European Journal of Economic and Financial Research
ISSN: 2501-9430
ISSN-L: 2501-9430
Available on-line at: http://www.oapub.org/soc

DOI: 10.46827/ejefr.v9i3.2005 Volume 9 | Issue 3 | 2025

ARCHITECTURAL AUDIT OF HIGH-LOAD FINTECH SYSTEMS: ASSESSMENT METHODOLOGIES, COMMON VULNERABILITIES, AND REFACTORING RECOMMENDATIONS

Kovalenko Aleksandr[i]
Bachelor's Degree,
Tomsk State University of
Control Systems and Radioelectronics,
Russia, Tomsk

Abstract:
This article examines the approach to architectural auditing of high-load FinTech systems, taking into account the specifics of their operation under increased load and compliance with regulatory standards. The main methods of architectural assessment are studied, including expert-based and metric-oriented approaches, as well as the application of hybrid strategies that combine quantitative analysis with simulation testing. A range of typical architectural vulnerabilities is explored, such as excessive component coupling, lack of isolation for critical services, insufficient scalability readiness, and inadequate protection against failures and attacks. Recommendations for architectural refactoring are provided, aimed at enhancing system resilience and security in the long term.

JEL: G21, L86, C80

Keywords: architectural audit, FinTech systems, high load, vulnerabilities, fault tolerance, architectural design, refactoring, microservices

1. Introduction

The fast progress of financial technologies has been accompanied by an exponential increase in the demands placed on the resilience and performance of software systems. FinTech platforms serve millions of users in real time and process sensitive financial data, where even minor disruptions can lead to significant reputational and financial consequences. In this context, software architecture emerges as a critical determinant of system reliability and security.

Modern FinTech service design and maintenance methodologies require not just high degrees of automation and compliance with predetermined standards but also

[i] Correspondence: email stepa.mikhail@rambler.ru

continuous reexamination of the architectural design decisions. Architectural audits play a key role in such a case as a mechanism to analyze the system's maturity, reveal possible bottlenecks, and propose concrete refactoring recommendations. Such practices take on special relevance in the case of scenarios in which it becomes necessary to shift from monolithic to microservice architectures or to apply patterns supporting fault resilience and horizontal scalability, among others.

Architectural auditing in the FinTech domain presents unique challenges due to domain-specific requirements, the high cost of downtime, and increased scrutiny of cybersecurity practices. Unlike conventional systems, financial platforms are frequently subject to regulatory audits and must ensure transparency of operations, traceability of transactions, and resilience to external threats. The goal of this study is to systematize the methodologies employed in the architectural auditing of high-load FinTech systems, identify common architectural vulnerabilities, and provide well-founded recommendations for addressing them through architectural refactoring. The work examines current approaches to conducting architectural audits, including key reliability and scalability metrics, and analyzes typical architectural flaws encountered in financial technology platforms. It further explores refactoring practices aimed at improving system resilience and performance under conditions of high operational load.

2. Main part. Architectural audit methods and evaluation of high-load FinTech systems

High-load FinTech systems operate in an environment where the resilience of architectural decisions is directly tied to business risk. In such systems, architectural auditing is a formal process of determining the status of the software architecture, the aim of which is to discover constraints related to scalability, fault tolerance, security, and alignment with business goals. Architectural auditing differs from code review in that it deals with a more abstract level, including the evaluation of logical, technical, and infrastructural structures, the organization of component interdependencies, and the satisfaction level of the non-functional requirements.

The specific characteristics of the FinTech domain create a distinct auditing context. Financial platforms process large volumes of transactions with high operational density, requiring minimal latency, continuous availability, and strict compliance with regulatory standards such as Payment Card Industry Data Security Standard (PCI DSS), ISO/IEC 27001, and General Data Protection Regulation (GDPR) (table 1).

Table 1: Regulatory standards for FinTech systems [1, 2]

Standard	Description	Primary jurisdiction
PCI DSS	It focuses on securing credit card transactions and cardholder data.	Global, strong enforcement in the U.S.
ISO/IEC 27001	International standard for information security management systems (ISMS), emphasizing risk management and best practices.	Global
GDPR	It governs data protection and privacy for individuals within the European Union.	European Union, with extraterritorial reach

Sarbanes-Oxley Act (SOX)	U.S. regulation ensuring accurate financial disclosure and internal controls in public companies.	United States
Digital Operational Resilience Act (DORA)	EU regulation aimed at strengthening digital resilience of financial entities.	European Union

Under high-load conditions, it becomes critically important not only to select architectural patterns judiciously but also to evaluate their suitability in relation to actual traffic profiles, user characteristics, and service-level agreements (SLA). Architectural audits in this setting must account for both the current performance indicators and the system's capacity to adapt to increasing traffic volumes and the growing complexity of business logic.

Architectural audit methodologies can be broadly categorized into expert-based, metric-oriented, and hybrid approaches. Expert-based methods rely on the professional judgment of experienced architects and involve the analysis of design artifacts, technical documentation, architectural diagrams, and the results of interviews with developers and product stakeholders. The primary strength of these methods lies in their flexibility and adaptability to specific contexts, while their main limitation is reduced reproducibility and a high dependency on the auditor's individual expertise.

Metric-oriented approaches employ quantitative indicators to provide a more objective assessment of software architecture. These include metrics such as component cohesion and coupling, response time, availability, load profiling, and architectural risk evaluation using automated tools like SonarQube, ArchUnit, CAST Highlight, and Structure101. For instance, in the context of structural dependency control, ArchUnit enables the enforcement of architectural constraints through declarative rules (fig. 1).

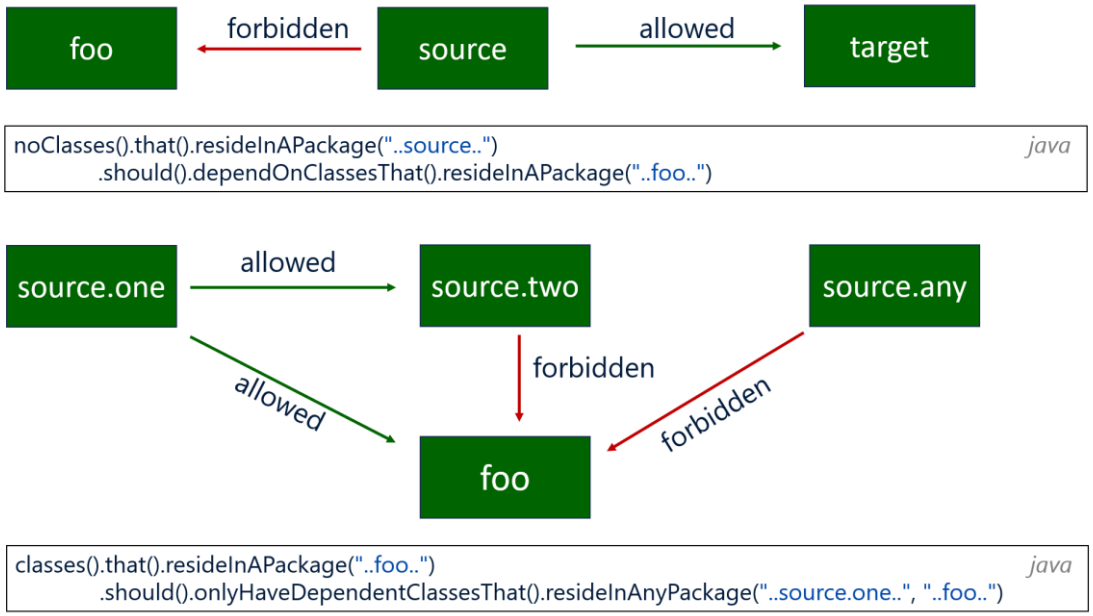

Figure 1: Example of package dependency rule enforcement in ArchUnit [3]

Hybrid approaches combine expert interpretation with quantitative data. Typically, these methods involve the formalization of auditor hypotheses into specific load or failure scenarios, which are then validated through practical testing techniques such as load testing, chaos engineering, and A/B experiments. Architectural viability can be assessed by simulating service-level failures and tracing their impact across the system, thereby identifying the degree of component coupling and the presence of single points of failure (SPOF). Architectural audits of high-load FinTech systems cannot be considered effective without accounting for the dynamic evolution of architectural decisions. An effective tool in this context can be a dependency map, which visualizes the current state of the system and the trajectories of increasing complexity [4].

Another essential dimension of architectural auditing involves assessing conformance to non-functional requirements (NFR), including security, scalability, reliability, and observability. For FinTech platforms, additional emphasis must be placed on transactional integrity, compliance with legal and regulatory frameworks, operational traceability, and the protection of personal data [5]. The presence of excessive abstractions, incompatible components, or outdated libraries can lead to increased response times and the accumulation of technical debt. For this reason, architectural audits must be closely integrated with CI/CD practices and DevOps toolchains.

In this light, architectural auditing of high-load FinTech systems should be understood as a multi-layered process that incorporates both static and dynamic analysis, combining expert judgment with quantitative metrics, and grounded in a thorough understanding of the business domain and regulatory constraints.

3. Common vulnerabilities and refactoring recommendations for FinTech system architectures

Financial technology systems impose stringent requirements for resilience, security, and predictable scalability. Even the most stable architectures might have weaknesses in the process of operational usage and thus pose issues capable of compromising the security or stability of the platform. Such weaknesses often stem from design compromises, tech debt, or legacy design choices that no longer match the needs of modern operational requirements.

One of the most common architectural weaknesses is overcoupling of the components, particularly in systems that have evolved without intentional modular design. A widely recommended mitigation strategy is the transition toward loosely coupled architectures, most notably, microservices or event-driven designs, where modules communicate asynchronously through message queues [6].

Another prevalent issue is the lack of clear isolation for critical components. In many cases, functions related to transaction verification, token management, one-time password (OTP) generation, or activity logging are implemented within shared services, complicating access control and observability. Refactoring efforts in such scenarios should aim to decouple security-sensitive components into dedicated services with strict

access restrictions enforced at both the network infrastructure and inter-service communication levels.

In practice, it is also common to encounter architectures where data storage and processing mechanisms are poorly suited to high transactional throughput. Centralized databases, for instance, frequently become bottlenecks as the volume of concurrent transactions increases. Architectural refactoring in such contexts may involve implementing data denormalization, transitioning to horizontally scalable storage solutions, or adopting a polyglot persistence strategy by employing specialized databases tailored to distinct workload profiles [7]. One common architectural solution to mitigate data contention and improve scalability is the adoption of the Command Query Responsibility Segregation (CQRS) pattern, which separates write and read operations into distinct models and services (fig. 2).

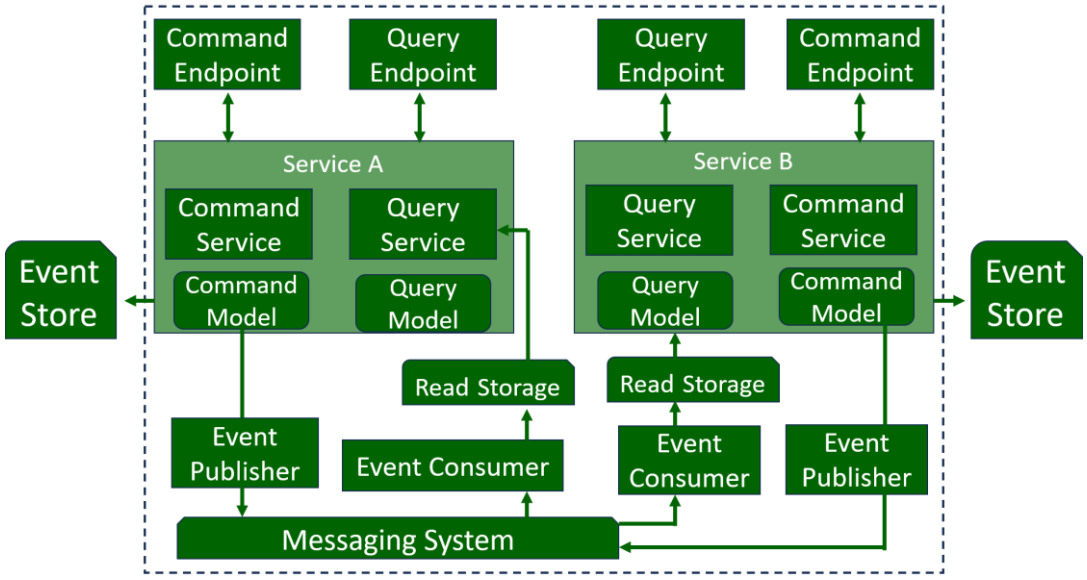

Figure 2: Generalized CQRS architecture with decoupled command and query models, event-driven communication, and polyglot read storage for service-specific workloads

Architectural decisions driven by the imperative for rapid growth often lack adequate provisions for fault tolerance. Common examples include configurations without redundancy for critical components, the absence of canary deployment mechanisms, and the inability to scale individual services independently. These design choices introduce significant risks of downtime during traffic surges or system updates. Recommended mitigation strategies include the adoption of service mesh infrastructures, automation of failure recovery scenarios (such as failover and fallback), implementation of horizontal scaling through Kubernetes, and the introduction of a dedicated load-balancing layer.

Security within FinTech architectures demands particular attention, as a number of recurring structural vulnerabilities are frequently observed. These include insufficiently isolated environments (e.g., shared infrastructure between staging and

production systems), direct database access from external API, lack of network perimeter segmentation, and flawed implementations of authentication and authorization mechanisms, such as misconfigured JWT tokens or insecure OAuth schemes [8]. Addressing them requires a fundamental reconfiguration of the infrastructure, including the centralization of secret management (e.g., through the implementation of HashiCorp Vault), revision of identity and access management (IAM) policies, and a broader transition toward a zero-trust architecture. A typical Zero Trust architecture model in distributed FinTech environments is characterized the integration of contextual sources such as Continuous Diagnostics and Mitigation (CDM), Security Information and Event Management (SIEM), Public Key Infrastructure (PKI), and Identity and Access Management (IAM) systems (fig. 3).

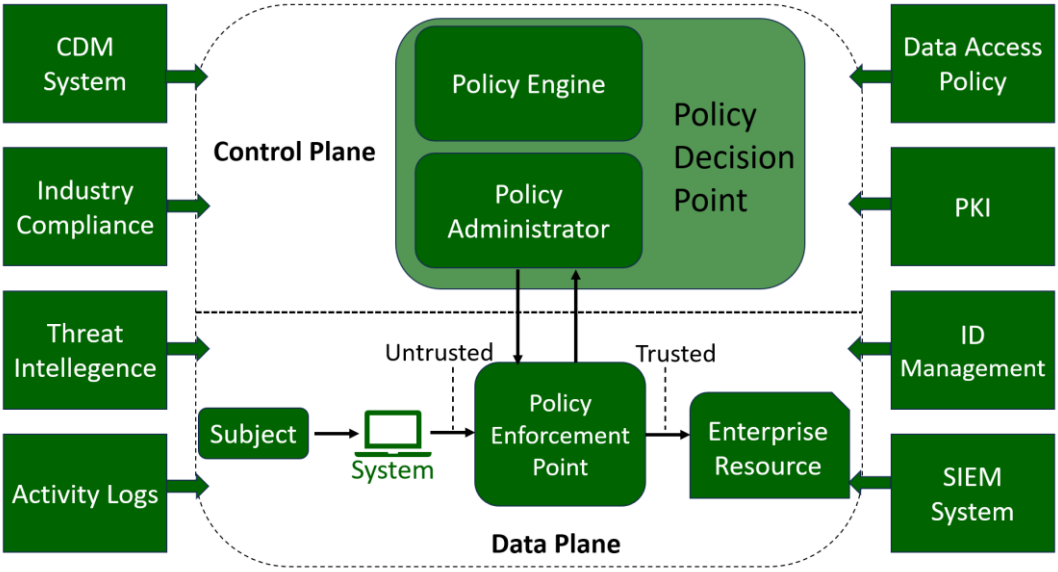

Figure 3: Core Zero Trust logical components [9]

FinTech architectures frequently lack built-in mechanisms to protect against system overload and denial-of-service (DoS) attacks. Common oversights include the absence of inbound traffic control (rate limiting), inefficient or nonexistent caching strategies, and failure to detect anomalies in user behavior. Such architectural shortcomings can lead to system outages not only due to malicious activity but also as a result of legitimate surges in user demand. To mitigate these risks, layered defense mechanisms must be implemented at the architectural level, ranging from application-level rate limiting and inter-service circuit breakers to active monitoring systems with automated incident response capabilities.

Architectural refactoring should not be confined to addressing existing flaws. It must be treated as a strategic process aimed at enabling the sustainable evolution of the system without increasing entropy. A crucial practice in this context is the adoption of continuous architectural evolution, developing living architectural documentation,

integrating architectural commits into the CI/CD pipeline, conducting regular architecture reviews, and automating the tracking of technical debt.

In summary, the typical vulnerabilities observed in FinTech platforms arise from both technical limitations and organizational deficiencies, particularly the absence of formal architectural governance processes. Systematic and proactive refactoring enables the elimination of inherent weaknesses, enhances the resilience of the code to failures, enhances the security stance, and ensures long-term sustainable growth.

In the future, FinTech system architectures will become more autonomous, smarter, and self-healing. The trends are to use AI-based observability tools that detect architectural drift automatically and suggest optimizations, use predictive analytics for capacity planning, and use serverless computing patterns that eliminate the need for long-lived infrastructure pieces [10]. Also, the shift toward IaC and GitOps pipelines will likely accelerate architectural innovation rates by tightly coupling infrastructure updates to versioned software lifecycles. As ecosystems become increasingly composable and interoperable, next-generation FinTech architectures will be embracing event-native design, decentralized orchestration, and API-first connectivity, making them more resilient, modular, and able to respond to rapidly changing markets.

4. Conclusion

Architecture audit of high-load FinTech systems serves as the key to guaranteeing resilience, scalability, and security of operation in the presence of increasingly mounting operational and regulatory loads. The application of formalized assessment methodologies, focused both on technical metrics and the specific characteristics of financial platforms, enables the identification of latent constraints and risks inherent in architectural decisions. Timely detection of common vulnerabilities, coupled with strategic architectural refactoring, reduces the likelihood of system incidents, enhances fault tolerance, and ensures the system's adaptability to future growth without compounding technical debt. Ultimately, a systematic architectural audit process supports informed decision-making and promotes long-term architectural sustainability. As FinTech ecosystems continue to evolve, such proactive practices will be essential in aligning technological capabilities with business agility and compliance requirements.

Conflict of Interest Statement

The author declares no conflicts of interest.

About the Author

Kovalenko Aleksandr, bachelor's degree, Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia.
ORCID: https://orcid.org/0009-0001-9683-0716
Email: amkovalenko90@rambler.ru.

References

Olaiya OP, Adesoga TO, Ojo A, Olagunju OD, Ajayi OO, Adebayo YO, 2024. Cybersecurity strategies in fintech: safeguarding financial data and assets. GSC Advanced Research and Reviews 20(1): 50–56. https://doi.org/10.30574/gscarr.2024.20.1.0241

Roszkowska P, 2021. Fintech in financial reporting and audit for fraud prevention and safeguarding equity investments. Journal of Accounting & Organizational Change 17(2): 164–196. https://doi.org/10.1108/jaoc-09-2019-0098

ArchUnit, 2025. ArchUnit User Guide. https://www.archunit.org/userguide/html/000_Index.html. Accessed 14 May 2025

Dashuber V, Philippsen M, Weigend J, 2021. A Layered Software City for Dependency Visualization. VISIGRAPP (3: IVAPP): 15–26. doi: 10.5220/0010180200150026. Retrieved from https://www.scitepress.org/Papers/2021/101802/101802.pdf

Topalidi A, 2025. Exploring architectural patterns for modular web applications: the Rails Engines approach to business logic isolation. Cold Science 13: 18–26. Retrieved from https://www.researchgate.net/publication/393987232_EXPLORING_ARCHITECTURAL_PATTERNS_FOR_MODULAR_WEB_APPLICATIONS_THE_RAILS_ENGINES_APPROACH_TO_BUSINESS_LOGIC_ISOLATION

Bolgov S, 2025. Optimizing microservices architecture performance in fintech projects. Bulletin of the Voronezh Institute of High Technologies 19(1). https://vestnikvivt.ru/ru/journal/pdf?id=1401. Accessed 14 May 2025

Conrad A, Utzmann P, Klettke M, Störl U, 2022. Metamodels to support database migration between heterogeneous data stores. In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings: 546–551. Retrieved from https://dl.acm.org/doi/10.1145/3550356.3561574

Zulkarneev I, Basalay KA, 2024. JSON Web Tokens Lifecycle-Based Threat Classification. 2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM): 1920–1924. http://dx.doi.org/10.1109/EDM61683.2024.10615042

NIST, 2020. Zero Trust Architecture. NIST Special Publication 800-207. p. 59. Retrieved from https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf

Seryan GN, Adamyan AV, 2025. Transformation of corporate financial strategies in the era of digital ecosystems. Professional Bulletin: Economics and Management 1/2025: 10–16. Retrieved from https://cyberleninka.ru/article/n/transformation-of-corporate-financial-strategies-in-the-era-of-digital-ecosystems