. Appl.*, 2022, doi: 10.1016/j.eswa.2022.118169.

[12] B. Sieglin, "Rapid prototyping of advanced control schemes," *Fusion Engineering and Design*, 2020, doi: 10.1016/j.fusengdes.2020.111958.

[13] W. Behutiye, "Quality Requirement Documentation Guidelines," *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3187106.

[14] N. R. Tam, K. W. Prasetyo, and B. K. Kristanto, "Analisis Kebutuhan sistem informasi," *J-INTECH*, 2022, doi: 10.32664/j-intech.v10i1.675.

[15] Z. Hoy and X. Xu, "Agile Software Requirements Engineering Challenges-Solutions," *Information*, 2023, doi: 10.3390/info14060322.

[16] S. Chang, J. Gao, and W. Wang, "Automatic Generation of Software Prototype Data," *Electronics (Basel).*, 2025, doi: 10.3390/electronics14173497.

[17] G. Islam and T. Storer, "Agile software development for safety-critical systems," *Reliab. Eng. Syst. Saf.*, 2020, doi: 10.1016/j.ress.2020.106954.

[18] R. Wahyuni and Y. Irawan, "Web-Based Employee Performance Assessment System in PT. Wifiku Indonesia," *Journal of Applied Engineering and Technological Science*, 2020, doi: 10.37385/jaets.v1i2.62.

[19] R. A. Chrismanto, "Developing Agriculture Land Mapping using RAD," *International Journal of Advanced Computer Science and Applications*, 2019, doi: 10.14569/IJACSA.2019.0101033.

[20] R. A. Saputra, I. A. Kautsar, N. Ariyanti, and C. Aurusta, "Implementasi Progressive Web Apps," *SMATIKA*, 2024, doi: 10.32664/smatika.v14i01.1228.

[21] R. Purnamafajari, Budiman, and Z. Niqotaini, "Design of Management Information System for Employee Performance," *IOP Conference Series*, 2021, doi: 10.1088/1757-899X/1115/1/012013.