INTERNATIONAL MULTI DISCIPLINARY JOURNAL FOR RESEARCH & DEVELOPMENT
eISSN 2394-6334 Impact factor: 7,854 Volume 12, issue 12 (2025)

# Reconceptualizing Legacy System Modernization Through the Evolution of ASP.NET to ASP.NET Core: Architectural, Organizational, and Strategic Implications

**Adrian Michael Kovacs**
Faculty of Informatics, Eotvos Lorand University, Hungary

**Abstract:** Legacy systems have long occupied a paradoxical position within organizational information infrastructures, functioning simultaneously as indispensable operational backbones and as formidable barriers to technological innovation. Over several decades, scholarly discourse has framed legacy systems as repositories of accumulated business knowledge, technical debt, and organizational memory, while also identifying them as sources of rigidity, escalating maintenance costs, and strategic inflexibility (Bennett, 1995; Bisbal et al., 1999). In parallel, the rapid evolution of software platforms and architectural paradigms has intensified the pressure on organizations to modernize these systems without disrupting mission-critical operations. Within this context, the evolution of ASP.NET to ASP.NET Core represents not merely a technological shift but a paradigmatic reorientation in how enterprise software modernization can be conceptualized, executed, and governed. This article develops a comprehensive theoretical and analytical examination of legacy system modernization through the lens of the ASP.NET to ASP.NET Core transition, situating this evolution within broader traditions of software reengineering, service-oriented migration, and cloud-oriented architectural transformation.

Drawing on an extensive body of literature on legacy system evolution, reengineering methodologies, and digital transformation, this study adopts an interpretive research design grounded in qualitative synthesis and conceptual analysis (Kitchenham, 2004; Petersen et al., 2008). Rather than proposing new empirical data, the article systematically integrates established theoretical frameworks with contemporary platform evolution insights, most notably the detailed analysis of ASP.NET Core tools, strategies, and implementation approaches articulated by Valiveti (2025). By embedding this platform-specific evolution within long-standing scholarly debates on legacy modernization, the article advances a multi-layered understanding of how technological change intersects with organizational decision-making, architectural strategy, and socio-technical constraints.

The findings suggest that the ASP.NET Core ecosystem exemplifies a form of evolutionary modernization that challenges traditional dichotomies between replacement and incremental reengineering. Its cross-platform orientation, modular architecture, and cloud-native design principles provide a compelling case for reconsidering established modernization taxonomies such as wrapping, migration, and redevelopment (Canfora et al., 2000; Seacord et al., 2003). At the same time, the analysis underscores that technical capabilities alone are insufficient to guarantee successful modernization outcomes. Organizational readiness, governance structures, regulatory contexts, and domain-specific constraints—particularly evident in highly regulated sectors such as insurance and public administration—play a decisive role in shaping modernization trajectories (Irani et al., 2023; IAIS, 2024).

Through an extended discussion, the article critically evaluates competing scholarly perspectives on legacy system evolution, addresses common counterarguments regarding platform lock-in and migration risk, and articulates nuanced implications for future research and practice. Ultimately, the study contributes to the literature by positioning the ASP.NET to ASP.NET Core transition as an analytically rich exemplar of contemporary legacy modernization, offering theoretical insights that extend beyond a single technology stack and informing broader debates on sustainable software evolution in the digital era (Valiveti, 2025).

**Keywords:** Legacy systems, ASP.NET Core, software modernization, digital transformation, cloud migration, enterprise architecture

## INTRODUCTION

The Legacy information systems have been a central concern of software engineering research and practice since the early recognition that long-lived systems often outlast the technologies, assumptions, and organizational structures within which they were originally conceived (Bennett, 1995). The term "legacy system" itself has evolved from a narrow technical designation to a broader socio-technical construct encompassing outdated architectures, obsolete programming languages, entrenched business rules, and deeply embedded organizational practices (Bisbal et al., 1999). Despite their perceived obsolescence, such systems persist because they continue to deliver critical business value, often supporting core transactional processes that cannot be easily interrupted or replaced (Ransom et al., 1998). This enduring reliance has generated a rich body of scholarship aimed at understanding how legacy systems can be assessed, evolved, or transformed in response to changing technological and strategic demands.

Within this expansive literature, legacy system modernization has been framed as a multifaceted challenge involving technical complexity, economic trade-offs, and organizational risk (Seacord et al., 2003). Early studies emphasized coping strategies such as maintenance optimization and selective enhancement, recognizing that wholesale replacement was rarely feasible due to cost, risk, and knowledge loss (Weiderman et al., 1997). Over time, the emergence of distributed architectures, client–server computing, and later service-oriented and cloud-based paradigms reshaped both the problem space and the proposed solutions, introducing new opportunities for incremental migration and architectural decomposition (Canfora et al., 2000; Lewis et al., 2005). Yet, despite decades of research, the fundamental tension between stability and change remains unresolved, particularly as digital transformation initiatives accelerate across industries (Irani et al., 2023).

The evolution of web application frameworks provides a particularly illuminating lens through which to examine these enduring challenges. ASP.NET, introduced by Microsoft in the early 2000s, rapidly became a dominant platform for enterprise web applications, offering a unified programming model, tight integration with the Windows ecosystem, and strong tooling support. Over time, however, the framework itself accrued characteristics commonly associated with legacy systems, including monolithic architectures, platform dependence, and performance constraints that increasingly conflicted with emerging demands for scalability, portability, and cloud readiness (Valiveti, 2025). The introduction of ASP.NET Core marked a decisive break from this trajectory, representing a comprehensive redesign rather than a simple incremental upgrade.

Scholarly attention to ASP.NET Core has thus far focused primarily on its technical features and developer-centric benefits, such as cross-platform support, modular middleware, and improved performance characteristics (Valiveti, 2025). While these attributes are undeniably significant, they acquire deeper meaning when situated within the broader theoretical context of legacy system evolution. From this perspective, the transition from ASP.NET to ASP.NET Core can be interpreted as an instance of platform-level reengineering that encapsulates many of the principles advocated in the legacy modernization literature, including modularization, separation of concerns, and alignment with contemporary architectural paradigms (Fleurey et al., 2007). At the same time, this transition raises critical questions about continuity, backward compatibility, and the socio-organizational implications of adopting fundamentally new development models.

The introduction of ASP.NET Core also coincides with a period of intensified digital transformation across both private and public sectors, where legacy systems are increasingly perceived as obstacles to agility, innovation, and data-driven decision-making (Irani et al., 2023). In highly regulated domains such as insurance and public administration, the pressure to modernize is compounded by compliance requirements, risk management considerations, and the need for high levels of reliability and security (IAIS, 2024). These contextual factors underscore the importance of examining platform evolution not in isolation, but as part of a complex ecosystem of technological, organizational, and regulatory forces.

Despite the richness of existing research on legacy system modernization, there remains a notable gap in integrative analyses that connect general theoretical frameworks with concrete, contemporary platform evolutions. Much of the literature either remains at a high level of abstraction or focuses on isolated case studies, limiting its applicability to current technological contexts characterized by cloud computing, microservices, and continuous delivery (Jamshidi et al., 2013). Conversely, practitioner-oriented discussions of ASP.NET Core often emphasize tactical migration guidance without engaging deeply with the theoretical insights developed over decades of legacy system research (Valiveti, 2025). This disconnect constrains both scholarly understanding and practical decision-making.

The present article addresses this gap by offering a comprehensive, theoretically grounded examination of legacy system modernization through the specific case of the ASP.NET to ASP.NET Core evolution. By synthesizing insights from classical legacy system literature with contemporary analyses of platform redesign, the study seeks to demonstrate how long-standing concepts such as reengineering, migration strategy, and architectural assessment remain relevant, albeit in transformed ways, within modern development ecosystems (Bianchi et al., 2003; Lewis et al., 2005). The central argument advanced here is that ASP.NET Core exemplifies a form of evolutionary modernization that challenges simplistic narratives of replacement versus maintenance, instead illustrating how platform-level transformation can enable incremental, risk-managed evolution aligned with strategic objectives.

To develop this argument, the article adopts an interpretive methodology grounded in systematic literature synthesis and conceptual analysis, drawing on established procedures for rigorous secondary research in software engineering (Kitchenham, 2004; Petersen et al., 2008). The analysis is structured to progressively build from foundational theories of legacy systems, through methodological considerations of modernization, to an in-depth discussion of the ASP.NET Core case and its broader implications. Throughout, the study emphasizes critical engagement with competing perspectives, acknowledging both the potential and the limitations of platform-driven modernization approaches.

By situating the ASP.NET to ASP.NET Core transition within the broader discourse on legacy system evolution, this article aims to contribute to both theory and practice. For scholars, it offers a nuanced conceptual framework that bridges historical insights and contemporary technological realities. For practitioners and decision-makers, it provides an analytically informed perspective on how platform evolution can be leveraged as part of a coherent modernization strategy, rather than treated as a purely technical upgrade (Seacord et al., 2003; Valiveti, 2025). In doing so, the study underscores the enduring relevance of legacy system research in an era often characterized as post-legacy, revealing instead that legacy concerns have been reconfigured rather than eliminated.

## METHODOLOGY

The methodological orientation of this study is grounded in qualitative, interpretive research traditions within software engineering and information systems, where the objective is not empirical generalization through statistical inference but theoretical enrichment through systematic synthesis and critical analysis of existing knowledge (Kitchenham, 2004). Given the conceptual nature of the research problem—namely, understanding the evolution of ASP.NET to ASP.NET Core as a manifestation of broader legacy system modernization dynamics—a secondary research design based on literature integration is both appropriate and necessary (Petersen et al., 2008). This approach aligns with established practices in legacy system research, where the complexity and contextual embeddedness of phenomena often preclude controlled experimentation or large-scale quantitative studies (Bisbal et al., 1999).

The primary methodological strategy employed is an extended narrative synthesis that integrates insights from multiple strands of literature, including legacy system evolution, software reengineering, service-oriented migration, cloud architecture, and domain-specific digital transformation. Narrative synthesis, when conducted rigorously, enables the identification of conceptual patterns, tensions, and trajectories across heterogeneous sources, facilitating the development of higher-order theoretical interpretations (Kitchenham, 2004). In this study, such synthesis is guided by a set of analytically derived themes rather than predefined hypotheses, reflecting the exploratory and integrative intent of the research (Petersen et al., 2008).

The corpus of literature analyzed was defined strictly by the references provided, ensuring conceptual coherence and traceability of claims. These sources span several decades of scholarship, from early foundational analyses of legacy systems and maintenance challenges (Bennett, 1995; Weiderman et al., 1997) to more recent examinations of digital transformation, cloud migration, and platform evolution (Irani et al., 2023; Valiveti, 2025). By juxtaposing historical and contemporary perspectives, the methodology facilitates a longitudinal interpretation of how legacy system concerns have evolved in response to shifting technological paradigms.

Within this corpus, particular analytical emphasis is placed on works that articulate explicit frameworks or methodologies for legacy system assessment and modernization, such as iterative reengineering models, service-oriented migration techniques, and model-driven engineering approaches (Bianchi et al., 2003; Lewis et al., 2005; Fleurey et al., 2007). These frameworks serve as conceptual lenses through which the ASP.NET to ASP.NET Core transition is interpreted, allowing for systematic comparison between theoretical prescriptions and observed platform characteristics. The analysis also incorporates domain-oriented studies, including research on public administration and insurance systems, to contextualize modernization challenges within regulatory and organizational environments (Irani et al., 2023; IAIS, 2024).

A key methodological decision underpinning this study is the treatment of the ASP.NET to ASP.NET Core evolution as a conceptual case rather than an empirical case study in the traditional sense. This distinction is significant because it shifts the analytical focus from specific organizational outcomes to the structural and architectural properties of the platform itself and their alignment with established modernization principles (Valiveti, 2025). By doing so, the study avoids the pitfalls of anecdotal inference while still grounding its analysis in concrete technological realities.

The analytical process proceeded through several iterative stages. First, the literature was read and annotated to identify recurring themes related to legacy system challenges, modernization strategies, and architectural evolution (Bisbal et al., 1999; Seacord et al., 2003). Second, these themes were mapped against the documented features and design principles of ASP.NET Core, as described in contemporary analyses (Valiveti, 2025). This mapping exercise enabled the identification of points of convergence and divergence between theoretical expectations and platform implementation. Third, the mapped themes were critically examined to surface underlying assumptions, unresolved tensions, and implications for both research and practice (Bianchi et al., 2003).

Throughout this process, reflexivity was maintained to acknowledge the interpretive nature of the analysis and the potential for researcher bias. Rather than presenting the synthesis as a definitive account, the methodology emphasizes transparency in reasoning and openness to alternative interpretations, consistent with best practices in qualitative software engineering research (Kitchenham, 2004). This reflexive stance is particularly important given the normative dimensions of modernization discourse, where prescriptive recommendations often mask underlying value judgments about technological progress and organizational change (Irani et al., 2023).

The methodological limitations of this study stem primarily from its reliance on secondary sources and the absence of primary empirical data. While this approach enables broad theoretical integration, it necessarily constrains the ability to make claims about specific organizational outcomes or implementation success rates (Petersen et al., 2008). Additionally, the exclusive focus on the provided references limits the scope of perspectives considered, potentially omitting emerging viewpoints not captured within this corpus. However, this constraint also serves to enhance internal coherence and analytical depth, allowing for sustained engagement with a well-defined body of knowledge (Kitchenham, 2004).

Despite these limitations, the chosen methodology is well suited to the research objectives of this article. By leveraging systematic synthesis and conceptual analysis, the study provides a robust foundation for interpreting the ASP.NET to ASP.NET Core evolution within the broader landscape of legacy system modernization. This methodological rigor supports the subsequent presentation of results and discussion, ensuring that interpretive claims are firmly anchored in established scholarly discourse (Valiveti, 2025;

Seacord et al., 2003).

## RESULTS

The results of this study are presented as an interpretive synthesis of patterns and insights emerging from the integrated analysis of legacy system literature and contemporary discussions of the ASP.NET to ASP.NET Core evolution. Rather than reporting empirical measurements, the results articulate conceptual findings that illuminate how platform evolution embodies, challenges, or extends established theories of legacy system modernization (Bisbal et al., 1999; Valiveti, 2025). These findings are organized around several interrelated dimensions that collectively characterize the modernization dynamics observed.

One prominent result concerns the reconfiguration of architectural modularity as a central principle of modernization. Classical legacy system research has consistently emphasized the importance of decomposing monolithic systems into more manageable components to facilitate evolution and reuse (Canfora et al., 2000; Bianchi et al., 2003). The ASP.NET Core architecture reflects this principle through its explicit modular design, where middleware components, dependency injection, and lightweight runtime elements replace the tightly coupled structures characteristic of earlier ASP.NET implementations (Valiveti, 2025). This architectural shift aligns closely with theoretical prescriptions advocating incremental reengineering over wholesale replacement, suggesting that platform redesign can operationalize long-standing modernization ideals.

A second key result relates to platform independence and environmental flexibility. Legacy systems have often been criticized for their dependence on specific hardware, operating systems, or vendor ecosystems, which constrains organizational agility and increases migration costs (Bennett, 1995; Weiderman et al., 1997). The cross-platform nature of ASP.NET Core represents a significant departure from this pattern, enabling deployment across diverse environments including Linux-based servers and cloud infrastructures (Valiveti, 2025). This flexibility resonates with service-oriented and cloud migration frameworks that emphasize decoupling application logic from underlying infrastructure as a prerequisite for sustainable evolution (Lewis et al., 2005; Jamshidi et al., 2013).

The analysis also reveals a nuanced relationship between backward compatibility and innovation. Traditional modernization strategies often grapple with the trade-off between preserving existing functionality and embracing new paradigms, a tension extensively documented in legacy system research (Ransom et al., 1998; Seacord et al., 2003). The ASP.NET Core transition embodies a deliberate prioritization of innovation over strict backward compatibility, requiring developers to adapt codebases and development practices (Valiveti, 2025). This finding highlights an implicit shift in modernization philosophy, where controlled disruption is accepted as a means of achieving long-term architectural sustainability, challenging earlier assumptions that minimal change is always preferable (Bianchi et al., 2003).

Another significant result pertains to the socio-organizational implications of platform evolution. Legacy system literature has long acknowledged that technical change is inseparable from organizational learning, skill development, and governance structures (Bisbal et al., 1999; Irani et al., 2023). The adoption of ASP.NET Core necessitates not only technical retraining but also cultural shifts toward practices such as continuous integration, open-source collaboration, and cloud-native deployment (Valiveti, 2025). This observation reinforces the view that modernization outcomes are contingent upon organizational readiness and cannot be fully understood through technical analysis alone (Seacord et al., 2003).

Finally, the results underscore the contextual variability of modernization strategies across domains. Studies in public administration and insurance systems illustrate how regulatory requirements, risk aversion, and legacy data dependencies shape the feasibility and pace of modernization initiatives (Irani et al., 2023; IAIS, 2024). Within such contexts, the modular and incremental adoption pathways supported by ASP.NET Core may offer particular advantages, enabling selective modernization while maintaining compliance and operational continuity (Valiveti, 2025). This finding suggests that platform evolution can function as an enabling condition for context-sensitive modernization strategies rather than a one-size-fits-all solution.

Collectively, these results indicate that the evolution from ASP.NET to ASP.NET Core exemplifies a multifaceted modernization trajectory that aligns with, yet also extends, established theoretical frameworks in legacy system research. The findings provide a conceptual foundation for deeper discussion of the implications, limitations, and future research directions associated with platform-driven modernization (Bennett, 1995; Valiveti, 2025).

## DISCUSSION

The interpretive results presented above invite a deeper theoretical discussion that situates the ASP.NET to ASP.NET Core evolution within enduring scholarly debates on legacy system modernization, software architecture, and organizational change. This discussion engages critically with competing viewpoints, addresses potential counterarguments, and explores the broader implications of platform-level transformation for both research and practice (Seacord et al., 2003; Valiveti, 2025).

A central theme emerging from the discussion is the reconsideration of modernization taxonomies that have traditionally categorized strategies as replacement, reengineering, wrapping, or maintenance (Weiderman et al., 1997; Canfora et al., 2000). The ASP.NET Core transition does not fit neatly into any single category, instead embodying a hybrid approach that combines elements of redevelopment and incremental migration. From one perspective, the breaking changes introduced by ASP.NET Core could be interpreted as a form of redevelopment, given the need for substantial code adaptation (Bianchi et al., 2003). From another perspective, the continuity of the .NET ecosystem, tooling, and language support suggests an evolutionary pathway that preserves significant portions of existing knowledge and assets (Valiveti, 2025). This hybridity challenges the adequacy of traditional taxonomies and points to the need for more nuanced conceptual models that account for platform evolution as a distinct modernization mechanism.

The discussion also revisits the long-standing debate over the role of technology push versus organizational pull in driving modernization initiatives. Early legacy system research often portrayed modernization as a reactive response to accumulating technical debt and maintenance burden (Bennett, 1995; Bisbal et al., 1999). In contrast, contemporary analyses emphasize proactive transformation aligned with strategic objectives such as digital innovation, customer experience, and data analytics (Irani et al., 2023). The ASP.NET Core evolution reflects this shift, as it was motivated not solely by technical deficiencies but by a strategic reorientation toward open-source development, cross-platform compatibility, and cloud integration (Valiveti, 2025). This observation supports the argument that modernization is increasingly shaped by external technological ecosystems and competitive pressures rather than internal system decay alone (Jamshidi et al., 2013).

A further dimension of the discussion concerns the implications of open-source development models for legacy system evolution. Traditional enterprise platforms, including earlier versions of ASP.NET, were characterized by vendor-controlled release cycles and limited transparency (Seacord et al., 2003). The open-source nature of ASP.NET Core introduces new dynamics of community participation, rapid iteration, and shared ownership, which can both accelerate innovation and complicate governance (Valiveti, 2025). From a legacy system perspective, this shift raises important questions about control, accountability, and long-term sustainability, particularly for organizations operating in regulated environments (IAIS, 2024). While open-source ecosystems can reduce vendor lock-in, they also require new competencies in risk assessment and dependency management (Irani et al., 2023).

Critics of platform-driven modernization may argue that the benefits of ASP.NET Core are overstated, pointing to the costs and risks associated with migration, including potential disruptions to stable systems and the erosion of institutional knowledge embedded in legacy codebases (Ransom et al., 1998). This counterargument resonates with cautionary perspectives in the literature that warn against modernization initiatives driven by technological fashion rather than demonstrable business value (Bisbal et al., 1999). However, the analysis suggests that such risks can be mitigated through incremental adoption strategies that leverage the modular architecture of ASP.NET Core, allowing organizations to modernize selectively rather than wholesale (Valiveti, 2025). This aligns with iterative reengineering approaches that emphasize learning and adaptation over time (Bianchi et al., 2003).

The discussion further engages with the implications of cloud-oriented architectures for legacy system theory. Cloud migration research has highlighted both the opportunities and challenges associated with moving legacy applications to elastic, service-based environments (Jamshidi et al., 2013). ASP.NET Core's cloud-native design principles, including statelessness and containerization support, exemplify how platform evolution can facilitate alignment with cloud architectures (Valiveti, 2025). At the same time, this alignment necessitates rethinking assumptions about system boundaries, performance metrics, and operational responsibility, extending legacy system discourse into domains traditionally associated with infrastructure management rather than application design (Rao, 2012).

In highly regulated sectors such as insurance, the discussion must also account for compliance, security, and risk management considerations. Legacy systems in these domains often embody regulatory logic and audit trails that are difficult to disentangle (IAIS, 2024). The adoption of modern platforms like ASP.NET Core must therefore be evaluated not only in terms of technical elegance but also in relation to governance frameworks and assurance mechanisms (Dhieb et al., 2020). This perspective reinforces the argument that modernization is an inherently socio-technical process, requiring alignment between technological capabilities and institutional constraints (Irani et al., 2023).

Looking forward, the discussion identifies several avenues for future research. One promising direction involves comparative analyses of platform evolution across different ecosystems, examining whether the patterns observed in the ASP.NET Core case generalize to other frameworks and languages (Valiveti, 2025). Another avenue concerns longitudinal studies that track organizational outcomes over extended periods, providing empirical grounding for theoretical claims about the benefits and risks of platform-driven modernization (Petersen et al., 2008). Additionally, there is scope for deeper exploration of the human and organizational dimensions of modernization, including skill transformation, knowledge retention, and cultural change (Bisbal et al., 1999).

In sum, the discussion underscores that the evolution from ASP.NET to ASP.NET Core is not merely a technical upgrade but a rich empirical and conceptual instance of contemporary legacy system modernization. By engaging critically with established theories and emerging practices, this article contributes to a more integrated understanding of how legacy systems can evolve in alignment with the demands of the digital era (Seacord et al., 2003; Valiveti, 2025).

## CONCLUSION

The persistence of legacy systems as both enablers and constraints of organizational performance has ensured their continued relevance within software engineering scholarship and practice. This article has argued that the evolution of ASP.NET to ASP.NET Core offers a uniquely illuminating perspective on contemporary legacy system modernization, bridging historical theoretical frameworks and present-day technological realities (Bennett, 1995; Valiveti, 2025). Through an extensive interpretive synthesis of established literature and platform-specific analysis, the study has demonstrated that platform evolution can function as a viable and strategically meaningful pathway for modernization, challenging simplistic binaries between maintenance and replacement.

By situating ASP.NET Core within broader discourses on modularization, service orientation, and cloud migration, the article has shown how long-standing modernization principles are rearticulated in modern development ecosystems (Canfora et al., 2000; Jamshidi et al., 2013). At the same time, the analysis has emphasized that technical innovation alone is insufficient, highlighting the decisive role of organizational readiness, governance, and contextual constraints in shaping modernization outcomes (Irani et al., 2023; IAIS, 2024). These insights reinforce the enduring value of legacy system research as a lens for understanding digital transformation, even as the technologies themselves continue to evolve.

Ultimately, this study contributes to the field by offering a theoretically grounded, critically engaged account of platform-driven modernization, inviting scholars and practitioners alike to reconsider how legacy systems are conceptualized and addressed in the digital era. The ASP.NET to ASP.NET Core transition, as examined here, underscores that legacy is not merely a condition to be eliminated but a dynamic context within which innovation unfolds (Seacord et al., 2003; Valiveti, 2025).

## REFERENCES

**1.** Bianchi, A., Caivano, D., Marengo, V., and Visaggio, G. Iterative reengineering of legacy systems. IEEE Transactions on Software Engineering, 29(3), 225–241.

**2.** International Association of Insurance Supervisors. Global Insurance Market Report – Mid-Year Update 2024. IAIS Market Reports, July 2024.

**3.** S. S. Sravanthi Valiveti, "Evolution of ASP.NET to ASP.NET Core: Tools, Strategies, and Implementation Approaches," 2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS), Bangalore, India, 2025, pp. 1-7, doi: 10.1109/ICITEICS64870.2025.11341480.

**4.** Bisbal, J., Lawless, D., Wu, B., and Grimson, J. Legacy information systems: Issues and directions. IEEE Software, 16(5), 103–111.

**5.** Jamshidi, P., Ahmad, A., and Pahl, C. Cloud migration research: A systematic review. IEEE Transactions on Cloud Computing, 1(2), 142–157.

**6.** Weiderman, N., Smith, D., and Tilley, S. Approaches to legacy system evolution. Software Engineering Institute Technical Report CMU/SEI-97-TR-014.

**7.** Irani, Z., Abril, R. M., Weerakkody, V., Omar, A., and Sivarajah, U. The impact of legacy systems on digital transformation in European public administration: Lessons learned from a multi-case analysis. Journal of Strategic Information Systems, 40(1), 101784.

**8.** Canfora, G., Cimitile, A., De Lucia, A., and Di Lucca, G. A. Decomposing legacy programs: A first step towards migrating to client–server platforms. Journal of Systems and Software, 54(2), 99–110.

**9.** Seacord, R. C., Plakosh, D., and Lewis, G. A. Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices. Addison-Wesley.

**10.** Bennett, K. Legacy systems: Coping with success. IEEE Software, 12(1), 19–23.

**11.** Lewis, G., Morris, E., and Smith, D. Service-oriented migration and reuse technique (SMART). Proceedings of the IEEE International Workshop on Software Technology and Engineering Practice, 222–229.

**12.** Fleurey, F., Breton, E., Baudry, B., Nicolas, A., and Jezequel, J. M. Model-driven engineering for software migration in a large industrial context. In Model Driven Engineering Languages and Systems, 482–497.

**13.** Ransom, J., Somerville, I., and Warren, I. A method for assessing legacy systems for evolution. Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering, 128–134.

**14.** Kitchenham, B. Procedures for performing systematic reviews. Keele University Technical Report, 1–26.

**15.** Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. Systematic mapping studies in software engineering. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, 68–77.

**16.** Chung, S., An, J., and Davalos, S. Service-oriented software reengineering: SOSR. Proceedings of the Hawaii International Conference on System Sciences.

**17.** Rao, P. C. Architecting the Cloud: Enterprise Architecture Patterns for Cloud Computing.

**18.** Dhieb, N., Ghazzai, H., and Besbes, H. A secure AI-driven architecture for automated insurance systems: Fraud detection and risk measurement.

.