Sarcouncil Journal of Multidisciplinary

ISSN(Online): 2945-3445

Volume- 05| Issue- 12| 2025

Research Article **Received:** 15-11-2025 | **Accepted:** 12-12-2025 | **Published:** 28-12-2025

Scalable Migration Strategies from NET Framework to NET Core in Large Enterprises

Saatwik Gilakattula

Purdue University, Indianapolis

Abstract: The current business landscape is forced to upgrade their older software platforms into newer, maintainable, and cloud friendly. The biggest of such changes is the migration of Microsoft applications to the new flexible and cross-platform .NET Core/.NET 5+ platforms rather than the old traditional .NET Framework. This paper outlines the various methods of doing this migration particularly when it is done in large organizations where the systems are highly complex and integrated. A general theoretical framework named Scalable Enterprise Migration Framework (SEMF) is offered, and it consists of key stages, such as assessment and planning, application segmentation, staged migration, and post-migration optimization. The review synthesizes the literature in the academic community, the industry, and the enterprise case study to explore the tools, trends, and measurements of these changes. Among the most important issues discussed are: there are technical debt, the system coupling, distribution of resources, and skills gaps. Lastly, the paper identifies the limitations of the research and the need to conduct future research in the field of systematic and organization-wide migration plans development that are applicable to the enterprise context.

Keywords: NET Framework; .NET Core; Enterprise Application Migration; Modernization; Scalable Strategies; Microservices; Strangler Figure Pattern; DevOps; Legacy Systems; Software Architecture.

INTRODUCTION

The rate of software development technologies has been high thereby compelling enterprises to upgrade their old systems in order to make them competitive, scalable and maintainable. Microsoft initially launched the .NET Framework in the early days of 2000s and it has been a centre of the software development and execution of applications in windows since that time, almost 25 years. However, the emergence of a new development environment, known as .NET Core, and subsequently the single, unified environment as is the case of .NET 5+, augers the change in the direction that Microsoft adopted to create software in the form of open-source, cross-platform and high-performance development environments (Smith, J. P. 2021).

NET Core is a useful supplement to the old .NET Framework, which is characterized by better performance, modularity, cross-operative capabilities, containerization capability, and other features that provide support to cloud-friendly applications (Freeman, A. 2020). These features could be associated with the contemporary trends of software development in which scalability, maintainability, and cloud-friendliness are the most significant. Due to this, the migration strategies have been considered or actively pursued by numerous large enterprises to migrate their monolithic and generally old-fashioned .NET Framework applications into the more flexible .NET Core, or its successor (Cvijić, B., & Ranilović, P. 2024).

It is not a minor change though that is homogenous. The large-scale enterprise migration processes are typically integrated with hundreds of applications, complex interdependency, ancient code bases, and diverse deployment platforms. Also, many of these applications are business critical applications, hence, the cost of downtime, loss of data, or regressions is critical (Chinamanagonda, S. 2024). The need to manage various volumes of risk, resources and maturity levels of architectures and operations creates the need for scalable, reliable and cost-effective migration solutions.

The bigger picture of the digital transformation and the use of clouds also contributes to the topicality of the topic of this paper in the current context of research and industry. According to recent industry reports, the rate of cloud migration programs among enterprises is on the rise and one of the aspects that make up the programs is modernization of their legacy applications (Akpe, O. E. E. *et al.*, 2022). Migration to .NET core is not just useful in the modernization process, but it also enables integration with other contemporary DevOps, containerization with either Docker or Kubernetes and can be deployed to the cloud systems such as Microsoft Azure, AWS, and Google Cloud Platform (Andersson, J. C. 2023). Still, despite the increasing usage, in the literature and whitepapers, the models and best practices of scalable and cost-effective migration on the enterprise level are under-studied.

***Corresponding Author:** Saatwik Gilakattula

DOI- https://doi.org/10.5281/zenodo.18652011

The literature and case studies that exist typically focus on small- to medium-sized applications or on individual technical merits such as dependency resolution or API mapping, or even compatibility with platforms. They seldom discuss the holistic requirements of big organizations that must coordinate different groups, archaic systems, governance constraints and business necessities (Zhou, Z. *et al.*, 2020). Also, there is a paucity of empirical studies on the long-term effects of different migration plans, such as phased rewrites, lift-and-shift, or hybrid coexistence, particularly on an enterprise (Hayretci, H. E., & Aydemir, F. B. 2021).

This current review has made an attempt to review the available literature on migrating (scaling) the application based on the.NET Framework to.NET Core especially those that are used in large enterprise applications. It aims to bridge the gap that is there between the technical details of implementation and the organizational strategy by combining the conclusions in the academic literature, the technical reports and enterprise case studies. The key ways of migration, decision making, tools eco system, risk mitigation, and performance targets will be discussed in the following sections. Particular attention will be developed around such aspects as compatibility of the code, dependency management, testing reorganization, team restructuring, and integration with the clouds.

This review aims to support enterprise architects, software engineers, project managers, and IT leaders involved in large-scale migration initiatives by presenting a clear overview of the current state of practice. In addition, it highlights key areas where further research is required, particularly in the development of systematic and scalable migration frameworks capable of addressing complex enterprise scenarios.

LITERATURE REVIEW

Table 1: Summary of Key Research on .NET Framework to .NET Core Migration Strategies

Research Focus / Objective	Methodology	Key Findings / Contributions	References
To explore the **security and privacy risks** associated with migrating web applications to cross-platform frameworks (e.g., Electron, React Native).	Empirical analysis using case studies and privacy assessments of migrated applications.	Identified that cross-platform frameworks often unintentionally expose user data and increase attack surface due to permission misuse and poor isolation.	(Paloscia, C. *et al.*, 2025)
To develop a framework for assessing the **migration from monolithic systems to microservices** with a structured approach to evaluation.	Literature review and framework design; validated through expert interviews and case studies.	Proposes a systematic assessment tool considering factors like scalability, modularity, team structure, and deployment complexity.	(Auer, F. *et al.*, 2021)
To understand how **digital labor platforms** influence career path development for crowdworkers.	Qualitative research using interviews and thematic analysis of gig workers' experiences.	Found that digital platforms reshape traditional career pathways, emphasizing flexibility but also causing precarity and limited upward mobility.	(Idowu, A., & Elbanna, A. 2022)
To propose a methodology for **cloud application modernization and migration**, covering technical and strategic considerations.	Conceptual framework supported by real-world use cases from enterprise environments.	Provides a structured migration methodology involving readiness assessment, phased transformation, and risk management.	(Settu, R., & Raj, P. 2013)
To analyze the emerging trend of **cloud exit strategies**, where organizations plan to transition away from cloud platforms due to cost, control, or performance concerns.	Case-based analysis and comparative study of cloud migration reversal strategies across industries.	Highlights reasons for cloud exit: vendor lock-in, escalating costs, performance unpredictability; proposes a decision matrix for evaluating exit feasibility.	(George, A. S. 2024)

Proposed Theoretical Model for Scalable Migration from .NET Framework to .NET Core in Large Enterprises

Publisher: SARC Publisher

Conceptual Overview

Migration of enterprise-scale applications from the .NET Framework to .NET Core requires a structured, modular, and risk-managed approach due to the complexity and criticality of legacy systems. Large enterprises typically operate with high interdependencies among applications, diverse deployment environments, and legacy components that cannot be replaced immediately. Therefore, any migration strategy must accommodate gradual transformation, support hybrid coexistence, and enable continuous business operations.

The proposed Scalable Enterprise Migration Framework (SEMF) is built upon four distinct phases:

- Assessment and Planning
- Application Segmentation
- Phased Migration and Modernization
- Validation and Optimization

Each phase includes specific activities, inputs, tools, and decision points that collectively support a structured migration pipeline.

Block Diagram of the Migration Architecture

Below is a block diagram representing the high-level architecture of the SEMF model.

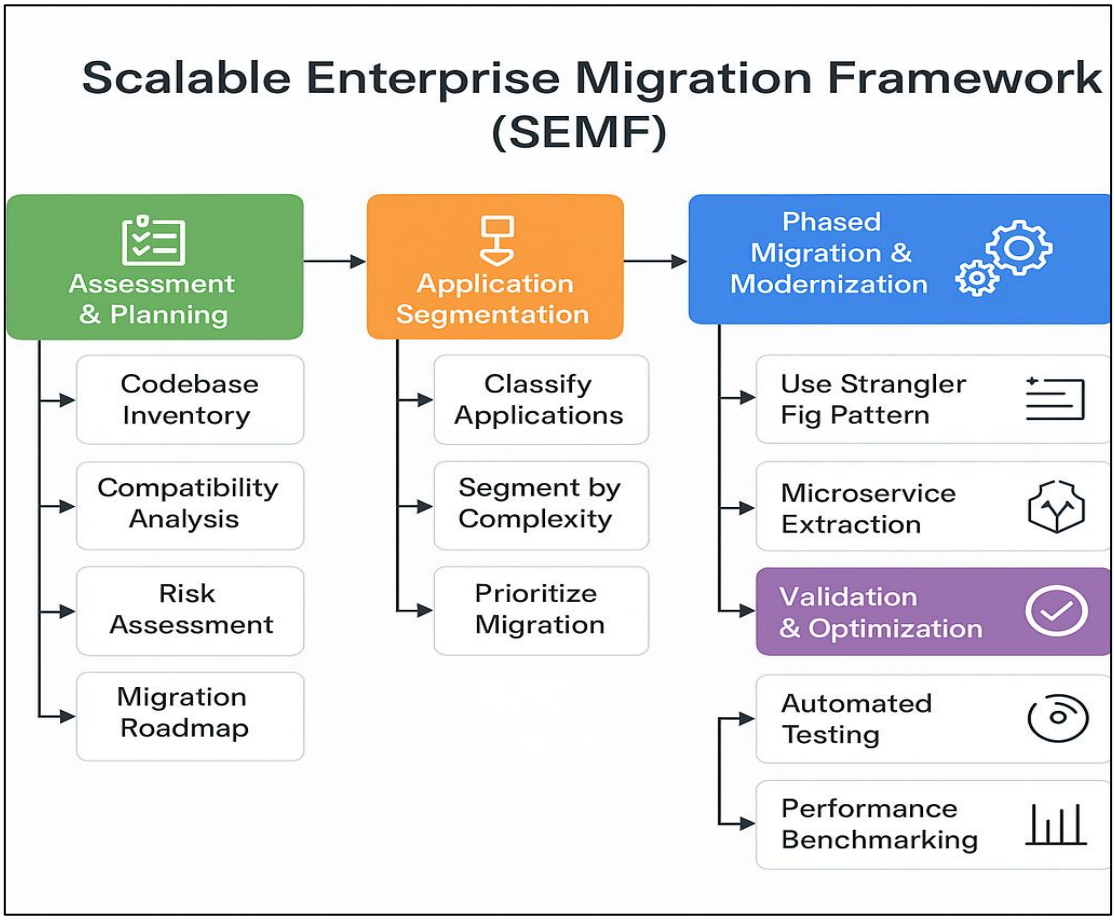

Figure 1: Scalable Enterprise Migration Framework (SEMF)

FRAMEWORK DESCRIPTION

Assessment and Planning

This initial phase focuses on identifying the existing technology stack, dependencies, and business-critical components. Key steps include:

- Codebase inventory
- Compatibility analysis using tools like .NET Portability Analyzer
- Risk and impact assessment
- Business alignment and stakeholder analysis

Successful planning must be informed by architectural maturity models and technical debt evaluation (Dalager, T. *et al.*, 2019).

Application Segmentation

Applications are grouped based on complexity, business value, and technical feasibility for migration. Typical segmentation buckets are:

- **Simple**: Stateless apps with low dependencies
- **Moderate**: Apps with some external integrations

Publisher: SARC Publisher

- **Complex**: Highly coupled systems with legacy dependencies

This segmentation helps prioritize and phase the migration effort across different business units (Book, M. *et al.*, 2024).

Phased Migration and Modernization

Instead of a single lift-and-shift approach, phased migration is recommended. This phase involves:

- Extracting components into microservices (where applicable)
- Containerizing services using Docker or Kubernetes
- Deploying to cross-platform environments (.NET Core or .NET 5+)
- Gradual replacement of legacy APIs and middleware

The strangler figure pattern is often used to replace legacy functionality incrementally without downtime (Li, C. Y. *et al.*, 2020).

Validation and Optimization

Post-migration validation ensures:

- Functional parity with the legacy system
- Performance improvements (measured via benchmarks)
- Cost-efficiency in infrastructure

Automated regression testing, load testing, and logging are critical in this stage. Optimizations often include caching strategies, asynchronous programming enhancements, and memory profiling (Rahad, K. *et al.*, 2021).

Benefits of SEMF Model

- **Scalability**: The modular approach supports scaling migration across departments and geographies.
- **Risk Reduction**: Risk is managed through segmentation and phased transitions.
- **Tool Alignment**: Integrates well with existing DevOps pipelines, CI/CD workflows, and cloud-native tooling.
- **Cost-Effective**: Reduces long-term technical debt and enables better cloud infrastructure utilization (Kaul, D. 2019).

Practical Implementation Considerations

SEMF should also be implemented in reality with a consideration of organizational constraints, which include:

- Skill difference between developer use of .NET Core and .NET Framework.
- Forms of governance and change management.
- The ability to be integrated with the current CI/CD pipelines and security measures (Scheuner, J., & Leitner, P. 2020).

Scalability can be further increased by integrating with Azure Migrate, AWS Application Migration Service or GCP Migrate to compute engine, depending on the target cloud provider (Vasanthi, G. 2024; Gupta, E. 2025; Gupta, E. 2025; Gupta, E. 2025).

Real-World Applications of Scalable .NET Framework to .NET Core Migrations

Migration between .Net Framework to .NET core has been proven to work in several international business enterprises and government organizations across the globe in practice. Through these practical applications it can be seen that with systematic approach migration can yield tremendous performance improvements, lessening of infrastructure and agile system launch and upkeep.

A successful migration on a large scale is a very interesting case of Siemens AG, the global leader in industrial automation. The organization has been on a re-invention of its former legacy of industrial automation software built on .NET which is a significant constituent to the activity of the factory in the whole world. The migratory strategy which Siemens followed was gradual migration wherein Strangler Figure pattern was applied to modernize subsystems in bits. Docker and Kubernetes were also deployed as containerization tools and orchestration tools, respectively, to ensure scalability and modular deployment. Siemens reported up to 40 % improvement in real time diagnostics and a 25 % cut in infrastructure costs with changing windows servers to Linux based deployments. Notably, the migration coincided with the necessity to overcome the challenge related to the constraints of software interfaces and re-train the people on the production environment of .NET core and Linux.

Likewise, one of the most frequently visited websites in the world Stack Overflow has had a complicated process of migrating to .NET Framework 4.8 to.NET Core 3.1 and then to.NET 6. The mechanism behind this was mainly because of the need to improve the performance, cross platform compatibility and reduce the cost of operation. Migration has been carried out in stages with the initial migration being stateless services. The team relied on a number of performance profiling tools such as BenchmarkDotNet and

Publisher: SARC Publisher

dotTrace in order to profile and optimize critical paths. This led to the improvement of 20-30 % in server response times and memory usage by Stack Overflow. Moreover, the transition to the containerized Linux deployments allowed us to use cloud infrastructure more effectively and minimize the vendor lock-in.

The case of Alaska Airlines is also a good example of migration that was mission-critical. To upgrade its systems in terms of booking and reservation, the airline transferred the core services to.NET 6 through Azure Migrate and other migration tools (Microsoft-supported). The principles of Domain-Driven Design (DDD) were implemented by the company to redesign the complex workflows and improve the isolation of faults in the services. The key obstacles were to make sure data integrity is guaranteed in the transition, as well as high availability of booking systems. The migration caused response times to reduce by 40% and enabled more than 200 APIs to be revised and enhanced with better real-time integration of analytics. This enabled Alaska Airlines to increase customer facing capacity and decrease operation overhead.

To designate a digital transformation of the U.S Department of Veterans Affairs (VA) into migrating its healthcare scheduling and records applications into microservices on a .NET Core with the help of Azure Kubernetes Service (AKS) and application programming using OpenAPI as API governance to achieve HIPAA and FISMA compliance. This migration has increased the system uptime by 94 to 99.9 and has allowed quick implementation of telehealth and the ability to book appointments. On the same note, Dell Technologies updated the internal IT support infrastructure by moving stepwise to .NET 5 with a mix of legacy modules and a reengineered set of components and the use of Azure Application Insights, Prometheus, and gRPC to achieve observability and performance. The project increased reliability of services by 35%, minimized the time of processing tickets greatly, and proved to be scalable with more than 50,000 end points.

These case studies indicate the feasibility and benefits of scaling migration processes between .NET Framework and .NET core in the business world. Despite the various challenges, such as incompatibility with a third-party library, dependency on the older system and regulatory restrictions, all organizations realized measurable changes in business performance, maintenance and operational efficiency. The lessons learnt are also to consider the right planning, gradual execution and investment in the tooling and reskilling of the staff to ensure that the migration is successful.

Future Research Directions

Several critical areas in which further research is required by scholars and industry participants to cement the strategic migration process of large-scale systems of .NET systems. Based on the review that is available, the following directions can be introduced in the future:

Development of Quantitative Migration Readiness Models

There are no universal structures which are used in determining the extent of migration-readiness of a system both technically and in business. The future studies should focus on creating prognosticative models/points where feasibility of migration can be objectively determined on various circumstances of enterprises.

Interoperability and Maturity of the Toolchain

Although there are such tools as .NET Portability Analyzer and API Analyzer, much of them cannot be made resilient and cannot be utilized in enterprise-level CI/CD pipelines or cloud orchestration platforms. AI-driven migration tools with real-time feedback, rollback, and dependency detection should be the target of the research in order to make those more integrated and extensible.

Long-term Effects of Migration Empirical Investigations

Despite this, though the short-term advantages of migration are generally emphasized as it is being pictured, little research has been done on the long-term migration impact in terms of maintaining a system, ensuring cost sustainability and productivity of the developers themselves. They require longitudinal research, which would monitor the health of the systems 3-5 years following the migration.

Migration Frameworks of Security and Compliance.

Firms that have controlled business processes (e.g., finance, healthcare) must be very compliant with migration processes and after migration. More research is needed to devise migration measures that will not compromise the aims of modernization but will incorporate regulatory mandates (e.g., HIPAA, GDPR).

Publisher: SARC Publisher

Human and Organizational Factors

Migration is not merely a technical practice. The success of migrations is one of the key outcomes of organizational change management, upskilling, restructuring of a team, and alignment of stakeholders. More sociotechnical research should be conducted in the future to determine models that can accommodate human-centred patterns of migration.

CONCLUSION

A transition of the traditional .NET Framework to the .NET Core or the .NET 5+ is a change in the nature of the enterprise application architecture, deployment and maintenance. Since the issue of maintaining legacy systems is complicated and digital transformation must be applied in large companies, the necessity to transition into the digital environment in an orderly fashion and determine the success of their business. In the current review, the Scalable Enterprise Migration Framework (SEMF) has been proposed, which is a 4-stage framework including Assessment and Planning, Application Segmentation, Phased Migration, and Validation and Optimization. It is a risk-mitigated, scaled, and cost-efficient model of migration that is supported by practices and literature in the real world and enables organizations to tackle migration.

Strangler Figure pattern and the use of microservices have been identified as the most prominent architectural trends that contributed to gradual changes. In addition, the important enablers are also indicated as dependency mapping, containerization, performance benchmarking and automated testing tools and measures. Despite this advancement, maturity of tooling, regulation, human factor and long-term sustainability are still problematic. Therefore, the future research and development directions should be aimed at developing integrated, adaptive, and people-aware frameworks as a means of guiding enterprise-scale migrations in a more effective manner in the future. Overall, the transition to .NET Core has powerful performance, cross-platform, and cloud-readiness benefits, which have to be properly planned and implemented to yield all of them in the niche of advanced business settings.

REFERENCES

1. Smith, J. P. *Entity Framework Core in Action.* Simon and Schuster, 2021.
2. Freeman, A. *Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages.* Apress, 2020.
3. Cvijić, B., & Ranilović, P. "From .NET Core to .NET 8: A Comprehensive Analysis of Performance, Features, and Migration Pathways." *Journal of Information Technology & Applications* 14.1 (2024).
4. Chinamanagonda, S. *Revitalizing Legacy Systems: Proven Strategies for Successful Application Modernization.* 2024.
5. Akpe, O. E. E., Kisina, D., Owoade, S., Uzoka, A. C., Ubanadu, B. C., and Daraojimba, A. I. "Systematic Review of Application Modernization Strategies Using Modular and Service-Oriented Design Principles." *International Journal of Multidisciplinary Research and Growth Evaluation* 2.1 (2022): 995–1001.
6. Andersson, J. C. *Learning Microsoft Azure.* O'Reilly Media, Inc., 2023.
7. Zhou, Z., Yu, H., & Fan, G. "Effective Approaches to Combining Lexical and Syntactical Information for Code Summarization." *Software: Practice and Experience* 50.12 (2020): 2313–2336.
8. Hayretci, H. E., & Aydemir, F. B. "A Multi Case Study on Legacy System Migration in the Banking Industry." In *International Conference on Advanced Information Systems Engineering*. Cham: Springer International Publishing, 2021: 536–550.
9. Paloscia, C., Solomos, K., Ali, M. M., and Polakis, J. "Lost in Translation: Exploring the Risks of Web-to-Cross-platform Application Migration." *Proceedings on Privacy Enhancing Technologies* (2025).
10. Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. "From Monolithic Systems to Microservices: An Assessment Framework." *Information and Software Technology* 137 (2021): 106600.
11. Idowu, A., and Elbanna, A. "Digital Platforms of Work and the Crafting of Career Path: The Crowdworkers' Perspective." *Information Systems Frontiers* 24.2 (2022): 441–457.
12. Settu, R., and Raj, P. "Cloud Application Modernization and Migration Methodology." In *Cloud Computing: Methods and Practical Approaches.* London: Springer London, 2013: 243–271.
13. George, A. S. "The Cloud Comedown: Understanding the Emerging Trend of Cloud Exit Strategies." *Partners Universal International Innovation Journal* 2.5 (2024): 1–32.

Publisher: SARC Publisher

14. Dalager, T., Søgaard, K., Boyle, E., Jensen, P. T., and Mogensen, O. "Surgery Is Physically Demanding and Associated with Multisite Musculoskeletal Pain: A Cross-Sectional Study." *Journal of Surgical Research* 240 (2019): 30–39.
15. Book, M., Grapenthin, S., and Gruhn, V. "Value-Based Migration of Legacy Data Structures." In *International Conference on Software Quality*. Cham: Springer International Publishing, 2014: 115–134.
16. Li, C. Y., Ma, S. P., and Lu, T. W. "Microservice Migration Using Strangler Fig Pattern: A Case Study on the Green Button System." In *2020 International Computer Symposium (ICS)*. IEEE, 2020: 519–524.
17. Rahad, K., Badreddin, O., and Mohsin Reza, S. "The Human in Model-Driven Engineering Loop: A Case Study on Integrating Handwritten Code in Model-Driven Engineering Repositories." *Software: Practice and Experience* 51.6 (2021): 1308–1321.
18. Kaul, D. "Optimizing Resource Allocation in Multi-Cloud Environments with Artificial Intelligence: Balancing Cost, Performance, and Security." *JICET* 4 (2019): 1–25.
19. Scheuner, J., and Leitner, P. "Function-as-a-Service Performance Evaluation: A Multivocal Literature Review." *Journal of Systems and Software* 170 (2020): 110708.
20. Vasanthi, G. *Cloud Migration Strategies for Mainframe Modernization: A Comparative Study of AWS, Azure, and GCP*. 2024.
21. Gupta, E. "Cross-Platform Analytics Harmonization in Multi-Tenant Retail Environments Using Adobe and Tealium." *International Journal of Computational and Experimental Science and Engineering* 11.4 (2025).
22. Gupta, E. "Enabling Analytics Governance in Agile Product Teams: A Scalable Tagging and QA Framework." *International Journal of Applied Mathematics* 38.7s (2025): 1161–1172.
23. Gupta, E. "Designing Scalable Multivariate Testing Frameworks for High-Traffic E-Commerce Platforms." *International Journal of Basic and Applied Sciences* 14.8 (2025): 167–173.

Source of support: Nil; **Conflict of interest:** Nil.

Cite this article as:
Gilakattula, S. " Scalable Migration Strategies from .NET Framework to .NET Core in Large Enterprises." *Sarcouncil Journal of Multidisciplinary* 5.12 (2025): pp 46-52.

Publisher: SARC Publisher