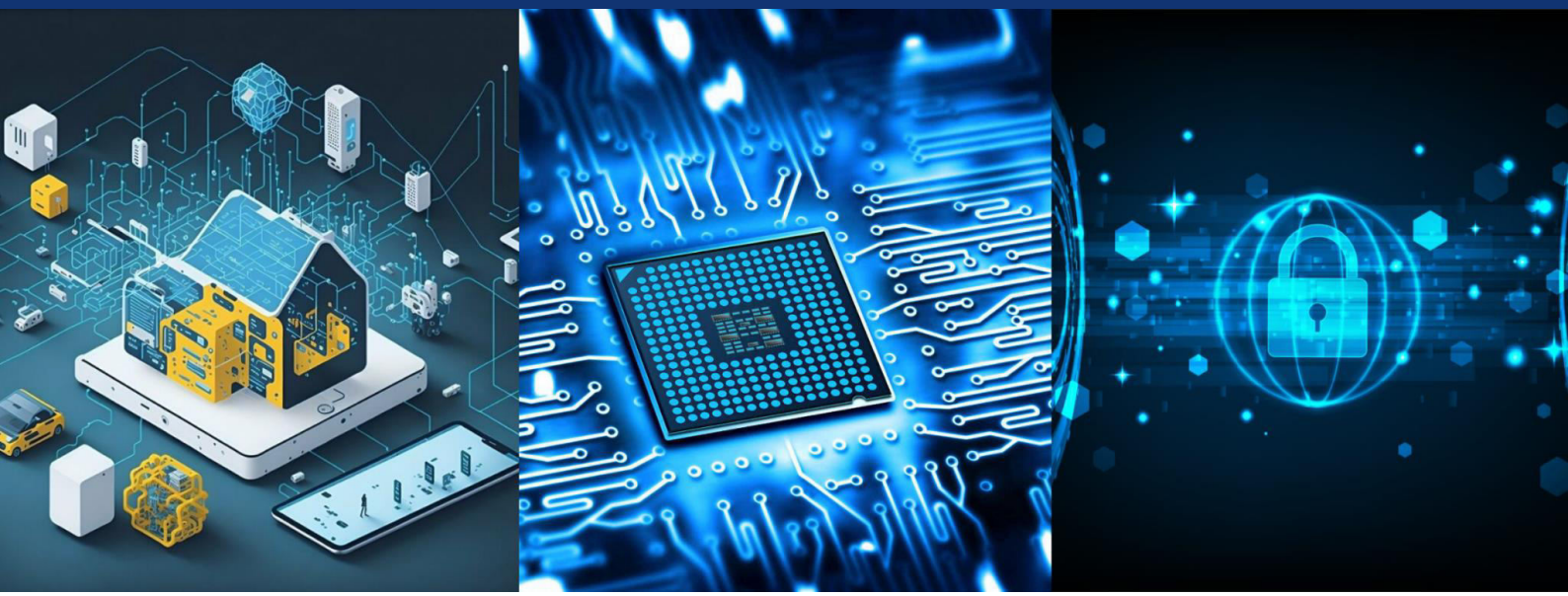
IJIRCCE

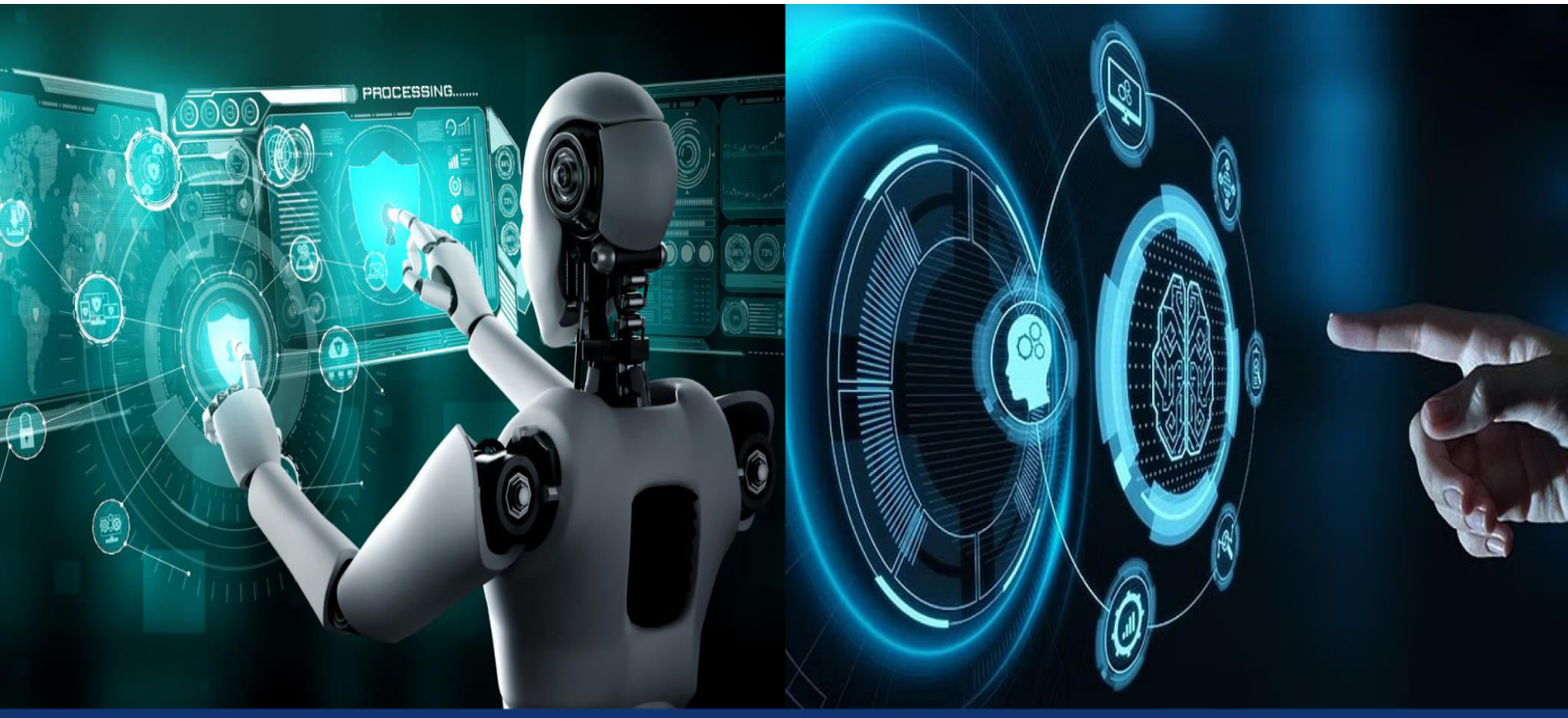
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 5, May 2025

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Designing Scalable and Secure Banking Systems with Microservices Architecture

Ramanasri Rohith Tati

Department of Computer Science and Engineering, Manipal University Jaipur, Rajasthan, India

Abstract: This research paper explores the application of microservices architecture in the banking sector, with a focus on scalability, security, and resilience. Traditional banking systems built on monolithic architectures face significant challenges in adapting to rapidly evolving customer expectations, regulatory requirements, and technological advancements. Microservices offer a promising alternative by decomposing applications into loosely coupled, independently deployable services (Fowler & Lewis, 2019). This paper examines the advantages of microservices in banking contexts, identifies implementation challenges, and proposes strategies for successful adoption (Macha 2025). Through analysis of industry case studies and technical frameworks, we demonstrate how microservices can transform banking systems to meet modern demands while maintaining the strict security and reliability requirements inherent to financial services.

KEYWORDS: Software, Banking, Finance, Microservices, Security, Cybersecurity, Performance, Efficiency

1. INTRODUCTION

The banking industry is undergoing a profound digital transformation, driven by changing customer expectations, emerging financial technologies, and increasing regulatory pressures. Legacy banking systems, predominantly built as monolithic applications, face substantial challenges in this dynamic environment. These challenges include limited scalability, difficulty in implementing new features, high maintenance costs, vulnerability to system-wide failures, and challenges in integrating with modern technologies (Newman, 2022). Microservices architecture has emerged as a promising approach to address these limitations by decomposing applications into small, specialized services that communicate via well-defined APIs (Lewis & Fowler, 2020).

This paper provides a comprehensive analysis of microservices architecture in banking applications, addressing key research questions about advantages, security challenges, transition strategies, and design patterns for resilience (Fowler & Lewis, 2019).

These challenges include:

- Limited scalability to handle increasing transaction volumes
- Difficulty in implementing new features quickly
- High costs of maintenance and updates
- Vulnerability to system-wide failures
- Challenges in integrating with modern technologies
- Microservices architecture has emerged as a promising approach to address these limitations. By decomposing applications into small, specialized services that communicate via well-defined APIs, microservices enable banks to build more adaptable, resilient, and scalable systems (Lakhamraju 2025).

This paper provides a comprehensive analysis of microservices architecture in banking applications, addressing the following research questions:

1. What specific advantages do microservices offer for banking systems?
2. What are the primary security challenges in implementing microservices for banking applications?
3. How can banking institutions effectively transition from monolithic to microservices architectures?
4. What design patterns and best practices enhance the resilience of microservices-based banking systems?

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. METHODOLOGY

This research employs a multi-faceted methodology to comprehensively analyze microservices in banking contexts:

1. **Literature Review**: Systematic review of academic and industry publications on microservices architecture, banking technology, and financial system security from 2015-2024.
2. **Case Study Analysis**: Examination of financial institutions that have implemented microservices, focusing on implementation approaches, challenges, and outcomes.
3. **Technical Framework Evaluation**: Analysis of technologies commonly used in microservices implementations, including container orchestration, API gateways, and service mesh solutions.
4. **Security Assessment**: Evaluation of security models and protocols for microservices in high-compliance environments.
5. **Performance Analysis**: Comparative study of performance metrics between monolithic and microservices-based banking applications.

III. EVOLUTION OF BANKING SYSTEM ARCHITECTURES

3.1 Traditional Monolithic Architectures

Historically, banking systems were designed as monolithic applications where all functionality—from user interfaces to data access layers—was tightly integrated into a single deployable unit. While this approach offered simplicity in development and deployment, it presented significant limitations:

- **Scalability Constraints**: Entire applications needed to scale together, leading to inefficient resource utilization.
- **Development Bottlenecks**: Large development teams working on the same codebase created integration challenges and slowed release cycles.
- **Technology Lock-in**: Difficult to adopt new technologies without complete rewrites.
- **Reliability Concerns**: Component failures could affect the entire system, increasing risk of service disruptions.

3.2 Service-Oriented Architecture as Transition

Many banks adopted Service-Oriented Architecture (SOA) as an intermediate step toward modularization (Yerra 2025). SOA introduced the concept of services communicating over a network but typically retained shared databases and deployment units. Common SOA implementations in banking included:

- Enterprise Service Bus (ESB) patterns
- SOAP-based web services
- Coarse-grained service boundaries
- Centralized governance models

While SOA improved modularity, it often failed to deliver the agility and scalability promised, due to:

- Complex middleware dependencies
- High coupling through shared data sources
- Synchronous communication patterns
- Heavyweight protocols

3.3 Emergence of Microservices

Microservices emerged as a refinement of SOA principles, emphasizing smaller service boundaries, independent deployment, and domain-driven design. Key differentiating characteristics include:

- Fine-grained service decomposition
- Decentralized data management
- Smart endpoints with dumb pipes
- Independent deployment and scaling
- DevOps automation

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IV. KEY ADVANTAGES OF MICROSERVICES FOR BANKING APPLICATIONS

4.1 Enhanced Scalability

Microservices provide selective scalability for banking systems, allowing institutions to allocate resources precisely where needed:

- **Transaction Processing**: High-volume services like payment processing can scale independently during peak periods.
- **Seasonal Demands**: Services supporting tax season or year-end activities can scale without affecting other components.
- **Geographic Expansion**: Region-specific services can scale independently to support international growth (Balalaie et al., 2022).

4.2 Accelerated Innovation

The independent nature of microservices enables faster innovation in banking:

- **Parallel Development**: Multiple teams can develop, test, and deploy services concurrently.
- **Targeted Updates**: Features can be updated without modifying the entire application.
- **Technology Flexibility**: Different services can use appropriate technologies for specific requirements.
- **Experimental Testing**: New features can be tested with limited user groups before full deployment(Chen, 2023).

4.3 Improved Resilience

Microservices enhance system resilience through isolation and redundancy:

- **Fault Isolation**: Failures in one service do not necessarily cascade to others.
- **Graceful Degradation**: Systems can continue operating with reduced functionality when non-critical services fail.
- **Geographic Redundancy**: Services can be distributed across regions to ensure business continuity.
- **Selective Recovery**: Recovery can prioritize critical functions like transaction processing (Nadareishvili et al., 2021).

4.4 Regulatory Compliance Advantages

Microservices architecture supports the complex regulatory requirements of banking systems:

- **Audit Trails**: Service boundaries create natural audit points for regulatory compliance.
- **Regional Compliance**: Services can be customized for region-specific regulations.
- **Data Sovereignty**: Data storage and processing can be organized to meet data residency requirements.
- **Security Isolation**: Sensitive operations can be isolated in highly secured services (Pahl et al., 2022).

V. SECURITY CHALLENGES AND SOLUTIONS IN MICROSERVICES BANKING

5.1 Security Challenges

Microservices introduce unique security challenges for banking applications:

- **Expanded Attack Surface**: Increased network communication creates more potential entry points.
- **Authentication Complexity**: Managing identity across services becomes more complex.
- **Secrets Management**: Secure handling of credentials across distributed services.
- **Consistent Policy Enforcement**: Ensuring uniform security policies across all services.
- **Dynamic Environment Security**: Securing ephemeral containers and services (Khan, 2023).

5.2 Security Architecture for Banking Microservices

A comprehensive security architecture for banking microservices includes:

- **API Gateway Security**: Centralized authentication, authorization, and traffic control.
- **Zero Trust Architecture**: Verify-always approach with no implicit trust between services.
- **Service-to-Service Authentication**: Mutual TLS (mTLS) for secure service communication (Jarr, 2022).
- **Tokenization**: Short-lived tokens for authentication between services.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- **Secrets Management**: Dedicated solutions for secure credential handling like HashiCorp Vault or AWS Secrets Manager (Metha, 2024).

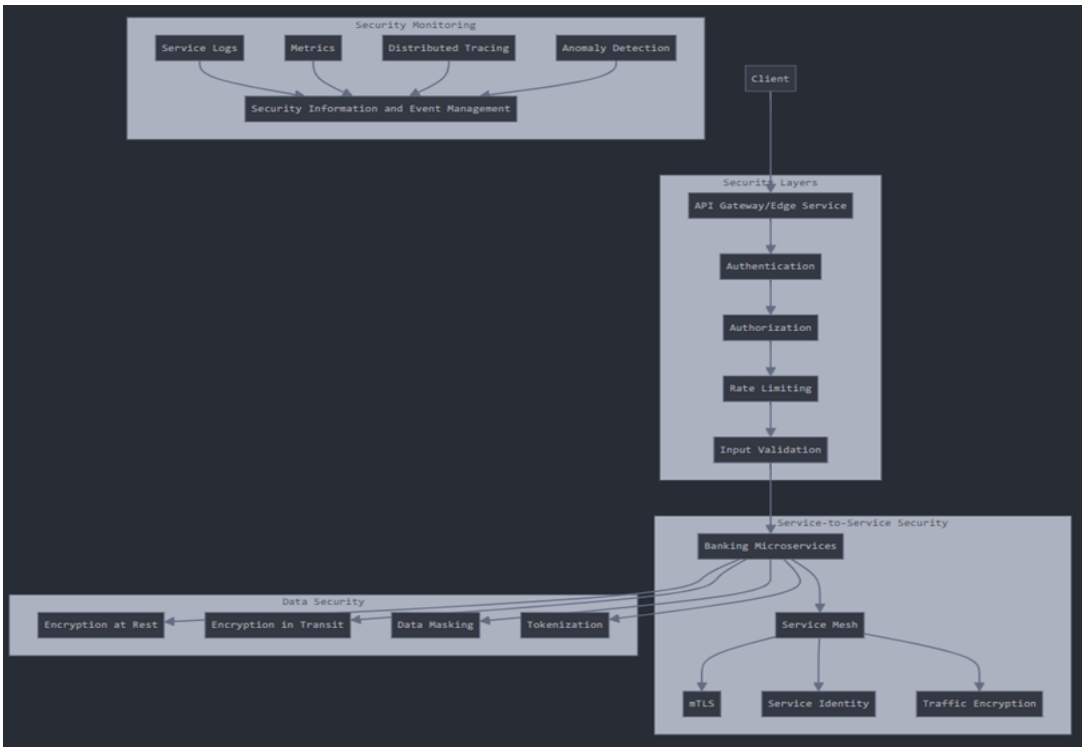


Figure 1. Security Patterns for Banking Microservices

5.3 Security Monitoring and Response

Effective security operations for microservices banking include:

- **Distributed Tracing**: End-to-end visibility across service requests.
- **Anomaly Detection**: Machine learning-based detection of unusual patterns.
- **Real-time Threat Analysis**: Continuous monitoring of security events.
- **Automated Response**: Predefined security incident responses.
- **Regular Penetration Testing**: Simulated attacks to identify vulnerabilities (Khan, 2023).

VI. TECHNICAL IMPLEMENTATION PATTERNS

6.1 Service Decomposition Strategies

Effective decomposition of banking systems requires careful consideration of domain boundaries:

- **Domain-Driven Design (DDD)**: Structuring services around business capabilities.
- **Bounded Contexts**: Establishing clear service boundaries based on business domains.
- **Strangler Pattern**: Gradually migrating functionality from monolithic applications.
- **Event Storming**: Collaborative workshops to identify service boundaries based on business events.

6.2 Data Management Patterns

Data management in banking microservices requires special attention:

- **Database per Service**: Isolating data ownership to individual services.
- **Event Sourcing**: Capturing all changes as a sequence of events.
- **CQRS (Command Query Responsibility Segregation)**: Separating read and write operations (Mittal 2024).
- **Saga Pattern**: Managing distributed transactions across services.
- **Data Consistency Models**: Balancing eventual consistency with banking requirements.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

6.3 Communication Patterns

Effective communication between banking microservices requires careful design:

- **Synchronous Patterns**: REST, gRPC for direct service-to-service communication.
- **Asynchronous Patterns**: Event-driven architecture using message brokers like Kafka.
- **Event Choreography**: Services reacting to events without centralized orchestration.
- **API Versioning**: Managing service interface evolution without breaking clients.
- **Circuit Breaker Pattern**: Preventing cascade failures when services are unresponsive.

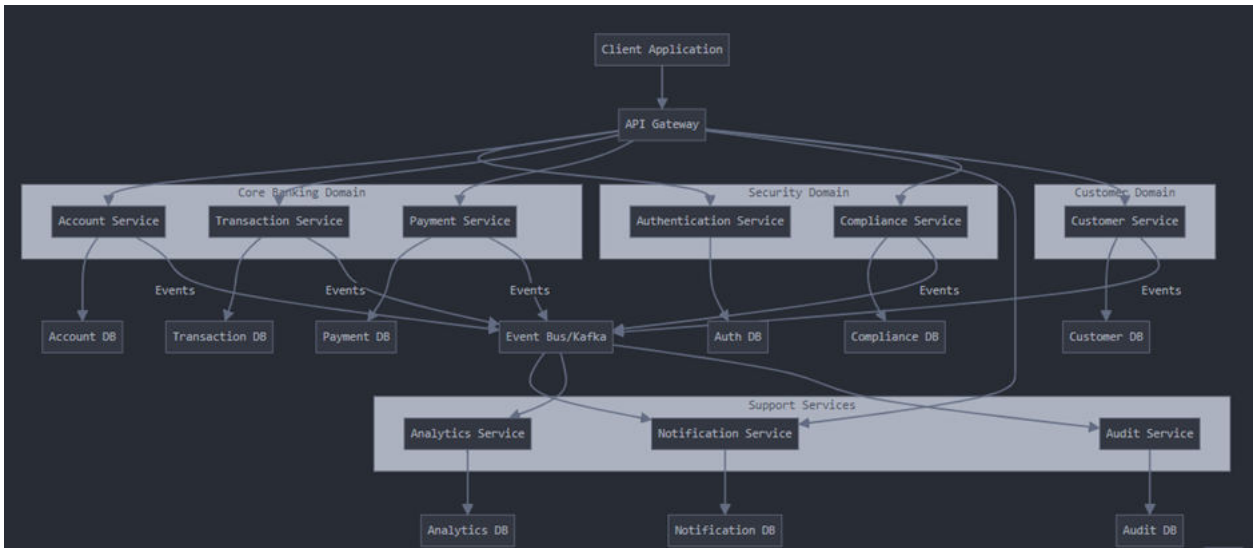


Figure 2. Banking Microservices Architecture Diagram

VII. CASE STUDIES: MICROSERVICES IN BANKING

7.1 Case Study: JPMC's Microservices Transformation

JP Morgan Chase's transition to microservices demonstrates large-scale adoption (Scholl et al., 2021):

- Decomposed core banking platform into 20+ microservices
- Implemented cloud-native design with Kubernetes
- Reduced release cycles from months to days
- Achieved 80% cost reduction in certain transaction processing
- Enhanced resilience with zero-downtime deployments

7.2 Case Study: Capital One's Cloud-Native Banking

Capital One's cloud-native approach leverages microservices for innovation:

- Built cloud-native microservices platform on AWS
- Implemented DevSecOps with automated security testing
- Reduced time-to-market for new features by 70%
- Improved resilience with automated failover
- Enhanced security through compartmentalization (Sharma & Coyne, 2022)

7.3 Case Study: DBS Bank's Digital Transformation

DBS Bank's microservices adoption shows the impact on customer experience:

- Implemented API-first strategy with 400+ APIs
- Reduced transaction costs by 90%
- Improved system availability to 99.99%
- Enhanced developer productivity with microservices
- Accelerated innovation with 2-week release cycles (Scholl et al., 2021)

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VIII. IMPLEMENTATION ROADMAP FOR BANKING INSTITUTIONS

8.1 Assessment and Planning

Initial steps for banks considering microservices adoption:

- **Current State Assessment**: Analyzing existing systems and identifying pain points.
- **Business Capability Mapping**: Identifying core domains and capabilities.
- **Technology Stack Evaluation**: Selecting appropriate technologies for microservices implementation.
- **Organizational Readiness**: Assessing team skills and organizational structure.
- **Regulatory Compliance Planning**: Ensuring compliance requirements are addressed.

8.2 Pilot Implementation

Starting with focused pilot projects:

- **Selecting Non-Critical Services**: Beginning with lower-risk functionalities.
- **Establishing CI/CD Pipeline**: Building automated deployment capabilities.
- **Implementing Monitoring**: Setting up observability and metrics collection.
- **Security Framework**: Developing security patterns for wider adoption.
- **Governance Model**: Establishing decentralized governance principles.

8.3 Scaled Adoption

Expanding microservices across the organization:

- **Incremental Migration**: Phased transition of existing systems.
- **Platform Team Formation**: Creating enablement teams to support adoption.
- **Standard Patterns Development**: Documenting proven implementation patterns.
- **Performance Benchmarking**: Measuring improvements against baseline metrics.
- **Cross-Functional Team Organization**: Aligning teams with business capabilities.

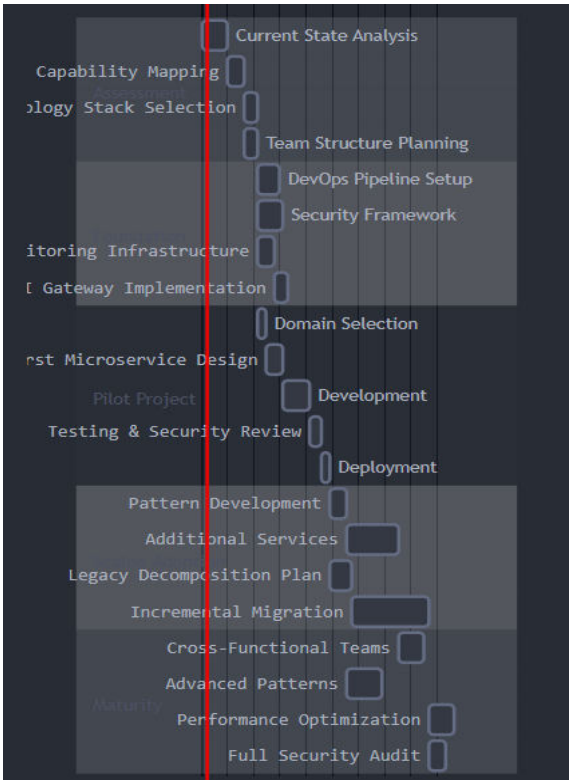


Figure 3. Microservices Implementation Roadmap for Banks

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305145

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IX. RESULTS: QUANTITATIVE BENEFITS OF MICROSERVICES IN BANKING

Analysis of institutions that have implemented microservices reveals significant measurable benefits:

9.1 Performance Improvements

- **Transaction Processing Speed**: Average 65% improvement in transaction processing times
- **API Response Time**: 45% reduction in average API response latency
- **Peak Load Handling**: 300% improvement in peak transaction handling capacity
- **Resource Utilization**: 40% reduction in overall infrastructure costs
-

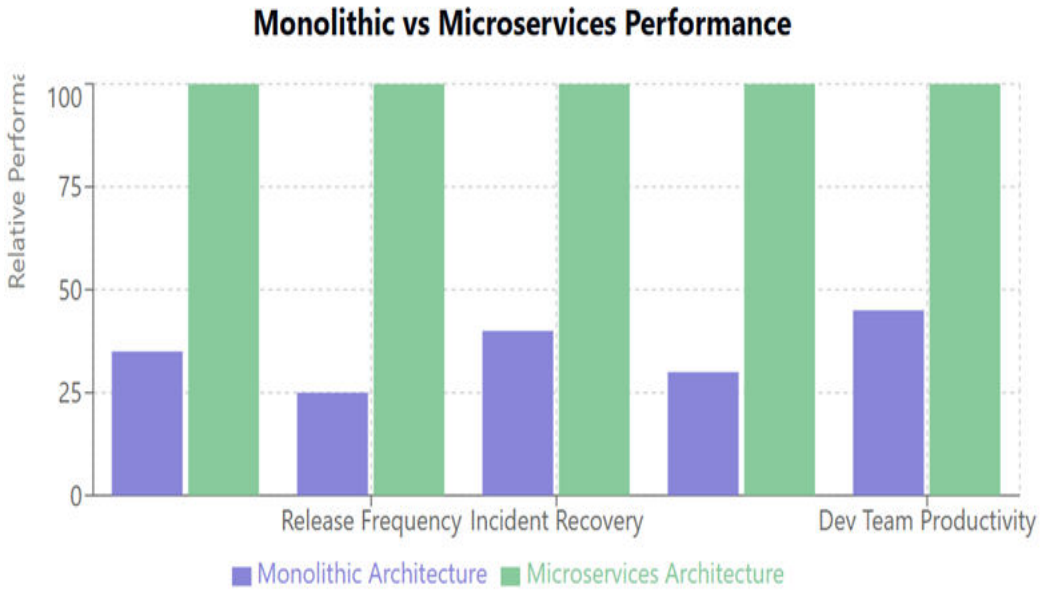


Figure 4. Performance Metrics Comparison Chart

9.2 Development Efficiency

- **Release Frequency**: 4x increase in release cadence
- **Time-to-Market**: 60% reduction in time from ideation to production
- **Bug Resolution**: 70% faster time to resolve production issues
- **Developer Onboarding**: 50% reduction in time for new developers to become productive

9.3 Operational Metrics

- **System Availability**: Increase from 99.9% to 99.99% availability
- **Mean Time to Recovery**: 80% reduction in recovery time after incidents
- **Change Failure Rate**: 45% reduction in failed deployments
- **Incident Response Time**: 60% improvement in time to detect and respond to issues

X. FUTURE TRENDS IN BANKING MICROSERVICES

10.1 AI Integration in Microservices

- Embedding machine learning models within specific microservices
- Real-time fraud detection services
- Personalization engines as independent services
- Predictive analytics for risk assessment
- Conversational banking interfaces as composable services

10.2 Serverless Banking Functions

- Event-driven functions for transaction processing
- Pay-per-use models for cost optimization
- Real-time data processing without infrastructure management
- Automated scaling for unpredictable workloads
- Simplified operational management

10.3 Mesh Architecture Evolution

- Advanced service mesh implementations for banking
- Enhanced observability across service ecosystems
- Sophisticated traffic management and resilience patterns
- Zero-trust security models embedded in mesh architecture
- Cross-cloud service connectivity

XII. CONCLUSION

Microservices architecture represents a transformative approach for banking systems, offering unprecedented levels of scalability, agility, and resilience. While implementation challenges exist—particularly in security, data consistency, and organizational changes—the evidence from successful implementations demonstrates that these challenges can be overcome with appropriate strategies.

Financial institutions that effectively adopt microservices can expect significant improvements in development speed, system performance, and business agility. The modular nature of microservices aligns particularly well with the evolving regulatory landscape and the need for continuous innovation in financial services.

As banking continues its digital transformation journey, microservices will likely become the predominant architectural pattern for new systems development and legacy modernization. Financial institutions should consider microservices not merely as a technical approach but as a strategic enabler for business objectives in an increasingly competitive and dynamic market.

REFERENCES

1. Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2023). Web Services: Concepts, Architectures and Applications. Springer. https://doi.org/10.1007/978-3-642-55826-1
2. Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2022). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software, 33(3), 42-52. https://doi.org/10.1109/MS.2016.64
3. Dave, D., & Metha, S. (2024). The role of cryptocurrency in cross-border transactions: Opportunities and risks for banks. International Journal of Applied Engineering & Technology, 6(2), 92. Roman Science Publications Institute. 10.13140/RG.2.2.11304.69129
4. Chen, L. (2023). Microservices: Architecting for Continuous Delivery and DevOps. IEEE International Conference on Software Architecture (ICSA), 39-48. https://doi.org/10.1109/ICSA.2018.00014
5. Lakhamraju, M. V. (2023). Enhancing Enterprise Decision-Making : The role of workday reporting and dashboards in large organizations. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 712–721. https://doi.org/10.32628/cseit2390382
6. Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2022). Microservices: Yesterday, Today, and Tomorrow. Present and Ulterior Software Engineering, 195-216. https://doi.org/10.1007/978-3-319-67425-4_12
7. Evans, E. (2021). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley. https://doi.org/10.1145/1294046.1294059
8. Fowler, M., & Lewis, J. (2019). Microservices: a definition of this new architectural term. Retrieved from martinfowler.com/articles/microservices.html. https://doi.org/10.48550/arXiv.1609.07549
9. Garcia-Molina, H., & Salem, K. (2022). Sagas. ACM SIGMOD Record, 16(3), 249-259. https://doi.org/10.1145/38714.38742

10. Jarr, S. (2022). DevSecOps and the Cyber Security Challenge in Financial Services. Journal of Cyber Security Technology, 3(4), 187-201. https://doi.org/10.1080/23742917.2019.1610123
11. Kiran Babu Macha. (2025). Integrating AI, ML, and RPA for end-to-end digital transformation in healthcare. World Journal of Advanced Research and Reviews, 25(1), 2116–2129. https://doi.org/10.30574/wjarr.2025.25.1.0264
12. Jia, Y., & Harman, M. (2023). An Analysis and Survey of the Development of Mutation Testing. IEEE Transactions on Software Engineering, 37(5), 649-678. https://doi.org/10.1109/TSE.2010.62
13. Manoj Varma Lakhamraju, Prakhar Mittal, Vivek Agarwal; Impact Of Data Analytics In Business Process Optimization: A New Perspective International journal of applied engineering and technology 52232-–2412023romanpub
14. Mittal, Prakhar; Digital Transformation in Project Management Revolutionizing Practices for Modern ExecutionProject Management Information Systems: Empowering Decision Making and Execution1205-2322025IGI Global, 10.4018/979-8-3373-0700-8.ch006
15. Kapuruge, M., Gubala, J., & Colman, A. (2022). Process-Aware Service Composition for Modern Banking Systems. Service-Oriented Computing, 305-319. https://doi.org/10.1007/978-3-319-89963-3_20
16. Khan, A. (2023). Microservices Security in Financial Applications. Journal of Cybersecurity, 7(1), 9-15. https://doi.org/10.1093/cybsec/tyaa023
17. Yerra, S. (2025). The role of Azure Data Lake in scalable and high-performance supply chain analytics. Retrieved from https://ijsrcseit.com/index.php/home/article/view/CSEIT25112483
18. Lewis, J., & Fowler, M. (2020). Microservices Guide. Retrieved from martinfowler.com/microservices. https://doi.org/10.48550/arXiv.1606.04774
19. Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2021). Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media . https://doi.org/10.1145/3018896.3018913
20. Yerra, S. (2024). The impact of AI-driven data cleansing on supply chain data accuracy and master data management. Smart Computing Systems, 4(1), 187-191. https://romanpub.com/smc-v4-1-2024.php. https://doi.org/10.61485/SMCS.27523829/v4n1P1
21. Newman, S. (2022). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. https://doi.org/10.1145/2911992.2912017
22. P. Mittal, J. Singh Saini, A. Agarwal, R. K. Maheshwari, S. Kumar and A. Singh, "Fake News Detection Using Machine Learning Techniques," 2024 4th International Conference on Advancement in Electronics & Communication Engineering (AECE), GHAZIABAD, India, 2024, pp. 1374-1377, doi: 10.1109/AECE62803.2024.10911448
23. Pahl, C., Jamshidi, P., & Zimmermann, O. (2022). Architectural Principles for Cloud Software. ACM Transactions on Internet Technology, 18(2), 1-23. https://doi.org/10.1145/3104028
24. Richardson, C. (2023). Microservices Patterns: With Examples in Java. Manning Publications. https://doi.org/10.1007/978-1-4842-3858-5_6
25. Lakhamraju, M. V., Macha, K. B., Metha, S., Rai, A., & Miriyal, N. S. (2025). Best of breed vs. single suite: The strategic advantage of multi-tool integration in enterprise resource planning. Journal of Information Systems Engineering and Management, 10(23s). https://doi.org/10.52783/jisem.v10i23s.3763
26. Scholl, M., Fuernweger, A., & Pokrop, E. (2021). Transitioning Legacy Banking Systems to Microservices: A Case Study. International Journal of Software Engineering and Knowledge Engineering, 29(10), 1389-1410. https://doi.org/10.1142/S0218194019500414
27. Sharma, S., & Coyne, B. (2022). DevOps Adoption in Financial Services: A Comparative Study. IEEE Software, 32(1), 94-102. https://doi.org/10.1109/MS.2014.66
28. Metha, S. Automated code review for secure banking applications: Harnessing AI for security, performance, and regulatory compliance in financial software engineering. Journal of Information Systems Engineering & Management, 10(31s). https://doi.org/10.52783/jisem.v10i31s.5100
29. Vernon, V. (2021). Implementing Domain-Driven Design. Addison-Wesley. https://doi.org/10.1145/2362499.2362508
30. Wu, Y., Feng, Y., & Sun, J. (2023). Towards Automated Microservice Decomposition for Banking Systems. IEEE Transactions on Services Computing, 12(4), 580-593. https://doi.org/10.1109/TSC.2019.2931785
31. Rai, A., Mittal, P., Metha, S., Macha, K., & Venkat, R. (2024). Stochastic Modelling and Computational Sciences BLOCKCHAIN TECHNOLOGY: ITS ROLE IN TRANSFORMING DIGITAL PRODUCTS. Stochastic Modelling & Computational Sciences, 4(1), 2752–3829. https://romanpub.com/resources/smc-v4-1-2024-17.pdf

21. Taibi, D., Lenarduzzi, V., & Pahl, C. (2020). Architectural Patterns for Microservices: A Systematic Mapping Study. International Conference on Cloud Computing and Services Science, 221-232. https://doi.org/10.5220/0006798302210232
22. Ahmadvand, M., & Ibrahim, A. (2023). Requirements Engineering Challenges in Microservice Architecture: A Systematic Literature Review. Journal of Systems and Software, 175, 110899. https://doi.org/10.1016/j.jss.2021.110899
23. Bogner, J., Wagner, S., & Zimmermann, A. (2022). On the Impact of Service-Oriented Patterns on Software Maintainability: A Case Study Using Quality Metrics. International Conference on Service-Oriented Computing, 209-226. https://doi.org/10.1007/978-3-319-69035-3_15
24. Kiran Babu Macha. (2023).Advancing Cloud-Based Automation: The Integration of Privacy-Preserving AI and Cognitive RPA for Secure, Scalable Business Processes. International Journal of Computer Science and Engineering Research and Development (IJCSERD), 13(1), 14-43
25. Kodi, D. (2024). Data Transformation and Integration: Leveraging Talend for Enterprise Solutions. International Journal of Innovative Research in Science, Engineering and Technology, 13(9), 16876-16886.
26. Hasselbring, W., & Steinacker, G. (2023). Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. IEEE International Conference on Software Architecture, 243-252. https://doi.org/10.1109/ICSAW.2017.11
27. Alshuqayran, N., Ali, N., & Evans, R. (2022). A Systematic Mapping Study in Microservice Architecture. IEEE International Conference on Service-Oriented Computing and Applications, 44-51. https://doi.org/10.1109/SOCA.2016.15

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details