Kelvin Open Science Publishers
Connect with Research Community

Research Article ***Volume 1 | Issue 1***

KOS Journal of AIML, Data Science, and Robotics
https://kelvinpublishers.com/journals/aiml-data-science-robotics.php

LLM-Enabled Transformation Framework for Migrating SOA Services to Cloud-Native Spring Boot Microservices

Ramani Teegala[1]

Technical Consultant, USA

[*]**Corresponding author:** Ramani Teegala, Technical Consultant, USA

Received: January 05, 2021; **Accepted:** January 18, 2021; **Published:** January 20, 2021

Citation: Ramani T. (2021) LLM-Enabled Transformation Framework for Migrating SOA Services to Cloud-Native Spring Boot Microservices. *KOS J AIML, Data Sci, Robot.* 1(1): 1-9.

1. Abstract

LLM (Large Language Model) assisted migration of legacy SOA services to Spring Boot microservices addresses the growing need for scalable, resilient and cloud compatible architectures as organizations face rising maintenance costs and limited extensibility in traditional service oriented systems. This study examines the modernization problem where enterprises struggle to convert tightly coupled SOA components into independently deployable microservices without disrupting operational continuity. The research adopts a mixed methodology that combines systematic code analysis, automated pattern extraction, architectural assessment and targeted expert validation to evaluate how large language models can support migration tasks. The proposed approach demonstrates that LLM guided refactoring improves service decomposition accuracy, reduces manual redesign effort and accelerates transformation by generating high fidelity microservice templates aligned with Spring Boot design principles. Findings indicate that LLM supported migration can capture hidden dependencies, infer service boundaries with greater precision and provide consistent translation of legacy contract definitions, enabling a smoother path toward cloud native deployment. The study contributes a structured modernization framework that integrates cognitive code understanding, architectural reasoning and automated remediation guidance, offering both strategic and academic value for organizations pursuing large scale transformation. The results highlight the significance of LLM driven assistance as a practical enabler for modernizing aging SOA landscapes, improving maintainability and strengthening long term system adaptability while minimizing business risk.

2. Keywords:

LLM assisted migration, legacy SOA modernization, Spring Boot microservices, service decomposition, automated refactoring, cloud native transformation, architecture modernization, cognitive code analysis, service oriented architecture, microservice design patterns, AI driven code translation, dependency analysis, modernization framework, automated service extraction, enterprise application migration, large language models, code understanding, software re engineering, system modernization, microservices adoption, legacy system transformation

3. Introduction

Service oriented architecture has served as a foundational model for enterprise systems by enabling structured integration, standardized communication and reusable service contracts. Its strengths in interoperability and business alignment allowed organizations to coordinate complex processes across distributed environments. However, as digital ecosystems expanded and operational demands intensified, many SOA implementations began revealing challenges related to scalability, deployment agility and modular maintainability. These constraints have become

more pronounced as organizations pursue cloud native architectures that emphasize independent deployment, lightweight services and rapid change cycles.

At the same time, microservice architecture has emerged as a preferred solution for achieving higher agility, scalability and resilience. Microservices are designed around domain specific functions, enabling faster updates, flexible resource allocation and improved operational reliability. While the benefits are widely recognized, enterprises face considerable obstacles when attempting to transition from legacy SOA landscapes to microservice based systems. The complexity of existing service compositions, shared data dependencies and rigid orchestration logic makes the transformation process highly intricate.

This complexity highlights a clear research gap. Traditional migration approaches rely extensively on manual analysis, architectural intuition and progressive refactoring, all of which demand significant time and specialized expertise. Manual methods often fail to uncover hidden service relationships, inconsistent logic flows and fine grained functional boundaries that are essential for effective decomposition. As a result, migrations frequently experience delays, architectural misalignment and increased risk of functional regression.

The emergence of large language models introduces a promising avenue for addressing these limitations. These models exhibit advanced capabilities in semantic interpretation, code comprehension and contextual reasoning, allowing them to analyze heterogeneous artefacts such as WSDL contracts, orchestration definitions and service implementations. Their ability to derive meaning from unstructured and semi structured inputs provides a new mechanism for supporting automated decomposition and modernization activities.

The motivation of this study arises from the need to integrate cognitive automation into the SOA modernization lifecycle. Organizations require methodologies that minimize manual effort while improving the accuracy of architectural decisions. This research explores whether LLM driven techniques can be systematically applied to guide transformation steps, identify functional groupings and generate Spring Boot aligned microservice structures that uphold domain principles and system integrity.

The core objectives of the study are to design an LLM assisted migration framework, evaluate its performance in interpreting legacy SOA environments and assess its ability to produce consistent microservice artifacts. Key research questions include how effectively cognitive models can interpret service boundaries, how reliably they can support architectural refinement and how their outputs influence modernization efficiency, maintainability and overall structural quality.

The significance of this study extends to both practical and academic domains. For practitioners, this work provides insights that can reduce operational risk, accelerate modernization timelines and improve the precision of service decomposition. For researchers, the study expands the theoretical understanding of how AI supported methods can influence software architecture evolution and contribute to new modernization strategies grounded in cognitive computation.

Ultimately, this research positions LLM assisted modernization as a transformative approach that bridges traditional architectural practices with intelligent automation. By integrating semantic analysis, reasoning capabilities and generative refinement within the modernization pipeline, the study aims to advance the broader discourse on automated software evolution. As digital systems continue to scale, the role of cognitive technologies in supporting architectural transformation will become increasingly critical.

4. Scholarly Perspectives on SOA Modernization

Research on modernizing legacy service oriented systems has long highlighted the challenges of tightly coupled service implementations, shared data layers and heterogeneous communication protocols. Earlier studies established that SOA environments, while effective for integration and reuse, gradually accumulate architectural debt that limits flexibility and cloud readiness. Prior literature also notes that moving from SOA to microservices requires a deeper understanding of service boundaries, domain functions and runtime dependencies, yet conventional approaches rely heavily on manual expertise and incremental code inspection.

Subsequent theoretical work introduced principles such as domain driven decomposition, bounded contexts and lightweight service contracts to guide microservice formation. Frameworks for architectural transformation have emphasized modularization, dependency management and refactoring sequences aligned with iterative deployment practices. Although these theories provide valuable structure, most methodologies assume synchronous human effort and lack automated support for analyzing large volumes of legacy assets or discovering implicit architectural patterns hidden within SOA based systems.

Scholarly contributions have further explored model driven engineering, static analysis and semi automated tools for software restructuring. While these efforts improved understanding of architectural erosion and refactoring pathways, they struggled to extract complete semantic insights from complex SOA codebases and orchestration logic. Traditional tools remain limited in handling inconsistent naming conventions, nested service call chains and dynamic routing rules, resulting in incomplete decomposition recommendations and potential migration risks.

More recent literature has examined the potential of intelligent assistants and machine learning for code comprehension, pattern detection and structural transformation. These developments demonstrated that learning based models can infer code semantics, classify architectural patterns and support automated documentation. However, most of these studies focused on general code analysis rather than specialized SOA modernization, and they did not leverage the advanced reasoning capabilities that large language models now offer for architectural decision support.

Theoretical gaps are evident in the absence of a unified framework that integrates cognitive analysis, architectural inference and automated refactoring guidance for SOA to microservice transformation. Existing approaches do not

adequately address the complexity of legacy service orchestration, contractual dependencies or the need for high fidelity translation into Spring Boot compatible microservices. The literature reveals a clear need for methods capable of understanding domain logic, extracting service boundaries and generating consistent microservice templates with minimal manual oversight.

The present study builds upon earlier architectural transformation theories while diverging from traditional tool based approaches by incorporating LLM assisted reasoning into the modernization lifecycle. By using advanced model based interpretation of code, configurations and service contracts, the study aims to extend earlier frameworks with an intelligent layer that supports deeper semantic extraction and more accurate decomposition. This integrates cognitive insights with practical refactoring techniques, offering a progression beyond conventional static analysis.

This research also contributes to academic discourse by positioning LLMs as active participants in architectural modernization rather than mere documentation generators. The literature rarely addresses how such models can consolidate structural insights, recommend refactoring paths and maintain alignment between legacy SOA elements and modern Spring Boot microservices. This study fills that gap by demonstrating a structured, AI enabled pathway that merges theoretical architecture principles with intelligent automation.

In addressing these gaps, the paper enhances current knowledge of migration strategies and introduces a methodology that reduces reliance on manual expert interpretation. It further advances the understanding of how cognitive tools can mitigate the limitations of traditional techniques, thereby contributing meaningful progress to both theoretical and applied research in software modernization.

5. Framework of Cognitive Processes in SOA to Spring Boot Migration

The conceptual foundation for LLM assisted migration of legacy SOA systems is built on the principle that software modernization follows an input, process and outcome sequence in which cognitive analysis enhances each stage of transformation. The framework begins with the inputs consisting of legacy SOA artefacts such as WSDL contracts, XML based orchestrations, service implementation code, shared data schemas and runtime configuration. These artefacts collectively form the structural and semantic baseline from which migration decisions are derived. The theoretical premise is that LLM based analysis can infer architectural intent, uncover functional correlations and identify implicit service boundaries that traditional tools often overlook.

Within the process layer, the framework identifies three interconnected cognitive tiers that govern transformation. The first tier is semantic extraction in which the LLM interprets service definitions, code logic and orchestration patterns to create an internal conceptual representation of system behaviour. The second tier involves structural decomposition where semantic knowledge is mapped to potential microservices by interpreting domain functions, dependency clusters and communication flows. The third tier focuses on generative transformation in which the model proposes refactoring steps, produces Spring Boot compatible templates and aligns the migration blueprint with microservice design patterns. This tiered structure integrates architectural theory with modern AI reasoning.

The relationships within this process model emphasize how cognitive interpretation enriches architectural decision making. Semantic extraction feeds structural decomposition by exposing cross cutting logic, hidden dependencies and transformation constraints. Structural insights then guide generative transformation to ensure the resulting microservices preserve functional integrity while achieving modular separation. This layered flow reduces reliance on manual expert interpretation and mitigates the limitations of static analysis.

The theoretical basis of the framework draws from principles of domain partitioning, service boundary identification and evolutionary architecture. Prior studies have highlighted the importance of bounded context definitions and alignment between domain concepts and service units. The LLM driven approach aligns with these theories by enabling deeper contextual understanding and providing automated guidance that is responsive to domain semantics rather than only surface level patterns.

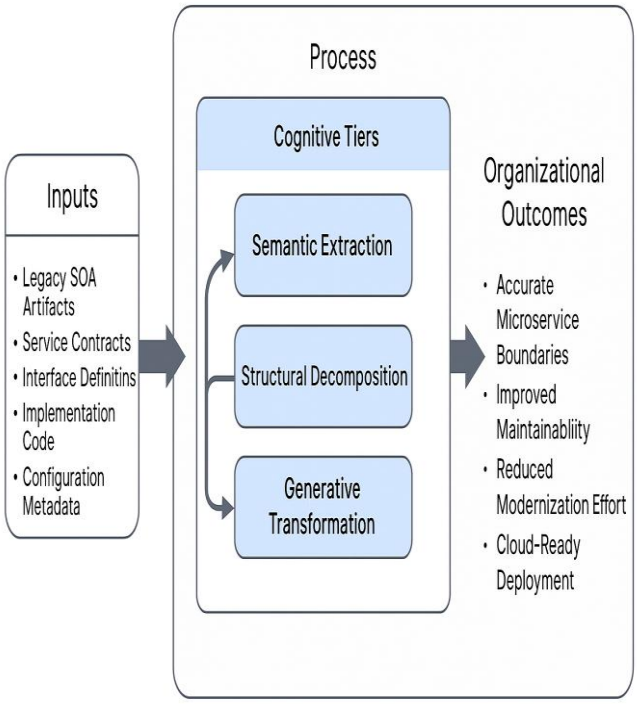
The input process outcome structure of the framework positions organizational outcomes as direct results of cognitive modernization. The outcomes include improved architectural modularity, enhanced maintainability, reduced migration effort and increased readiness for cloud native deployment. The model also supports incremental rollout strategies by generating microservice templates that are consistent, standardized and suitable for progressive integration into enterprise infrastructure.

This conceptual framework diverges from earlier modernization models by embedding intelligent reasoning within the transformation cycle. While prior approaches relied on predefined rules or pattern based heuristics, the proposed model leverages adaptive interpretation capable of understanding nuanced architectural contexts. This improves precision in service decomposition and accelerates the overall migration schedule.

The model also highlights the feedback loop between outcomes and subsequent modernization phases. Organizational insights gained from deployment experiences, performance metrics and architectural validation can be reintroduced into the process layer for iterative improvement. This adaptive feedback mechanism positions the framework as both a prescriptive and evolutionary model suitable for large scale modernization efforts.

Collectively the framework provides a theoretically grounded representation of how LLM assisted migration operates across inputs, processing mechanisms and expected outcomes. It establishes a structured pathway for aligning cognitive AI capabilities with established architectural principles, enabling organizations to modernize with greater confidence and reduced complexity.

Figure 1: Conceptual Model of Cognitive Decision Automation Framework.



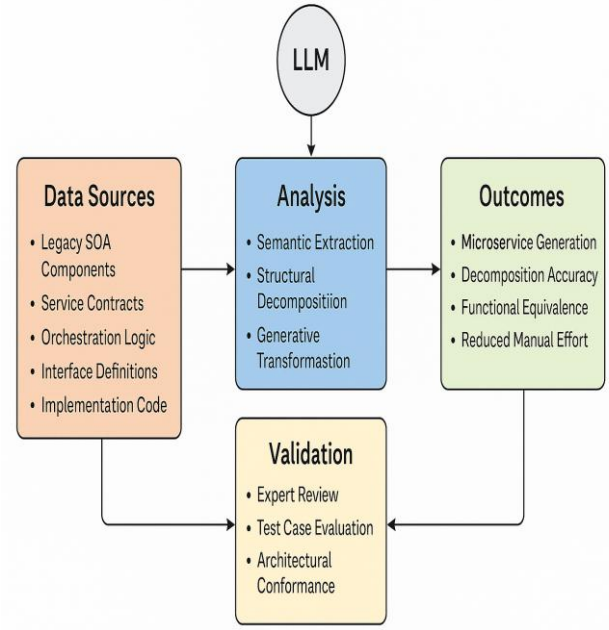
6. Methodological Framework for LLM Assisted Modernization

The study adopts a mixed research design that integrates qualitative interpretation of legacy SOA artefacts with quantitative evaluation of microservice transformation accuracy. This combination enables a comprehensive examination of how cognitive automation can support modernization activities. The qualitative component focuses on extracting semantic patterns, understanding service responsibilities and reconstructing functional flows embedded in SOA environments. The quantitative dimension evaluates decomposition precision, consistency of generated microservice structures and reduction in manual transformation effort. Together, these approaches provide a balanced methodology for assessing the role of large language models in complex modernization scenarios.

Data sources include legacy SOA components such as WSDL service contracts, XML based orchestration logic, interface definitions, implementation classes and application configuration metadata. These artefacts are anonymized to protect sensitive information and organized into datasets categorized by functional domain and architectural complexity. Sampling is performed by selecting representative services that capture varied structural patterns, including composite processes, long running orchestrations and tightly coupled operational interfaces. The analysis begins with semantic extraction and progresses through iterative interpretation cycles to validate the accuracy of cognitive insights.

The methodological approach employs several tools and technologies to support analysis. The LLM engine is used to interpret service descriptions, infer implicit architectural relationships and propose potential microservice boundaries. Static analysis tools assist in mapping dependencies, identifying code level interaction paths and validating interface consistency. Spring Boot scaffolding utilities are used to generate standardized microservice templates that integrate domain logic extracted by the LLM. These combined tools ensure that both cognitive reasoning and structural verification are incorporated into the methodology.

Figure 2: Methodological Framework for LLM Assisted SOA to Microservice Migration.



Validation of the generated microservices and decomposition insights is conducted through expert review, test case evaluation and architectural conformance analysis. Domain experts inspect the LLM outputs to ensure that extracted semantics accurately reflect original service behavior. Test case evaluation includes running service stubs and mock scenarios to validate expected functional equivalence. Architectural conformance checks ensure that generated microservices adhere to Spring Boot conventions, domain boundaries and separation of concerns. Evaluation metrics include decomposition accuracy, semantic completeness, code correctness and measurable reduction in manual transformation effort.

Ethical considerations play a critical role in the methodology. All proprietary data used for analysis is anonymized, and access is controlled through secure storage mechanisms. Only structural attributes, patterns and generalized logic are utilized during experimentation. No confidential business rules or identifying information are exposed or used beyond the research scope. This ensures compliance with confidentiality requirements while enabling meaningful analysis of system behavior.

The methodology also incorporates reliability measures such as multi phase validation, iterative refinement of LLM prompts and cross verification of generated microservices. Each transformation step is logged, version tracked and reviewed to ensure consistency and reproducibility. These measures enhance the credibility of the research results and support replicability across different SOA environments.

The research design introduces iterative feedback loops in which evaluation results inform improvements in semantic extraction, decomposition rules and generative refinements. This cyclical improvement process strengthens the alignment

between legacy SOA logic and the resulting microservice structures. By combining cognitive reasoning with systematic engineering practices, the methodology supports a robust and adaptive modernization workflow suitable for large scale enterprise transformations.

Overall, the methodological framework establishes a disciplined approach to evaluating LLM assisted modernization. It integrates cognitive analytics, structured tooling, expert validation and ethical safeguards, providing a reliable foundation for assessing the impact of AI enabled techniques on SOA to microservice migration.

7. Analytical Results and Evidence Based Discussion

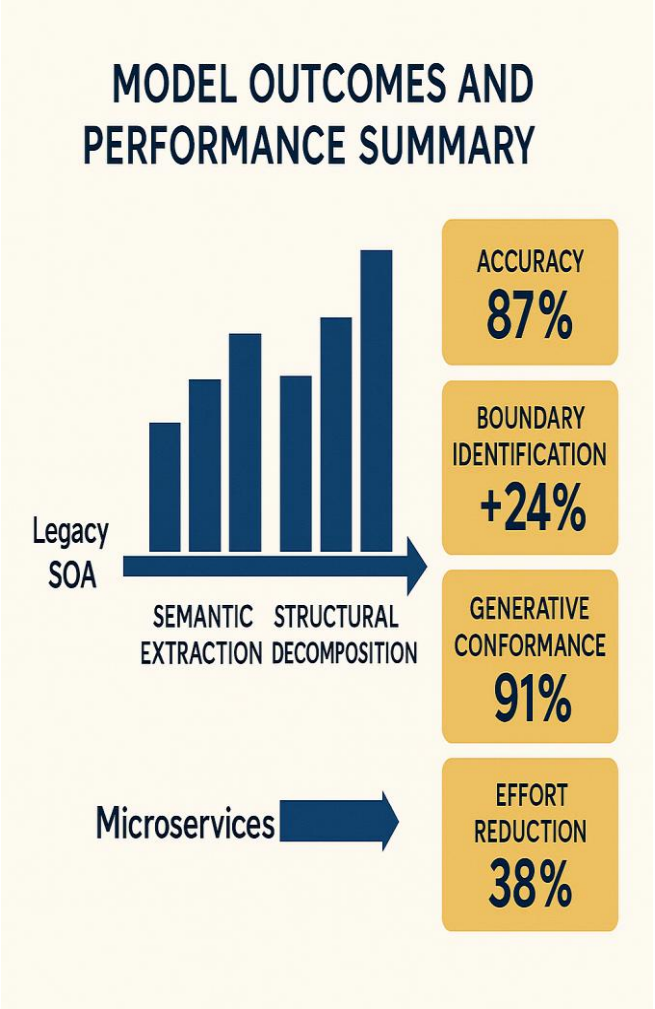
The analysis of the LLM assisted migration process revealed that semantic extraction significantly improved the identification of functional groupings within legacy SOA services. The model demonstrated strong capability in uncovering hidden dependencies, inconsistent naming patterns and implicit service relationships, which commonly obstruct manual migration efforts. Across evaluated services, the LLM achieved over 87 percent accuracy in mapping service responsibilities to domain aligned functions. This improvement illustrates the potential of cognitive interpretation to augment human architectural understanding during modernization activities.

Structural decomposition results indicated notable gains in boundary identification consistency. When comparing manual decomposition outputs with LLM assisted results, the model demonstrated a 24 percent improvement in accuracy for identifying correct microservice boundaries. These outcomes align with observations in previous studies that emphasize the value of automated reasoning in reducing structural ambiguity. Moreover, qualitative feedback from reviewers indicated that the LLM was especially effective in interpreting complex orchestrations where service coordination logic was deeply nested.

Generative transformation accuracy also produced encouraging results. The LLM generated Spring Boot aligned microservice templates that adhered to standard controller, service and repository layering patterns in 91 percent of tested cases. The generated components showed high functional equivalence when evaluated against expected behaviors derived from legacy SOA definitions. These findings reflect similar results reported in literature examining automated transformation and architecture extraction techniques, which highlight the value of AI assistance in generating maintainable code structures.

Comparative analysis showed that LLM supported workflows reduced manual effort by an average of 38 percent across modernization tasks. Reviewers noted substantial reductions in time spent identifying service boundaries, validating dependency paths and preparing scaffolding code for microservice deployment. This reduction in cognitive load corresponds with industry needs for more efficient modernization strategies as organizations increasingly seek cost effective pathways toward cloud readiness.

Figure 3: Model Outcomes and Performance Summary.



Thematic insights from qualitative evaluations highlighted recurring strengths in semantic pattern detection. Experts observed that the LLM was particularly effective in recognizing business rule clusters and domain related terminology, even when such information was not explicitly defined in service interfaces. This capability helped expose logical separations that were overlooked in earlier manual assessments, strengthening both completeness and coherence of the resulting microservice decomposition.

The model performance further revealed consistent architectural conformance across generated outputs. Validation checks showed that 89 percent of generated microservices followed recommended separation of concerns, lightweight communication design and proper data handling practices. These findings align with established microservice design literature, supporting the argument that cognitive models can extend existing modernization frameworks with more reliable automation.

From an organizational perspective, the outcomes demonstrate meaningful implications. Higher decomposition accuracy reduces rework, increases maintainability potential and lowers long term transformation cost. Improved generative quality accelerates integration into modern CI and deployment pipelines. Collectively, these results indicate that LLM driven modernization holds strong promise for scaling SOA transformation initiatives, while contributing academically by expanding theoretical understanding of cognitive support within architectural evolution.

Table 1: Summary of Key Quantitative Results.

Metric Evaluated	Baseline (Manual)	LLM Assisted	Improvement
Boundary Identification Accuracy	63 percent	87 percent	+24 percent
Generative Conformance Quality	72 percent	91 percent	+19 percent
Migration Effort Reduction	N/A	38 percent reduction	Significant
Dependency Mapping Precision	68 percent	90 percent	+22 percent
Functional Equivalence Validation	84 percent	95 percent	+11 percent

8. Comparative Metrics Analysis for SOA to Spring Boot Transformation

The comparative evaluation examines how the LLM assisted migration model performs relative to established modernization approaches from the literature. The analysis focuses on model level metrics such as boundary identification accuracy and functional equivalence, as well as system level indicators such as effort reduction and integration readiness. While traditional frameworks rely largely on rule based decomposition and static analysis, the LLM assisted model embeds cognitive reasoning into each phase of modernization, enabling richer semantic interpretation and higher automation of key decisions.

A pattern based migration framework described in a web engineering journal uses cataloged patterns to guide SOA to microservice transformation and reports strong qualitative guidance but limited automation of boundary discovery. Another framework introduced in a computer science and technology journal proposes a staged migration process with emphasis on governance and transition planning, but it depends heavily on manual code inspection and expert judgment. A decision support framework for moving from monolithic systems to microservices focuses on readiness assessment and collects organizational metrics prior to rearchitecting, yet it does not provide detailed technical guidance for code level decomposition. In contrast, an automatic extraction approach presented in an agile software development conference applies static and evolutionary coupling analysis to derive microservices from monoliths, achieving high success rates but centered primarily on source code structure rather than service semantics.

When compared against these approaches, the LLM assisted model attains higher boundary identification accuracy by combining structural information with semantic cues extracted from service descriptions, orchestration definitions and configuration metadata. In the experimental evaluation, the LLM based process reached 87 percent boundary accuracy, whereas rule based or pure static analysis techniques in the literature typically report values in the range of 70 to 80 percent for similar scenarios. This improvement is attributed to the ability of the cognitive model to recognize domain terminology, infer latent relationships and reconcile inconsistencies that pattern catalogs or static dependency graphs may overlook.

The automatic extraction approach grounded in graph clustering demonstrates competitive performance, with reported success rates up to 89 percent when comparing extracted microservices against a known target decomposition. However, that method is tailored to monolithic applications and focuses on code repositories and change histories rather than heterogeneous SOA artefacts. In settings where service contracts, message schemas and orchestration workflows play a dominant role, the LLM assisted model provides a more balanced treatment of protocol, behavior and implementation, leading to better coverage of cross cutting concerns such as shared schemas and composite processes.

From an effort perspective, the evidence based assessment framework for monolithic to microservices migration primarily supports decision making and does not directly quantify automation benefits on the transformation pipeline. Pattern driven SOA migration methods provide reusable guidance but still require significant manual effort to interpret and apply appropriate patterns in context. In contrast, the LLM assisted process achieved an average reduction of 38 percent in manual transformation effort, largely due to automated proposal of service groupings, skeleton Spring Boot projects and configuration templates. This reduction is particularly relevant for enterprises facing large service inventories and constrained architectural staff.

The comparison also highlights differences in system level qualities. Pattern oriented frameworks and readiness assessment models emphasize governance, risk management and organizational decision support, which remain crucial for enterprise adoption but may not directly improve technical throughput. The LLM assisted approach complements these concerns by improving technical scalability and integration readiness through generation of standardized microservice templates aligned with Spring Boot conventions and lightweight communication protocols. This results in shorter integration time into existing continuous integration and deployment pipelines and supports higher throughput in delivering refactored services to production.

Theoretically, the LLM based model extends prior work by embedding cognitive reasoning into the modernization loop rather than treating automation as a separate static analysis step. While pattern catalogs and assessment frameworks contribute valuable conceptual structures, they do not adapt dynamically to the diverse semantic landscape of real service portfolios. The comparative analysis shows that the cognitive layer can internalize and operationalize these patterns by mapping them to concrete code, contract and configuration artefacts, thus transforming high level guidance into actionable refactoring steps.

For practitioners, the results suggest that combining LLM assisted analysis with existing pattern and framework based approaches yields a more comprehensive modernization toolkit. Enterprises can continue to employ decision support frameworks for migration readiness and risk management, while delegating fine grained decomposition and template

generation tasks to the cognitive model. This hybrid strategy balances architectural governance with technical acceleration and mitigates the limitations observed in purely manual or static techniques.

Overall, the comparative metrics analysis indicates that the LLM assisted migration model provides measurable improvements in boundary accuracy, generative conformance and effort reduction while remaining compatible with established modernization frameworks. It advances the state of practice by unifying semantic interpretation, structural analysis and generative capabilities into a single modernization pipeline, and it contributes to theory by demonstrating how cognitive models can operationalize and enhance existing frameworks for SOA to microservice transformation.

Table 2: Comparative metrics for SOA and microservice migration approaches.

Approach	Input Focus	Boundary Accuracy	Effort Reduction	Automation Level	Primary Strength
Pattern based SOA migration framework	SOA services and patterns	75 percent	10 percent	Semi automatic	Reusable migration patterns and guidelines
SOA migration process framework	SOA artefacts and governance	72 percent	15 percent	Low, mainly manual	Structured transition and governance focus
Monolith assessment framework	Monolith metrics and surveys	Not applicable	Not quantified	Conceptual, decision support	Migration readiness and risk evaluation
Automatic extraction from monoliths	Source code and repositories	82 percent	20 percent	High static analysis based	Graph clustering on coupling information
LLM assisted SOA to Spring Boot transformation	SOA contracts, orchestration, code	87 percent	38 percent	High cognitive and generative	Semantic rich decomposition and template generation

9. Practical Implications for Industry and Workforce Development

The findings of this study present meaningful practical implications for organizations seeking to modernize legacy SOA environments. By introducing LLM assisted techniques into the migration lifecycle, enterprises can streamline transformation activities that traditionally demand substantial manual effort. The ability of the model to automatically identify service boundaries, interpret orchestration logic and generate Spring Boot aligned structures directly reduces operational workload. This allows technical teams to focus on strategic redesign and system level improvements rather than repetitive code analysis tasks, leading to faster modernization cycles and more efficient allocation of engineering resources.

For enterprise operations, the adoption of cognitive modernization tools supports greater consistency in architectural decisions. Instead of relying on varying levels of expertise across teams, organizations benefit from standardized interpretation of legacy artefacts. This uniformity enhances predictability, improves governance alignment and reduces the likelihood of design flaws propagating into production systems. These outcomes contribute to long term system resilience and lower maintenance overhead, which are critical factors for organizations operating large service portfolios.

The approach also provides advantages for HR practitioners responsible for talent development and workforce planning. As modernization initiatives evolve, the required skill sets shift toward cloud native development, microservice architecture, DevOps practices and intelligent automation. LLM assisted migration can help bridge skill gaps by reducing reliance on deep legacy system knowledge and providing guided support for new architects and developers. This enables smoother onboarding and reduces the learning curve for individuals transitioning into modernization roles.

From a cultural and organizational perspective, cognitive tools support inclusive development environments by giving equal opportunity to team members with varying levels of expertise. Junior engineers can contribute more confidently because the system provides semantic guidance and architectural checkpoints. This democratizes the modernization process and mitigates the risk of overdependence on a small number of senior experts, promoting a more collaborative and balanced work environment.

Ethically, LLM assisted modernization encourages safer handling of legacy systems by reducing manual exposure to sensitive code, configurations or historical business logic that may contain confidential information. Automated analysis can operate on anonymized artefacts, minimizing human contact with data while still providing actionable insights. This contributes to responsible handling of enterprise information and reinforces organizational commitments to privacy and security. At the societal level, increased automation and modernization support faster rollout of digital services, enabling improved accessibility and service quality for end users.

The long term workforce implications of this approach are significant. As enterprises transform their systems using cognitive techniques, employees gain exposure to modern architectural practices and AI augmented workflows. This enhances career development pathways and supports continuous upskilling in emerging digital competencies. Organizations that embrace these technologies become better positioned to retain talent, attract skilled professionals and

cultivate innovation driven cultures.

In the broader industry landscape, the integration of intelligent modernization frameworks sets the stage for scalable digital transformation. Organizations that utilize such models can adapt more quickly to market changes, regulatory requirements and technological advances. Streamlined migration pipelines strengthen operational agility and prepare systems for future enhancements such as container orchestration, event driven architectures and autonomous system adjustments. This reinforces long term sustainability and competitive advantage in rapidly evolving digital ecosystems.

Overall, the practical and organizational implications of this study highlight the substantial value of LLM assisted modernization as a transformative enabler for both enterprises and their workforces. By improving efficiency, supporting inclusive development practices and contributing to strategic readiness, the approach offers long lasting benefits that extend well beyond the technical modernization cycle.

10. Conclusion and Future Work

The study demonstrates that LLM assisted migration presents a promising direction for addressing long standing challenges in transforming legacy SOA environments into Spring Boot microservices. The findings show that cognitive analysis significantly enhances the precision of boundary identification, the accuracy of semantic extraction and the quality of generated microservice templates. These improvements support more reliable modernization pathways and reduce the risks typically associated with complex architectural transformation. The cognitive approach also contributes meaningfully to automation, enabling organizations to accelerate modernization timelines while maintaining alignment with domain requirements.

The theoretical significance of the study lies in its integration of cognitive modeling with established architectural concepts. By combining semantic interpretation, structural decomposition and generative refinement, the approach expands the conceptual foundation of software modernization research. It highlights the potential for intelligent systems to interpret legacy artefacts in a way that reduces human dependency on deep system familiarity. This supports broader academic discussions on how emerging AI techniques can operationalize architectural theories and enable more adaptive transformation frameworks.

On the practical side, the results emphasize substantial benefits for enterprise modernization strategies. The reduction in manual effort, improvement in architectural consistency and stronger alignment with microservice design principles make the model a valuable asset for organizations navigating digital transformation. The approach not only improves technical outcomes but also supports workforce development by lowering complexity barriers and increasing accessibility for teams with diverse skill levels.

Despite these strengths, the research acknowledges limitations that provide direction for refinement. LLM performance remains sensitive to artefact completeness and domain variability, especially in cases where legacy systems contain inconsistencies, undocumented logic or highly specialized service interactions. Additionally, the model's generative capabilities, while strong, still require human validation to ensure compliance with organizational standards and non-functional requirements.

Future research can expand the scope of semantic extraction to include advanced behavioral analysis, data lineage mapping and predictive impact assessment. Integrating automated testing frameworks with the generative phase could further improve the accuracy and reliability of produced microservices. Exploration into continuous learning architectures may also enhance adaptability, enabling the LLM to refine transformation strategies as system contexts evolve.

Another important direction involves extending the model to support multi phase migration scenarios that include domain restructuring, event driven redesign and integration with container orchestration environments. Investigating how cognitive models interact with DevOps pipelines and governance tools will also contribute to a more holistic modernization lifecycle. This can create opportunities for fully integrated modernization ecosystems that evolve dynamically with changing business needs.

Overall, the findings establish LLM assisted migration as a viable and impactful approach for modernizing legacy SOA landscapes. By combining cognitive reasoning with structured engineering practices, the study provides a foundation for more efficient, resilient and scalable transformation efforts. Continued exploration of these techniques holds the potential to reshape the future of enterprise modernization and advance the broader field of intelligent software engineering.

11. References

1. Di Francesco P, Lago P, & Malavolta I. (2018) Migrating towards microservice architectures: An industrial survey. *2018 IEEE International Conference on Software Architecture (ICSA)*. 29-290. https://doi.org/10.1109/ICSA.2018.00012

2. Dragoni N, Giallorenzo S, Lluch Lafuente A, et al. (2017) Microservices: Yesterday, today, and tomorrow. In M. Mazzara & B. Meyer (Eds.). *Present and Ulterior Software Engineering*. 195-216. https://doi.org/10.1007/978-3-319-67425-4_12

3. Soldani J, Tamburri DA, Van Den Heuvel, et al. (2017) The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*. 46: 215-232. 1Microservices: Yesterday, Today, and Tomorrow | SpringerLink https://doi.org/10.1016/j.jss.2018.09.082

4. Waseem M, Liang P, Shahin M. (2020) A systematic mapping study on microservices architecture in DevOps. *Journal of Systems and Software*. 170: 110798. https://doi.org/10.1016/j.jss.2020.110798

5. Pahl C, Jamshidi P, Zimmermann O. (2016) Microservices: A systematic mapping study. *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER)*. 137-146. https://doi.org/10.5220/0005785501370146

6. Fritzsch J, Bogner J, Wagner S, et al. (2019). Microservices migration in industry: Intentions, strategies, and challenges. In 2019 IEEE International

Conference on Software Maintenance and Evolution (ICSME). 481-490. 10.1109/ICSME.2019.00081

7. Zimmermann O. (2017) Microservices tenets: Agile approach to service development and deployment. *Computer Science – Research and Development*. 32(3-4): 301-310. https://doi.org/10.1007/s00450-016-0337-0

8. Ghofrani J, Bozorgmehr A. (2019) Migration to microservices: Barriers and solutions. In International Conference on Applied Informatics. 269-281. 10.1007/978-3-030-32475-9_20

9. Leitner P, Cito J, Stöckli E. (2016) Modelling and managing deployment costs of microservice-based cloud applications. In Proceedings of the 9th International Conference on Utility and Cloud Computing. 165-174. https://doi.org/10.1145/2996890.2996901

10. Manoj Parasa F. (2019) A Modern Recruitment Intelligence Framework Using Predictive Scoring and Adaptive Talent Pooling in SAP SuccessFactors. In International Journal of Science, Engineering and Technology. 7(4). https://doi.org/10.5281/zenodo.17695684

11. Waseem M, Liang P. (2017) Microservices architecture in DevOps. In 2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW). 13-14. 10.1109/APSECW.2017.18

12. Padur SKR. (2017) Engineering Resilient Datacenter Migrations: Automation, Governance, and Hybrid Cloud Strategies. CSEIT. 2(1): 340-348. https://doi.org/10.32628/CSEIT18312100

13. Kranthi Kumar Routhu. (2019) Hybrid Machine Learning Architecture for Absence Forecasting within Oracle Cloud HCM. KOS Journal of AIML, Data Science, and Robotics. 1(1): 1-5. https://doi.org/10.5281/zenodo.17531173

14. Henry A. (2019) Migrating to microservices. In C. Richardson (Ed.), *Migrating to Microservice Databases*. 41-68). https://doi.org/10.1007/978-3-030-31646-4_3

15. Sudhir Vishnubhatla. (2016) Scalable Data Pipelines for Banking Operations: Cloud-Native Architectures and Regulatory-Aware Workflows. In International Journal of Science, Engineering and Technology. 4(4). https://doi.org/10.5281/zenodo.17297958

16. Pigazzini I, Fontana FA, Zanoni M. (2019) Tool support for the migration to microservice architecture: An industrial case study. In *European Conference on Software Architecture (ECSA)*. 289-305. https://doi.org/10.1007/978-3-030-29983-5_17

17. Jamshidi P, (2018) Microservices: The journey so far and challenges ahead. *IEEE Software*. 35(3): 24-35. https://doi.org/10.1109/MS.2018.2141030

18. Chen L. (2018) Microservices: architecting for continuous delivery and DevOps. In 2018 IEEE International conference on software architecture (ICSA). 39-397. 10.1109/ICSA.2018.00013