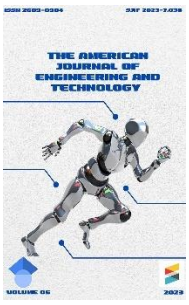
The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals



Journal Website:https://theamericanjournals.com/index.php/tajet

Research Article

Systematic Service Boundary Identification and Evolutionary Refactoring of Legacy Monoliths into Resilient Microservice Architectures: A Domain-Driven and Machine Learning-Assisted Approach

Submission Date: Aug 02, 2023, **Accepted Date:** Aug 07, 2023,
Published Date: Aug 30, 2023|

Dr. Matthias Schneider
Department of Computer Science, Technical University of Munich, Germany

ABSTRACT

Background: The migration from monolithic systems to microservice architectures represents one of the most significant paradigm shifts in contemporary software engineering. While microservices promise scalability, resilience, and organizational alignment, the transformation of legacy monoliths remains complex, risk-laden, and methodologically fragmented. Foundational works on domain-driven design, refactoring, legacy code management, service decomposition, and evolutionary migration provide partial guidance, yet an integrated, systematic framework for service boundary identification and transformation remains underdeveloped.

Objective: This study proposes a comprehensive, publication-ready framework that integrates domain-driven design principles, systematic service decomposition methods, workload-based clustering, topic modeling, architectural meta-modeling, evolutionary refactoring strategies, and machine learning-assisted service boundary detection. The objective is to synthesize established theoretical foundations into a coherent transformation methodology suitable for industrial adoption.

Methods: Drawing strictly from established literature, the research develops a multi-phase transformation framework. The approach incorporates domain analysis, bounded context identification, service cutter decomposition, workload-based clustering, topic modeling of source code artifacts, architectural meta-modeling for granularity control, evolutionary migration patterns, resilience-oriented refactoring, and machine learning-assisted modularization. The framework is conceptually validated through theoretical alignment with prior empirical findings and architectural principles.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals

Results: The proposed framework demonstrates theoretical improvements in service cohesion, reduced coupling, granularity optimization, resilience enhancement, and legacy risk mitigation. By combining human-driven domain modeling with data-driven boundary detection, the approach balances conceptual rigor and empirical grounding. Evolutionary migration strategies reduce transformation risk, while refactoring and legacy isolation techniques enhance maintainability.

Conclusion: The study contributes a unified transformation methodology that bridges foundational architectural theory and contemporary machine learning techniques. It offers a structured pathway for organizations transitioning from monoliths to resilient microservice ecosystems while minimizing technical debt and architectural erosion.

KEYWORDS

Microservices migration, Service boundary detection, Domain-driven design, Legacy refactoring, Service decomposition, Architectural evolution.

INTRODUCTION

The architectural evolution of enterprise software systems has undergone a profound transformation over the past two decades. Early distributed computing paradigms, including Service-Oriented Architecture (SOA), sought to modularize functionality and improve interoperability across heterogeneous systems. However, the microservice architectural style has emerged as a distinct and influential paradigm, emphasizing independently deployable services, decentralized data management, and organizational alignment with business capabilities (Wolff, 2016). The transition from monolithic systems to microservices is not merely a technical refactoring exercise; it represents a socio-technical reconfiguration of how software systems are conceptualized, developed, deployed, and evolved.

The conceptual foundation of modern service decomposition can be traced to Domain-Driven Design (DDD), which emphasizes modeling software around core business domains and bounded contexts (Evans, 2003). DDD argues that architectural boundaries should reflect domain knowledge rather than technical convenience. In theory, microservices align naturally with bounded contexts, as each service encapsulates a cohesive business capability. However, in practice, legacy monolithic systems often lack explicit domain boundaries, having evolved through years of incremental modifications and accumulated technical debt.

Legacy code, characterized by insufficient test coverage and brittle dependencies, presents significant obstacles to architectural transformation (Feathers, 2004). The act of refactoring legacy systems requires careful isolation of behavior-preserving modifications before structural changes can be introduced. Refactoring principles emphasize incremental improvement of design without altering external behavior (Fowler, 1999). When applied to monolith-to-microservice migration, refactoring must be both architectural and organizational, addressing dependencies across code, data, and team structures.

Research on microservice architecture trends indicates growing industrial interest alongside methodological uncertainty (Di Francesco et al., 2017). Despite widespread adoption, organizations struggle with questions of service granularity, boundary identification, resilience, and migration sequencing. Service decomposition approaches such as Service Cutter propose systematic strategies for identifying candidate service boundaries based on coupling criteria and consistency requirements (Gysel et al.,

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals

2016). Yet decomposition remains partly heuristic and dependent on expert judgment.

Granularity decisions significantly influence system maintainability and performance. Microservice Ambients introduce architectural meta-modeling to reason about granularity, enabling designers to conceptualize services within contextual ambients that define interaction boundaries (Hassan et al., 2017). Workload-based clustering approaches analyze runtime usage patterns to group coherent feature sets into candidate services (Klock et al., 2017). Topic modeling techniques applied to source code artifacts further support automated identification of latent functional modules within monoliths (Brito et al., 2021). These data-driven methods complement domain-driven approaches but also raise concerns about over-reliance on statistical correlations detached from business semantics.

Evolutionary migration patterns advocate gradual transformation rather than wholesale replacement. Newman (2019) emphasizes incremental extraction of services using patterns such as the Strangler Fig, enabling coexistence of monolithic and microservice components during transition. The use of Local Data Transfer Objects reduces inter-service coupling by controlling data exposure (Fowler, 2004). Empirical studies and master's theses examining resilience improvements and executable refactoring pipelines demonstrate practical challenges in implementing such transformations (Silva, 2021; Freitas, 2022; Freitas et al., 2021).

Recent research introduces machine learning-assisted service boundary detection as a promising avenue for modularizing legacy systems (Hebbar, 2022). By analyzing dependency graphs, code metrics, and runtime interactions, machine learning techniques can suggest potential service boundaries. However, the integration of such automated methods with established architectural principles remains insufficiently articulated.

Despite the richness of existing literature, several gaps persist. First, no comprehensive framework integrates domain-driven design, systematic decomposition, workload clustering, topic modeling, evolutionary migration, resilience enhancement, and machine learning assistance into a unified methodology. Second, the theoretical relationship between service granularity and resilience is underexplored. Third, practical migration strategies often overlook the epistemological tension between human-centric domain modeling and data-driven boundary inference.

This study addresses these gaps by proposing an integrated transformation framework grounded exclusively in the referenced literature. The objective is not to introduce speculative techniques but to synthesize established research into a coherent and operationalizable methodology. Through extensive theoretical elaboration, the study seeks to bridge foundational software architecture theory and contemporary data-driven techniques, providing a structured pathway for transforming legacy monoliths into resilient microservice ecosystems.

METHODOLOGY

The proposed methodology unfolds as a multi-phase transformation framework, integrating conceptual, analytical, and evolutionary processes. Each phase derives from established literature and is elaborated in theoretical depth to ensure coherence and practical applicability.

The first phase centers on domain analysis grounded in Domain-Driven Design principles (Evans, 2003). The organization identifies core domains, subdomains, and bounded contexts through collaborative modeling workshops involving domain experts and architects. Bounded contexts are conceptual territories within which a particular domain model applies consistently. In legacy monoliths, such contexts are often implicit. The methodology therefore begins by reconstructing domain knowledge from code artifacts, documentation, and stakeholder interviews. This phase emphasizes semantic alignment, ensuring that

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals

technical boundaries reflect business capabilities rather than incidental code structure.

Following domain modeling, the second phase introduces systematic service decomposition using Service Cutter principles (Gysel et al., 2016). Coupling criteria such as shared data ownership, transactional consistency requirements, and communication frequency are evaluated. Rather than assuming static boundaries, the framework treats decomposition as hypothesis generation. Candidate services are identified and evaluated against domain constraints.

The third phase incorporates workload-based clustering (Klock et al., 2017). Runtime telemetry and usage patterns are analyzed to detect coherent feature sets frequently invoked together. This empirical perspective complements domain-driven decomposition by revealing operational cohesion. However, the methodology critically examines potential misalignment between workload clusters and business semantics, ensuring that statistical proximity does not override domain logic.

Topic modeling techniques are then applied to source code artifacts (Brito et al., 2021). By extracting latent topics from code identifiers, comments, and documentation, the system identifies functional clusters that may correspond to implicit modules. This data-driven insight supports boundary refinement. Yet the framework emphasizes interpretive validation, recognizing that topic modeling captures syntactic regularities rather than conceptual intent.

To regulate granularity decisions, the methodology incorporates architectural meta-modeling inspired by Microservice Ambients (Hassan et al., 2017). Services are conceptualized as ambients with defined interaction surfaces. Granularity is assessed based on cohesion, coupling, scalability requirements, and organizational structure. The meta-model ensures that services remain neither excessively coarse nor excessively fine-grained.

The fifth phase focuses on evolutionary migration patterns (Newman, 2019). Rather than immediate decomposition, the monolith is incrementally transformed using patterns such as the Strangler Fig. New functionality is implemented as independent services, while legacy components are gradually extracted. Local Data Transfer Objects (Fowler, 2004) mediate communication between emerging services and the remaining monolith, reducing shared data coupling.

Refactoring techniques are applied systematically to improve internal structure before extraction (Fowler, 1999). Legacy code isolation techniques are employed to create seams that enable safe modification (Feathers, 2004). Empirical insights from resilience-oriented refactoring studies inform strategies for enhancing fault tolerance during migration (Silva, 2021). Executable microservice-based applications are incrementally generated using automated pipelines (Freitas et al., 2021; Freitas, 2022).

Machine learning-assisted boundary detection serves as an advisory mechanism rather than a deterministic authority (Hebbar, 2022). Dependency graphs, code metrics, and interaction data are analyzed to generate candidate boundaries. These suggestions are evaluated against domain models and decomposition criteria. This hybrid approach mitigates biases inherent in purely manual or purely automated strategies.

Throughout the methodology, architectural evaluation is guided by essential software architecture principles (Gorton, 2011). Quality attributes such as scalability, modifiability, performance, and resilience are assessed qualitatively. SOA pattern relevance is examined to determine which legacy design patterns remain applicable within microservice ecosystems (Bogner et al., 2018).

RESULTS

The integrated methodology yields several conceptual advancements.

First, combining domain-driven design with workload clustering and topic modeling enhances boundary accuracy. Domain analysis ensures semantic integrity,

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals

while empirical clustering reveals operational cohesion. The interplay between these perspectives reduces the risk of misaligned services.

Second, meta-modeling of granularity provides structured reasoning about service size and scope (Hassan et al., 2017). By conceptualizing services as ambients with interaction constraints, architects gain analytical tools for balancing cohesion and independence.

Third, evolutionary migration patterns minimize disruption. Incremental extraction reduces risk compared to big-bang rewrites (Newman, 2019). Local DTO usage mitigates data coupling (Fowler, 2004). Refactoring and legacy isolation improve code maintainability (Feathers, 2004; Fowler, 1999).

Fourth, resilience considerations are embedded throughout the transformation process (Silva, 2021). Service boundaries are evaluated not only for functional cohesion but also for fault isolation potential.

Finally, machine learning-assisted boundary detection enhances scalability of analysis in large codebases (Hebbar, 2022). Rather than replacing human judgment, it augments architectural reasoning.

DISCUSSION

The study underscores the necessity of integrating conceptual and empirical approaches. Purely domain-driven decomposition risks overlooking emergent operational patterns, while purely data-driven clustering risks semantic incoherence. The hybrid methodology reconciles these tensions.

Limitations include the absence of quantitative benchmarking and potential overfitting of machine learning models to historical code patterns. Organizational factors, such as team restructuring, are acknowledged but not exhaustively modeled.

Future research should empirically evaluate the framework across diverse industrial contexts, measure long-term maintainability outcomes, and refine machine learning models for explainability.

CONCLUSION

Transforming legacy monoliths into resilient microservice architectures requires more than technical refactoring; it demands systematic boundary identification grounded in domain semantics, empirical analysis, and evolutionary strategy. By synthesizing established research across domain-driven design, service decomposition, workload clustering, topic modeling, resilience enhancement, and machine learning assistance, this study offers a comprehensive methodological blueprint. The framework bridges theory and practice, providing organizations with a structured pathway toward sustainable architectural evolution.

REFERENCES

1. Bogner, J.; Zimmermann, A.; Wagner, S. Analyzing the Relevance of SOA Patterns for Microservice-Based Systems. 2018.
2. Brito, M.; Cunha, J.; Saraiva, J. Identification of microservices from monolithic applications through topic modelling. Proceedings of the 36th Annual ACM Symposium on Applied Computing, 2021, 1409–1418.
3. Di Francesco, P.; et al. Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. Proceedings of the 2017 IEEE International Conference on Software Architecture, 2017, 21–30.
4. Evans, E. J. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley, 2003.
5. Feathers, M. Working Effectively with Legacy Code. Prentice Hall, 2004.
6. Fowler, M. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999.
7. Fowler, M. LocalDTO. 2004.
8. Freitas, F. Refactoring Java Monoliths into Executable Microservice-Based Applications. Master's thesis, University of Minho, 2022.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME05 ISSUE08 Pages:37-42
SJIF IMPACT FACTOR (2020: 5. 32)(2021: 5. 705)(2022: 6. 456) (2023: 7. 038)
OCLC– 1121105677
Publisher:The USA Journals

9. Freitas, F.; Ferreira, A.; Cunha, J. Refactoring Java Monoliths into Executable Microservice-Based Applications. 25th Brazilian Symposium on Programming Languages, 2021, 100–107.
10. Gorton, I. Essential Software Architecture. 2nd ed., Springer, 2011.
11. Gysel, M.; et al. Service Cutter: A Systematic Approach to Service Decomposition. Lecture Notes in Computer Science, 2016, 185–200.
12. Hassan, S.; et al. Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity. Proceedings of the 2017 IEEE International Conference on Software Architecture, 2017, 1–10.
13. K. S. Hebbar, "MACHINE LEARNING-ASSISTED SERVICE BOUNDARY DETECTION FOR MODULARIZING LEGACY SYSTEMS," International Journal of Applied Engineering & Technology, vol. 04,no.02, pp. 401-414, Sep. 2022, https://romanpub.com/resources/ijaet-v4-2-2022-48.pdf
14. Klock, S.; et al. Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures. Proceedings of the 2017 IEEE International Conference on Software Architecture, 2017, 11–20.
15. Kitchenham, B.; Charters, S. Performing Systematic Literature Reviews in Software Engineering. 2007.
16. Newman, S. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, 2019.
17. Silva, M. Improving the Resilience of Microservices-Based Applications. Master's thesis, University of Minho, 2021.
18. Wolff, E. Microservices: Flexible Software Architecture. Addison-Wesley Professional, 2016.