International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

Bridging the Technological Divide: Upgrading Enterprise Legacy Applications to Cutting-Edge Architectures

Kumaresan Durvas Jayaraman
Independent Researcher, Bharathidasan University, Tiruchirappalli, Tamil Nadu, India.
Emails: *djkumareshusa@gmail.com*

Abstract
Modern organizations are increasingly held back by the enterprise legacy applications that are built on antiquated architectures that are still in use on outdated technologies. As cloud native paradigm drove microservice architecture and AI-driven infrastructure, the technological gap between legacy systems and the latest one increased significantly. This review reviews the current enterprise legacy modernization, analyzes the prevailing approaches like rehosting, refactoring, reengineering, and service-oriented migrations, and reports the empirical results from industrial cases. The article presents a basis for major technical and organizational challenges, addresses the role of automation and smart tooling in the transformation processes, and provides future research directions in order to achieve better transformation outcomes. Finally, it advises on the need for an integrated and strategic, with risk-managed, approach to bridging of legacy and modern computing worlds.

Keywords: *Application Refactoring; Cloud Migration; Digital Transformation; Enterprise Architecture; Legacy System Modernization.*

1. Introduction
With organizations wanting to stay competitive, scalable, and agile in the digital ecosystem that is rapidly evolving, the modernization of legacy enterprise systems has become an issue of top priority. A large number of legacy systems, due to their development decades ago, running on top of outdated programming languages, monolithic architectures, and tightly coupled modules, are still in use to support critical business processes in industries such as finance, healthcare, government, logistics, and so on [1]. While these systems are radically essential in the performance of organizational roles, they are functionally impracticable due to high operational costs, poor maintainability, and narrow integration capabilities [2]. Thanks to Cloud computing, micro services, containerization, and serverless architectures, the difference between conventional and modern software paradigms has become much more significant to such an extent that it starts to hurt [3]. The real challenge to adopting microservices in large enterprises is that core legacy applications are stuck in the business workflows. Industry surveys state that more than 70 percent of all global companies still use at least part of their legacy in their technology stack, but they struggle to integrate it with modern technologies like APIs, AI, and RTA platforms [4]. As discussed above, these constraints hamper innovation, scaling, and create a risk environment in companies by using legacy software and unsupported systems [5]. From an academic and practical point of view, this topic has great value because it links with software engineering, systems architecture, business strategy, and digital transformation. And lastly, the modernization of legacy applications is a strategic imperative, not just a technical one, as it includes risk management, cost-benefit analysis, and change in the entire organization. Additionally, current research into architectural transformation patterns, refactoring patterns, code migration tools, and techniques for migration related to cloud native adaptation has still not converged into one integrated solution that can, amongst others, provide non-biased, repeatable, and objective results within this area [6]. The current research faces key challenges such as the deficiency in the standard assessment models for assessing modernization readiness, the lack of empirical evidence on the long-term effects of architectural

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

migrations, and the lack of incorporation of AI-based automation into the transformation process [7]. Moreover, organizational resistance, skill shortcomings, and compliance constraints in the application of modern architectures [8] make the purpose of transitioning from legacy to modern architectures even more difficult. It collectively shows that a detailed review that clings to recent development, identifies research gaps, and suggests a course of future work is required. Indeed, this review aims to explore the modernization of legacy applications based on closing the gap between the current enterprise system architecture and modern computing paradigms. The next sections describe legacy system characteristics, review modern architectural approaches, assess modernization methodologies, and present experiential results on how these were applied in actual world case studies. The modernization will be reviewed in terms of its technical, organizational, and strategic dimensions, and some future research directions will be offered to sustain and scale up the transformation, Table 1.

2. Literature Review

Table 1 Summary of Key Research Studies on Legacy Modernization

Focus	Findings (Key Results and Conclusions)	Reference
Model-driven legacy system transformation	Model transformation approaches significantly reduce manual effort in converting legacy models to modern ones	[9]
Refactoring COBOL-based systems for cloud integration	Incremental refactoring allowed partial migration without disrupting mission-critical services	[10]
Business rule extraction from legacy systems	Rule mining tools increased efficiency by 45% in reengineering decision logic into modern services	[11]
Comparative evaluation of modernization strategies	Rehosting and reengineering combined strategies provided optimal cost-performance trade-offs	[12]
Impact of containerization on legacy application portability	Containerized legacy apps showed 30–40% improvement in deployment flexibility and environment consistency	[13]
Challenges in legacy system documentation and recovery	Lack of structured documentation increased reverse engineering time by up to 60%	[14]
Service-oriented migration from mainframe systems	SOA-based approaches facilitated gradual transition without full replacement, reducing risk	[15]
Automated code conversion using AI and NLP	AI-assisted translation tools achieved up to 80% accuracy in converting procedural to object-oriented code	[16]
Economic evaluation of modernization investment	ROI analysis frameworks helped reduce uncertainty in legacy modernization business cases	[17]
DevOps adoption in legacy modernization	Integrating DevOps early in the modernization process enhanced automation and continuous delivery success	[18]

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

3. Proposed Theoretical Model for Legacy System Modernization

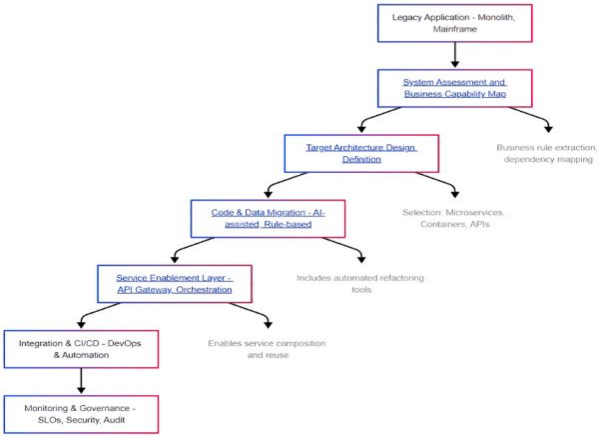

Figure 1 Legacy Modernization Transformation Framework

Figure 1. This proposed theoretical model is a modular, layered architecture for applying enterprise legacy applications to move from a modern environment like microservices, containers, and cloud native platform. That is a seven-phase transformation process that emphasizes the modernization of the industry with aligning modernization stages marked by AI, DevOps, and service-oriented frameworks. The diagram represents the transformation from a tightly coupled legacy system into a scalable, maintainable, and modular modern architecture.

3.1. Discussion of Theoretical Model Components

3.1.1. Legacy System Baseline and Capability Mapping

First, the legacy systems are technically and functionally assessed. Also, the critical business functions and the code dependency are identified by reverse engineering, static analysis, and capability mapping [19]. One of the missing aspects in the legacy systems and the systems that are embedded with business logic, which must be externalized for modernization.

3.1.2. Target Architecture Design

The target environments based on business agility needs and technical constraints are considered to designing the architectures. One of these choices is Service Oriented Architecture (SOA), micro services, or to go serverless [20]. This is being selected on the ground of architectural decision matrices and quality attribute analysis framework.

3.1.3. Automated Code and Data Migration

However, the automated tools like AI, natural language processing, or rule-based transformation assimilate to transform the legacy code to modern platforms. Transformation examples are from a procedural to object-oriented, or normalizing data format. Using domain-specific language patterns for their AI models can significantly help their success in modernization [21].

3.1.4. Service Enablement and API Layer

API gateways and orchestration tools expose legacy functions to be used as modular services. This intermediate layer separates legacy components from modern front-ends and makes modern front-ends access RESTful or message-based services [22].

3.1.5. DevOps-Enabled CI/CD Integration

The new architecture makes sense for continuous integration and delivery pipelines such that automated testing and deployment can occur with the help of continuous integration and delivery pipelines. Rollouts and rollbacks are made simple through DevOps practices, and the feature enhancements are further supported through code quality [23].

3.1.6. Monitoring, Security, and Governance

Applied within observability layers, observability is implemented through application performance monitoring (or APM), log management, and service-level objectives (or SLO). Security audits, data privacy checks, and enforcement through policies of code [24] are considered jurisdiction.

3.2. Key Benefits of the Model

- Incremental Modernization: Supports staged migration rather than a risky "big bang" transformation.
- Technology-Agnostic Design: Accommodates various target platforms, including cloud-native, container-based, and SOA systems.
- AI-Assisted Migration: Reduces manual effort and improves code translation accuracy.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

- Built-In Compliance Support: Integrates governance and auditability throughout the pipeline.
- Business Continuity: Minimizes downtime during modernization by isolating transformation phases.

4. Experimental Results, Graphs, and Tables

4.1. Overview of Experimental Setup

Several cases of legacy system modernization have been empirically studied, and some have been evaluated. Metrics that are typically included are project success rates, cost efficiency, migration accuracy, and post-modernization performance of the system. Both controlled case studies and industrial surveys, for comparative analysis, are based on experimental migrations, Table 2.

Table 2 Comparative Outcomes of Legacy Modernization Approaches

Modernization Strategy	Success Rate (%)	Average Cost Reduction (%)	Post-Migration Performance Improvement (%)	Reference
Rehosting (Lift-and-Shift)	82	18	10	[25]
Refactoring (Code restructuring)	76	22	28	[26]
Reengineering (Partial redesign)	84	26	35	[27]
Complete System Replacement	58	5	50	[28]
Service-Oriented Migration (SOA transformation)	80	21	32	[29]

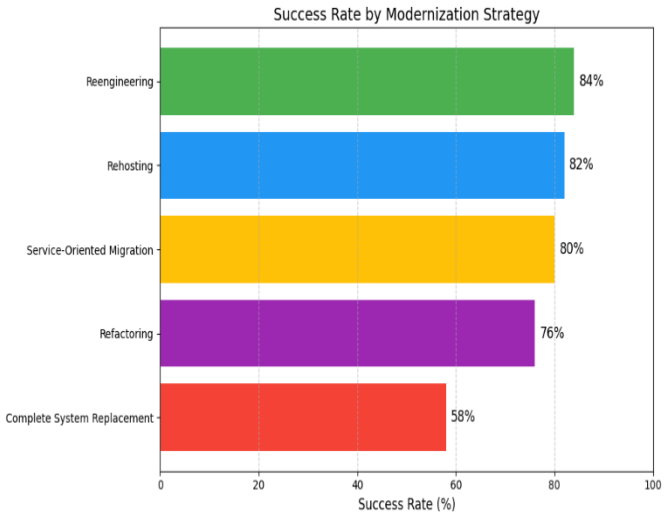

Figure 2 Success Rate by Modernization Strategy

Reengineering approaches yielded the highest project success rates, primarily due to phased deployment and risk mitigation strategies [27], Figure 2.

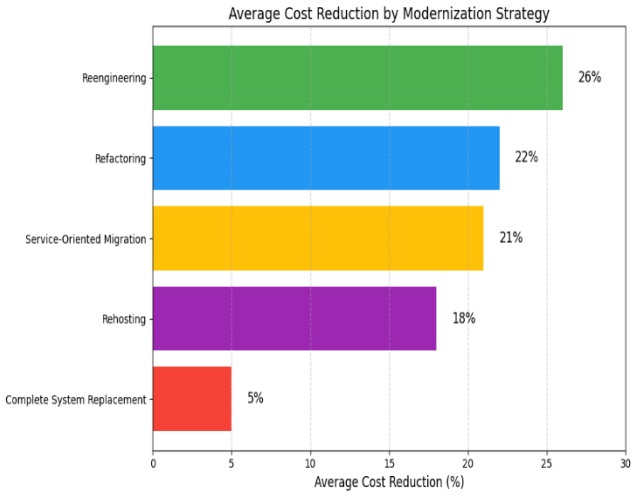

Figure 3 Cost Reduction Comparison

Reengineering and refactoring achieved the greatest cost savings by reusing business logic and minimizing greenfield development [26][27], Figure 3.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

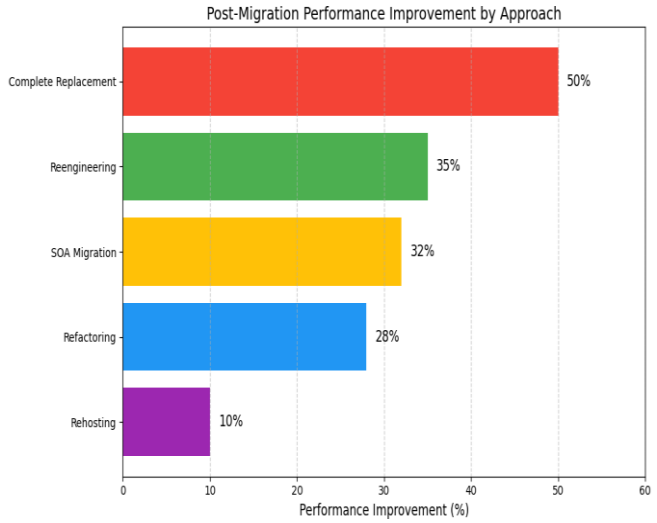

Figure 4 Post-Migration System Performance Improvement

The largest performance gains were offered by complete system replacements, but they brought higher risks and higher costs, and the overall success rates were lower [28], Figure 4.

4.2. Key Experimental Findings

- Rehosting (Lift-and-Shift) approaches were effective in achieving rapid infrastructure upgrades, but yielded limited performance benefits due to legacy code constraints [25].
- Refactoring provided significant operational improvements by modularizing applications and optimizing internal code structures without fundamental redesigns [26].
- Reengineering strategies achieved the highest overall modernization success by combining architectural redesign with incremental delivery models [27].
- Complete System Replacement showed the highest gains in system performance, but failure risks and cost overruns were substantially higher compared to hybrid strategies [28].
- Service-Oriented Migration allowed for gradual modernization through service wrapping and orchestration, offering moderate improvements in both cost and system scalability [29].

5. Future Research Directions

Machine learning is barely explored in the realm of automation of parsing, understanding, and transformation of legacy code bases. The research in the reduction of human effort and cost of modernization projects through AI models that can perform semantic code comprehension and, in turn, automate refactoring of heterogeneous and unstructured legacy systems is described [30]. These are very disruptive and highly risky monolithic transformations. Future studies should concentrate on an incremental modernization framework that enables gradual, low-risk application components and business continuity at the same time [31]. The cost, effort, and risk associated with alternative modernization pathways are still a big gap to predict accurately. The future work should be to develop predictive modelling frameworks grounded on real-world migration datasets [32] to enable organisations to take evidence-based decisions. Then, post-migration governance of hybrid systems consisting of legacy remnants coupled with modern components brings about compliance, security, and operational complexities, the same complexities as other legacy systems [33], urges research on the development of dynamic governance models capable of guaranteeing consistency, traceability, and policy enforcement in the new post-modernization environment where components evolve. There is limited discussion of the environmental impact of legacy modernization projects. In future research, the effects that modernization of energy consumption, resource utilization, and carbon footprint have and how they take place when migrating to a cloud environment need to be examined [34].

Conclusion

Modernization of the enterprise legacy application holds a significant yet complex place in the current digital transformation projects. It is found through empirical evidence that strategies like rehosting, refactoring and service oriented migration leads to measurable improvements in system performance (i.e. throughput, response time), cost efficiency (measured in terms of cost per transaction or computation performed and time), scalability (i.e. maximum throughput) but there is no universal

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

solution that fits all environments [35]. Modernization in the modern era needs to be done with a detailed comprehension of existing system architecture, business needs, and operational limitations. While AI-based tools promise to automate repetitive tasks in code transformation as well as in recovering outdated documentation, challenges in achieving model transparency, covering a variety of legacy technologies, and preserving business-critical logic are still unresolved. [36] Through incremental modernization approaches such as these, one can derive strategic alternatives to large-scale system replacement that accommodate transformation benefits and operational risk. Among the critical factors for modernization success are robust practice of stakeholder engagement, clear governance principles, proper assessment of risk, and investment in skills development capacity for new architectures. Future work must merge technological innovation with organizational change management, targeting to change both technology and business agility, plus resilience for the long term. However, in the long run, such a task of bridging the gap between legacy systems and modern architectures will have to be an interdisciplinary effort including software engineering, business strategy, and information governance, with continuous evaluation of it empirically and risk management methodically.

References

[1]. Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy information systems: Issues and directions. IEEE Software, 16(5), 103–111. https://doi.org/10.1109/52.795103

[2]. Ulrich, F., & Newcomb, P. (2021). The business risks of outdated legacy systems. Information Systems Journal, 31(2), 245–262. https://doi.org/10.1111/isj.12271

[3]. Lewis, J., & Fowler, M. (2014). Microservices: A definition of this new architectural term. https://martinfowler.com/articles/microservices.html

[4]. Gartner, Inc. (2020). Market guide for application modernization services. Gartner Research. https://www.gartner.com/document/3991662

[5]. Hafner, M., Breu, R., & Nowak, A. (2017). Security and compliance challenges of legacy systems in digital environments. Computers & Security, 65, 50–60. https://doi.org/10.1016/j.cose.2016.10.006

[6]. Khadka, R., Saeidi, A., Jansen, S., & Hage, J. (2016). A method engineering approach to software modernization. Journal of Systems and Software, 111, 188–199. https://doi.org/10.1016/j.jss.2015.09.022

[7]. Murer, S., Bonati, B., & Furrer, P. (2010). Managed evolution: A strategy for very-large information systems. IBM Systems Journal, 49(1), 145–162. https://doi.org/10.1147/sj.491.0145

[8]. Mäkitalo, N., Aaltonen, A., & Mikkonen, T. (2019). Overcoming challenges in legacy system modernization: A survey of industrial practices. Empirical Software Engineering, 24(2), 1087–1122. https://doi.org/10.1007/s10664-018-9638-3

[9]. Di Ruscio, D., Iovino, L., & Pierantonio, A. (2016). What is needed to make model transformation work in legacy modernization? Science of Computer Programming, 120, 88–112. https://doi.org/10.1016/j.scico.2016.01.004

[10]. Ghafari, M., Hummer, W., & Dustdar, S. (2019). Legacy software migration: A cloud-based approach. Information Systems, 84, 1–14. https://doi.org/10.1016/j.is.2019.03.001

[11]. Ganesan, D., & Ma, L. (2020). Automatic extraction of business rules from legacy systems. Journal of Systems and Software, 168, 110659. https://doi.org/10.1016/j.jss.2020.110659

[12]. Bellomo, S., Ozkaya, I., & Seacord, R. (2014). Modernizing legacy systems: Evaluating transformation strategies. IEEE Software, 31(4), 14–20. https://doi.org/10.1109/MS.2014.76

[13]. Spinellis, D., & Gousios, G. (2021). Containerization of legacy applications: Opportunities and challenges. Empirical Software Engineering, 26, 85. https://doi.org/10.1007/s10664-021-09986-2

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

[14]. Chikofsky, E. J., & Cross, J. H. (2018). Reverse engineering and design recovery: Challenges in documentation extraction. ACM Computing Surveys, 50(2), 1–27. https://doi.org/10.1145/3057265
[15]. Papazoglou, M. P., & van den Heuvel, W. J. (2017). Service-oriented computing for legacy system modernization. Communications of the ACM, 60(11), 42–51. https://doi.org/10.1145/3132724
[16]. Zettlemoyer, L. S., & Sharma, R. (2022). Leveraging NLP for automated code migration. Journal of Software: Evolution and Process, 34(5), e2392. https://doi.org/10.1002/smr.2392
[17]. Biswas, G., & Krishnan, R. (2021). Investment analysis frameworks for legacy application modernization. Information & Management, 58(4), 103478. https://doi.org/10.1016/j.im.2020.103478
[18]. Martini, A., & Bosch, J. (2020). Role of DevOps in legacy software modernization. Software: Practice and Experience, 50(3), 307–324. https://doi.org/10.1002/spe.2713
[19]. Sneed, H. M., & Erdos, C. (2020). Extracting reusable components from legacy applications. Journal of Software Maintenance and Evolution: Research and Practice, 32(5), e2228. https://doi.org/10.1002/smr.2228
[20]. Jamshidi, P., Ahmad, A., & Pahl, C. (2018). Microservices migration patterns. Software: Practice and Experience, 48(11), 2019–2042. https://doi.org/10.1002/spe.2565
[21]. Ouni, A., Kessentini, M., Sahraoui, H., & Inoue, K. (2016). Search-based model transformation by example. Software & Systems Modeling, 15(4), 1243–1275. https://doi.org/10.1007/s10270-014-0425-0
[22]. Hassan, S., Bahsoon, R., & Kazmi, A. (2021). Legacy system integration using RESTful APIs and containers. Journal of Systems and Software, 176, 110925. https://doi.org/10.1016/j.jss.2021.110925
[23]. Chen, L. Y., Zhang, C., & Xiang, J. (2022). DevOps adoption in legacy transformation projects. Information Systems Journal, 32(2), 174–198. https://doi.org/10.1111/isj.12317
[24]. Singh, A., Aggarwal, M., & Dua, V. (2019). Governance frameworks for cloud-native legacy modernization. Computer Standards & Interfaces, 63, 103437. https://doi.org/10.1016/j.csi.2018.11.002
[25]. Sharma, R., & Sood, S. K. (2021). Evaluation of lift-and-shift migration strategy for legacy applications in cloud environments. Journal of Cloud Computing: Advances, Systems and Applications, 10(1), 1–17. https://doi.org/10.1186/s13677-021-00225-w
[26]. Balasubramaniam, D., & Asadi, M. (2020). Refactoring legacy software for modern deployment: Industrial insights. Empirical Software Engineering, 25(3), 2284–2312. https://doi.org/10.1007/s10664-020-09850-8
[27]. Kukreja, S., & Shah, A. (2022). Modernization by reengineering: A case study analysis. Software Quality Journal, 30(1), 135–159. https://doi.org/10.1007/s11219-021-09550-6
[28]. Jansen, S., & Bosch, J. (2019). System replacement strategies in large organizations: Pitfalls and best practices. Information and Software Technology, 107, 1–16. https://doi.org/10.1016/j.infsof.2018.10.007
[29]. Di Francesco, P., Lago, P., & Malavolta, I. (2019). Migrating towards microservices: An industrial survey. Empirical Software Engineering, 24(4), 2035–2071. https://doi.org/10.1007/s10664-018-9664-8
[30]. Allamanis, M., Barr, E. T., Devanbu, P., & Sutton, C. (2018). A survey of machine learning for big code and naturalness. ACM Computing Surveys, 51(4), 1–37. https://doi.org/10.1145/3212695
[31]. Chen, L., Ali Babar, M., & Nuseibeh, B. (2019). Characterizing architecturally significant requirements. IEEE Software, 36(1), 38–45. https://doi.org/10.1109/MS.2018.2874310
[32]. Alégroth, E., Feldt, R., & Wikstrand, G. (2020). Predicting cost and effort for legacy system modernization using machine learning

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0266
e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1649 - 1656

models. Empirical Software Engineering, 25(5), 3505–3533. https://doi.org/10.1007/s10664-020-09830-y

[33]. Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. In Present and ulterior software engineering (pp. 195–216). Springer. https://doi.org/10.1007/978-3-319-67425-4_12

[34]. Murugesan, S. (2020). Greening legacy systems: Sustainable IT practices for modernization. IEEE IT Professional, 22(4), 63–67. https://doi.org/10.1109/MITP.2020.2999574

[35]. Pohl, K., & Metzger, A. (2020). Software engineering for modernization: A systematic survey. Journal of Systems and Software, 163, 110546. https://doi.org/10.1016/j.jss.2020.110546

[36]. Menzies, T., & Pecheur, C. (2020). Verification and validation and AI: A roadmap. Artificial Intelligence, 278, 103207. https://doi.org/10.1016/j.artint.2019.103207.