

RISK ANALYSIS OF MICROSERVICES ARCHITECTURE FOR A BANK'S INFORMATION SYSTEM

DOI: 10.5937/EMC26285M

Tamara Milić

University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Republic of Serbia

E-mail: tamaraamilic021@gmail.com

Biljana Radulović

University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin Republic of Serbia

Igor Vecštejn

University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Republic of Serbia

Vuk Amižić

University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Republic of Serbia

ABSTRACT

Microservices architecture is a modern approach to information systems development that enables greater scalability, modularity, and more flexible maintenance of software solutions. However, due to the system's distributed structure, microservices also introduce certain risks related to component interdependence, coordination between teams, system performance, and the reliability of data exchange itself. Such risks are particularly significant in banking information systems, where transaction accuracy, service availability, and user trust are of paramount importance. This paper analyses the microservices architecture of a banking information system using the Bank of Anthos application as an example. A particular focus is placed on identifying the critical dependency points between frontend services, backend microservices, and databases. The results of the analysis show that risks in microservice architecture are not just a technical problem, but also a challenge from the perspective of management, organization, and coordination. The timely identification and control of such risks can contribute to greater system reliability, better user experience, and the preservation of banking services in the long term.

Keywords: Risk Management, Microservices, Microservice Architecture, Information System

INTRODUCTION

Data processing, in an attempt to obtain well-organized information, is a significant and time-consuming task for businesses, as large and growing amounts of data require further processing. However, advancements in computer systems have greatly contributed to the development of systems to simplify specific practices associated with data processing, particularly the collection, storage, processing, and analysis of data, as well as the extraction and distribution of information for specific purposes. Such systems are called Information Systems (IS), which play an important role within an organization with the aim of increasing the efficiency of management decision-making (Nikoloski, 2014; Taherdoost, 2021). In addition to the general purposes of an information system, its main advantages within an organization can be listed as accessing large amounts of data, making efficient decisions in an automated manner, reducing human error through the use of complex algorithms, and assisting in the development of new approaches to generating revenue and increasing the number of customers. An excellent example of the importance of an information system within an organization is the e-commerce company Amazon, whose business success is largely based on the implementation of a microservices architecture (Mutwiri, 2020; Taherdoost, 2021; Stricker et al., 2018). Although microservices architecture contributes to scalability and more efficient development of information systems, its structure requires a certain complexity, and therefore various risks can arise in the system's operation.

In addition to the wide range of benefits and success factors that an information system offers, there are also certain challenges that limit its implementation and can lead to potential risks affecting the organization's performance. These include: (i) the constant, day-to-day changes in technology, which require up-to-date users to be in a state of continuous learning, (ii) security issues related to suppliers, banking and the distribution of core data, intellectual property and company records, (iii) inefficient establishment of an information system due to the high costs and expenses associated with it. The sheer number of risks and risk factors associated with business and information systems establishes a driving force towards the development of an organizational framework for managing them and responding accordingly (Taherdoost, 2021).

MICROSERVICE ARCHITECTURE IN THE CONTEXT OF RISK MANAGEMENT

The emergence of microservices is closely linked to the early 2000s, when companies like Amazon began to transition from monolithic architectures to architectures based on loosely coupled services. Amazon, as one of the world's most well-known e-commerce companies, was (and remains) one of the pioneers in adopting this architectural style to accelerate software development (Peša, 2025). The application of Microservice Architecture (MSA) spans multiple industries, from technology to e-commerce and banking, as well as various development environments, from cloud computing to on-premise solutions (Lewis & Fowler, 2014; Amri *et al.*, 2024). MSA breaks down a monolithic application into a set of smaller, discrete, and interconnected services (Amri *et al.*, 2024; Newman, 2015). The microservices approach to software development offers an alternative to the conventional monolithic style. Although the advantages of microservices-based development over the monolithic style are clear, industry experts agree that neither style provides an absolute advantage in all situations. Proponents argue that the microservices approach to software development facilitates the mapping of organizational changes, which manifest from a more dynamic business environment, into corresponding IT/IS (Information Technology/Information Systems) changes. Each microservice corresponds to a single function around a business capability. Each one contains its own data resources and is deployed quickly. Examples range from paying bills online on a banking website to an alternative flight service on an airline's website. Although the benefits are clear, there are significant challenges to the adoption and maintenance of MS-based software development. Many of these challenges stem from the novelty of the approach and a lack of understanding of the technical and organizational parameters that affect MS-based development. If these challenges are not well understood, companies are unlikely to realize the full benefits of MSs and may be in an even worse position than before modernization (Vitharana & Daya, 2024). Consequently, under certain circumstances, the challenges can be risky for a company's business.

Risk classification

One of the challenges of microservices architecture is that teams developing different microservices become more isolated from the system, and collaboration between teams becomes a greater challenge (Söderqvist & Göthe, 2022; Jamshidi *et al.*, 2018). Implementing corrective maintenance to resolve an issue spanning multiple microservices becomes a complicated procedure for teams, as it requires cross-team communication. Figure 1 illustrates the problem (Söderqvist & Göthe, 2022).

In Figure 1, all the lines connecting the different services represent the links between microservices, i.e., the points where they communicate and exchange data. The red lines indicate which microservices may be affected when a problem occurs in one microservice, as this problem can also lead to difficulties in other related microservices. If multiple microservices are affected, teams must collaborate to resolve the issue. The yellow dashed lines represent the collaboration between teams (Söderqvist & Göthe, 2022). The specific case illustrated in Figure 1 indicates the interconnection of microservices and communication between multiple teams within the company, which represents a risk concerning coordination and dependency between components.

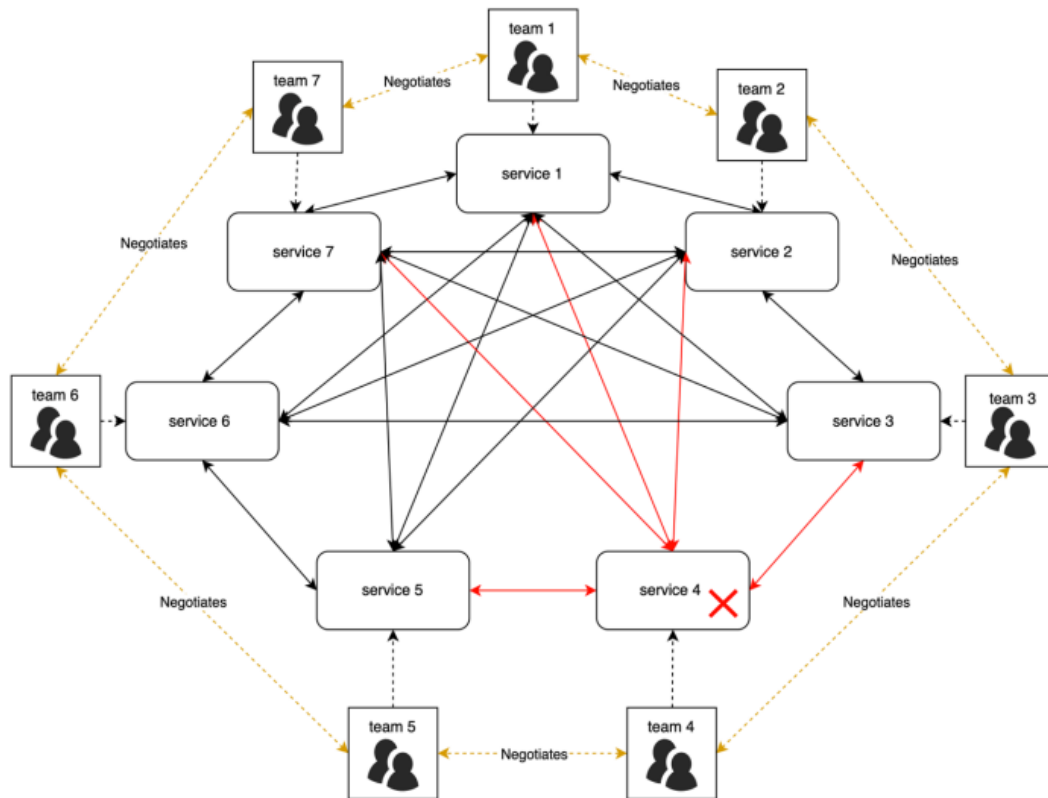


Figure 1: Illustration of microservice architecture with an issue (Söderqvist & Göthe, 2022)

A significant risk in microservices architecture is the risk caused by insufficient system performance. The fine-grained decomposition of microservices makes the system vulnerable to performance issues. Today's microservices systems consist of tens of thousands of microservices. Consequently, each microservice can suffer from performance issues due to many unpredictable problems, such as network failures, host issues. Furthermore, the control flow and data flow for processing a single request can traverse hundreds of microservices; a performance issue with any one of them will affect the entire request. Therefore, to ensure system performance and reliability, it is essential to monitor and efficiently diagnose performance issues in every microservice. If a system's performance degrades, the consequences can include general customer dissatisfaction, slower processing of business processes, and an increase in system maintenance costs. Figure 2 shows an example of a request flow in a microservices system. A simple request on the front-end can cause a problem in a lower-level microservice and further affect numerous higher-level microservices (Hou et al., 2021).

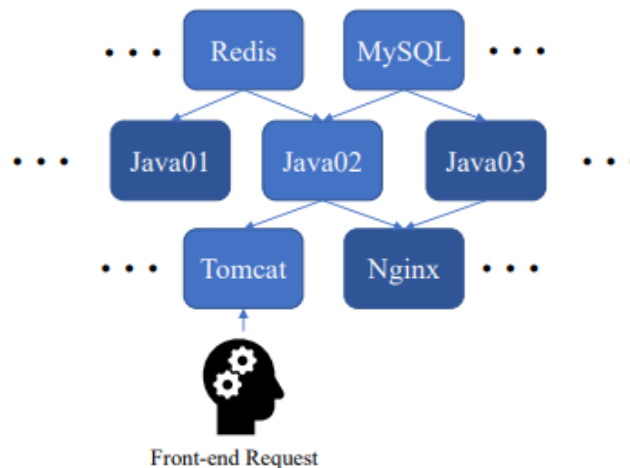


Figure 2: An overview of the request flow in a microservices architecture (Hou et al., 2021)

BANK INFORMATION SYSTEM

This section will present the adaptive microservices architecture diagram for the Bank of Anthos application. Bank of Anthos is an example of a multi-tier, HTTP-based web application that simulates a bank's payment processing network, allowing users to create artificial bank accounts and perform transactions. Google uses this application to demonstrate how developers can modernize business applications using cloud-native technologies and Kubernetes-based deployment. This application can run on any Kubernetes cluster (GoogleCloudPlatform, n.d.). Figure 3 represents the adapted Bank of Anthos application model used in this paper.

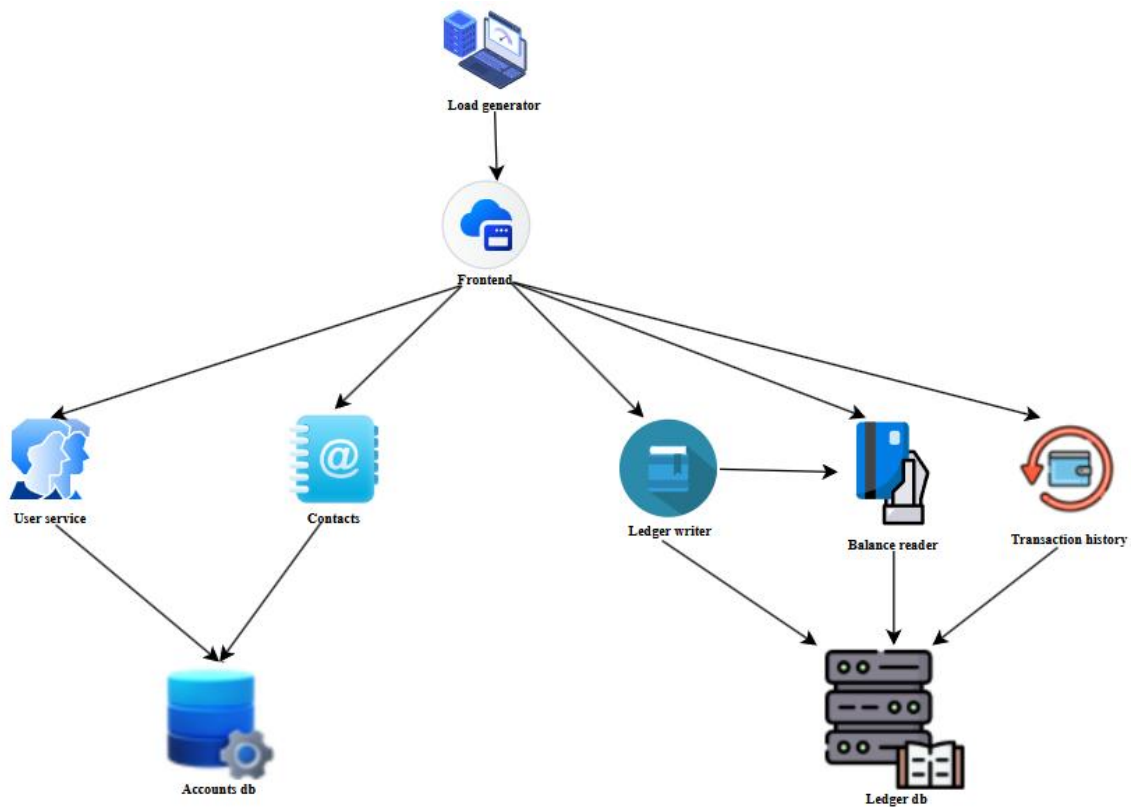


Figure 3: Bank of Anthos application diagram

Based on Figure 3, it can be observed that the presented diagram contains five backend microservices, namely: User Service, Contacts, General Ledger Entry Service, Balance reader, and Transaction History. The load generator tests the system by simulating the sending of a large number of requests. The frontend is also a microservice; it is the part of the application visible to users, serving for user login, viewing account balances, creating transactions, etc. The user service (user-service) is responsible for user authentication and authorization. Contacts provide an overview of all the contacts a user has, allowing them to add new ones, edit or delete existing ones. The accounts database (accounts-db) is an example of a relational database that contains the necessary information about users and user accounts. User service and contact data are stored in the accounts database. The ledger-writer service first validates the entered transactions and then saves them permanently. The general ledger entry service is connected to a balance reader, which allows the reader to see the current balance of a user's account. The transaction history provides data on previously executed transactions for a specific user, while the ledger database, as the main transaction database, stores all necessary transaction data. The data flow in the system circulates from the load generator to the frontend service by sending the appropriate requests. The frontend then sends a request to a specific microservice, depending on the activity the user is requesting. Requests concerning contacts and user data will be directed to the User service and Contacts microservices, which communicate with the Accounts database. Conversely, requests concerning the Ledger writer, Balance reader, and Transaction history will be directed to the microservices of the same name, which communicate with the Ledger database.

ANALYSIS AND DISCUSSION

The risk analysis in this paper was conducted based on an overview of communication flows and functional dependencies among microservices and components shared for data storage. Critical points are identified according to the following criteria: (1) whether the component participates in communication with multiple services, (2) whether its failure affects the functionality of the system, (3) whether the consequences of the failure may have an impact on user activities, service availability, or transaction data accuracy. Based on these criteria, three critical points were identified: (1) frontend - user-oriented communication layer, (2) Accounts DB - common database for user and contact data, and (3) Ledger DB - central component of the transaction system.

Figure 4 shows the risk analysis of coordination and dependencies among the current components in the system, based on the previously described application architecture.

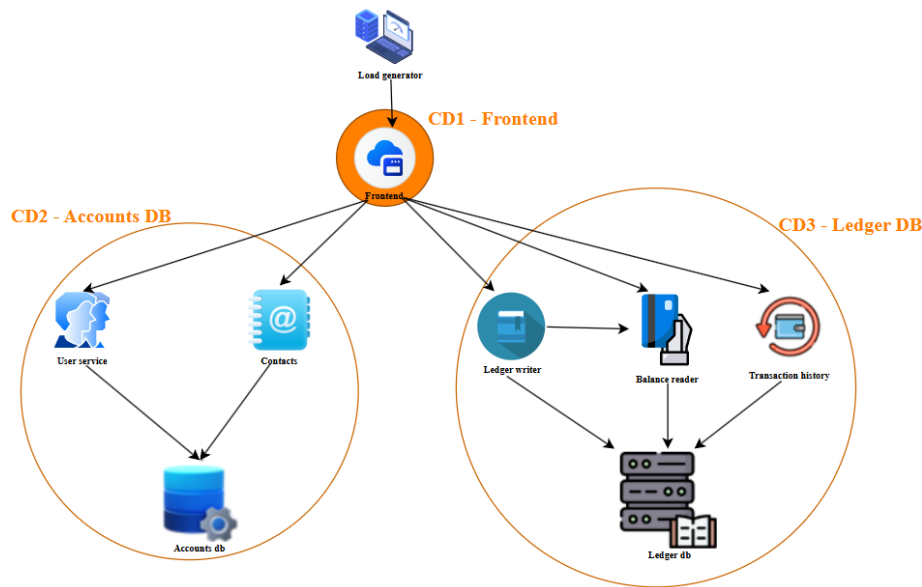


Figure 4: Analysis of coordination and dependency risks among components

Marked critical points indicate clusters of dependencies where component failure may affect related services, user-facing functions, or transaction-related operations.

- Critical dependency point (CD1 – Frontend) – The part of the application visible to the user is the frontend. When the user wants to view their account balance, the frontend sends a request to the corresponding backend microservice in the background. If a failure or delay occurs with the given backend microservice when forwarding the request, the user does not receive a correct response, or the response arrives with a delay. The consequences can include degraded user service, an increased number of user complaints, reduced service quality, slower application performance, and increased customer support costs.
- Critical dependency point (CD2 – Accounts DB) – The critical point consists of the User service, Contacts, and Accounts DB. When a user attempts to log in with a user account, the User service accesses the database to retrieve the necessary user information. This means that the user service and user contacts data are stored in the Accounts DB. Should a problem occur with the aforementioned database, the consequences are not limited to a single service but can affect both the User service and Contacts. This risk can result in login and authentication issues, user dissatisfaction, increased maintenance costs, and an increase in customer support requests.
- Critical dependency point (CD3 – Ledger DB) – This critical point represents the transactional part of the system and is composed of: Ledger writer, Balance reader, Transaction history, and Ledger DB. If an incorrect transaction entry is made by the Ledger writer, the Balance reader will display an inaccurate account balance, and the Transaction history will show incorrect transaction history. Furthermore, as all transaction data is stored in the Ledger DB, an error can

affect multiple microservices simultaneously. The consequences can include reputational risk for the bank, a loss of competitive advantage, a breach of customer trust, and jeopardized continuity of transactional processes.

CONCLUSION

The analysis of the microservice architecture of the Bank of Anthos application shows that risks arise not only from individual components but also from dependencies between services in the system, database, and user-oriented communication flows. The paper identified three critical dependency points: the frontend communication layer, Accounts DB, and Ledger DB. The frontend is critical because it mediates the process of forwarding user requests to backend services, which is why any failure or delay can affect service availability and lead to a poor user experience. Accounts DB represents a common data dependency for services related to users and contacts. Ledger DB represents a particularly sensitive point because it supports transaction writing, balance reading, and transaction history display, whereby inconsistencies may affect the accuracy of transaction data and user trust.

These results confirm that risk management in banking microservice architectures requires both technical and organizational control. From a technical perspective, it is necessary to monitor system service performance, database availability, and transaction data consistency. From an organizational perspective, it is necessary to define clear team responsibilities and coordination mechanisms in order to reduce the possibility of cascading system disruptions and preserve the reliability of the bank's services.

REFERENCES

- Amri, M. G., Raharjo, T., Fitriani, A. N., & Hutasuht, N. R. P. (2024). Critical success factors of microservices architecture implementation in the information system project. *International Journal of Advanced Computer Science and Applications*, 15 (10). <https://doi.org/10.14569/IJACSA.2024.0151064>.
- GoogleCloudPlatform. (n.d.). Bank of Anthos. GitHub repository. Available at: <https://github.com/GoogleCloudPlatform/bank-of-anthos/blob/main/README.md>
- Hou, C., Jia, T., Wu, Y., Li, Y., & Han, J. (2021). Diagnosing performance issues in microservices with heterogeneous data source. In *2021 IEEE International Conference on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking*. Los Alamitos, CA: IEEE, 493-500. <https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00074>
- Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. *IEEE Software*, 35 (3), 24-35. <https://doi.org/10.1109/MS.2018.2141039>
- Lewis, J., & Fowler, M. (2014). Microservices: A definition of this new architectural term. Available at: <https://martinfowler.com/articles/microservices.html>
- Mutwiri, W. (2020). *Amazon Business Information Systems: Data Acquisition and Management in Its Value Chain*. Munich: GRIN Verlag
- Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA: O'Reilly Media
- Nikoloski, K. (2014). The role of information technology in the business sector. *International Journal of Science and Research*, 3 (12), 303-309
- Peša, M. (2025). *Uzorci dizajna u mikroservisnoj arhitekturi* (Doctoral dissertation, Sveučilište u Zagrebu, Sveučilište u Zagrebu Fakultet organizacije i informatike, Katedra za razvoj informacijskih sustava)
- Söderqvist, E., & Göthe, D. (2022). Maintain multiple microservices without compromising the autonomy of Agile teams: Developing a strategy for increased productivity in maintenance of microservices.
- Stricker, R., Müssig, D., & Lässig, J. (2018). Microservices for redevelopment of enterprise information systems and business processes optimization. In *Proceedings of the 20th International Conference on Enterprise Information Systems*, Vol. 2. Setúbal, Portugal: SCITEPRESS, 719-726. <https://doi.org/10.5220/0006791607190726>
- Taherdoost, H. (2021). A review on risk management in information systems: Risk policy, control and fraud detection. *Electronics*, 10 (24), 3065. <https://doi.org/10.3390/electronics10243065>
- Vitharana, P., & Daya, S. A. (2024). Challenges in adopting and sustaining microservice-based software development: Organizational challenges can be more difficult than technical ones. *ACM Queue*, 22 (1), 48-72. <https://doi.org/10.1145/3649402>