Microservices Design Patterns with Java

70+ patterns for designing, building, and deploying microservices

Sergey Seroukhov

www.bpbonline.com

First Edition 2024

ISBN: 978-93-55517-005

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My parents ***Anatoly*** *and* ***Ludmila***
my wife ***Natalya****, and kids* ***Michael*** *and* ***Alexandra***

About the Author

Sergey Seroukhov, a passionate Technology Evangelist, resides in Tucson, AZ, with his wife Natalya and two children, Michael and Alexandra. He is the visionary founder of Enterprise Innovation Consulting, a boutique consulting firm that empowers development teams to embrace modern development methods, enhance productivity, reduce costs, accelerate time to market, and foster innovation.

Enterprise Innovation Consulting(https://www.entinco.com/) was founded in 2016, driven by the dream of helping software teams build more and better software faster. With his group of talented engineers, Sergey assisted multiple organizations in developing complex enterprise systems utilizing microservices, microfrontends, and DevOps. Moving the business to the next level, Enterprise Innovation Consulting, under Sergey's leadership, created several programs called "Better Microservices (https://www.entinco.com/programs/better-microservices)," "Better Microfrontends (https://www.entinco.com/programs/better-microfrontends)," "Better Delivery (https://www.entinco.com/programs/better-delivery)," and "Better Testing (https://www.entinco.com/programs/better-testing)" that aimed at drastically improving development productivity through standardization of architecture and implementation of development patterns and practices. Moreover, all those programs represented the 1st step toward the "Software Factory", a new development model that brings a step-change in productivity and cost reduction through standardization, deeper specialization of labor, and conveyor-like development processes. Emerging Generative AI combined with the Software Factory (https://www.entinco.com/programs/software-factory) model represents a perfect fit, allowing the systematic and incremental increase of automation in software development until it finally reaches the "Light-off Factory" state when most of the software is generated automatically. The world is not there yet, but the work of visionaries like Sergey Seroukhov and companies like Enterprise Innovation Consulting are making that future come sooner.

Sergey's journey in coding began at the age of 14, and he implemented his first commercial software product using dBase around 1991, even before graduating from high school. After completing his master's degree at Donetsk State Technical University in 2001, he embarked on a new chapter in the United States with his wife, Natalya. Over the next two decades, Sergey honed his skills, working as a Software Developer, Team Lead, Solution Architect, and eventually as a CTO in several startups. His foray into microservices started around 2005, leading the creation of a distributed system architecture composed of loosely coupled services and composable frontends. Since 2012, when microservices gained recognition, he has been instrumental in the development of numerous microservices systems, using a wide range of programming languages like .NET, Java, Node.js, Go, Python, and Dart.

About the Reviewers

- **Praharsh Jain** is a passionate programmer and an information security enthusiast with hands-on experience developing web, mobile and desktop applications using multiple technology stacks.

  As a seasoned engineering leader, he loves tackling complex problems and mentoring other team members. He has extensive professional experience in creating highly scalable backends leveraging technologies like Java, Golang and Node.js.

  With a background in Computer Science, Praharsh possesses strong knowledge of CS fundamentals.

  He is currently working with Grab as a part of the Risk Engineering team creating and maintaining systems that are pivotal in evaluating millions of real-time transactions to prevent fraud.

- **Venkata Karthik Penikalapati** is a distinguished software developer with a decade's expertise in distributed systems and AI/ML pipelines, holding a Master's in Computer Science from the University at Buffalo. At Salesforce's Search Cloud, Venkata drives Data, AI and ML innovation, showcasing his influence in tech. An accomplished speaker and published author, his insights resonate at conferences and in scholarly papers, highlighting his thought leadership. Venkata's role in evolving AI-driven solutions marks him as a pivotal figure in technology's future.

Acknowledgement

The journey of writing this book has been a profoundly rewarding experience, made possible by the unwavering support and contributions of several key individuals and groups.

My deepest gratitude goes to my family, whose encouragement and belief in my work have been the bedrock of my motivation. Their support has been invaluable throughout this process.

The team at Enterprise Innovation Consulting has played a crucial role, sharing their expertise and experiences in microservices systems, which have greatly enriched the content of this book. Their dedication has been instrumental to our collective success.

I extend my thanks to Venkata Karthik and Praharsh Jain for their meticulous technical review, which has significantly enhanced the book's quality. Eugenio Andrieu's editing and Danil Prisyazhniy's preparation of code samples have been vital in ensuring the clarity and applicability of the material presented.

The entire team at BPB Publication has been exceptional in guiding me through the book writing and publishing process, helping to refine and polish the content to fit within the confines of the book without compromising its richness.

This book is a testament to the collaboration, expertise, and support of each individual mentioned and more. To everyone involved, thank you for helping turn this vision into reality.

Preface

In the evolving landscape of software architecture, microservices have emerged as a cornerstone for building scalable, resilient systems**. Microservices Design Patterns with Java** is crafted for professionals navigating this complex domain, offering over 70 design patterns and practices essential for developing robust microservices. Tailored for architects, team leads, developers, and DevOps engineers with a solid grounding in microservices and Java, this book serves as a comprehensive guide to mastering the intricacies of microservices architecture.

With a practical approach, we present patterns ranging from architectural design to deployment, each accompanied by Java code examples. This format allows readers to apply the concepts directly to their projects, facilitating a deeper understanding and immediate implementation. The book is structured as a flexible reference, enabling professionals to explore topics in any order and apply patterns to various challenges.

Distinguishing itself in a crowded field, this publication targets experienced practitioners, offering a concise compilation of established and emerging patterns. It aims to equip readers with the knowledge and tools to tackle the challenges of microservices development, ensuring the delivery of efficient, scalable, and reliable systems.

Chapter 1: Defining Product Vision and Organization Structure - Explores the importance of aligning microservices with organizational structure and product vision for successful implementation.

Chapter 2: Architecting Microservices Systems - Introduces architectural patterns for decomposing systems into microservices, covering communication styles, security models, and deployment strategies.

Chapter 3: Organizing and Documenting Code - Discusses best practices for structuring and documenting microservices code to ensure maintainability and scalability.

Chapter 4: Configuring Microservices - Covers various configuration strategies for microservices at different lifecycle stages, emphasizing dynamic configuration for flexibility.

Chapter 5: Implementing Communication - Details synchronous and asynchronous communication patterns, including HTTP/REST, gRPC, and message-driven approaches, ensuring efficient service interaction.

Chapter 6: Working with Data - Presents data management patterns for microservices, including CRUD, CQRS, event sourcing, and strategies for database architecture.

Chapter 7: Handling Complex Business Transactions - Explores patterns for managing business transactions in microservices, including state management, distributed transactions, and reliability strategies.

Chapter 8: Exposing External APIs - Discusses designing and securing external APIs, highlighting the importance of API gateways, authentication, and versioning for external integration.

Chapter 9: Monitoring Microservices - Introduces monitoring strategies for microservices, covering logging, metrics collection, distributed tracing, and health checks.

Chapter 10: Packaging Microservices - Explores packaging strategies for deploying microservices across various platforms, including Docker, serverless, and traditional JEE servers.

Chapter 11: Testing Microservices - Details patterns for automating microservices testing, covering both functional and non-functional aspects to ensure robustness and performance.

Chapter 12: Scripting Environments - Discusses the use of scripted environments for efficient microservices delivery, emphasizing automation in infrastructure management.

Chapter 13: Automating CI/CD Pipelines - Introduces continuous integration and continuous delivery pipelines tailored for microservices, focusing on incremental delivery and secure deployment strategies.

Chapter 14: Assembling and Deploying Products - Provides a guide to assembling and deploying microservices-based products, covering product packaging, version management, and deployment strategies.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/b4tfcj2

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Microservices-Design-Patterns-with-Java.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Defining Product Vision and Organization Structure

Introduction

In recent years, there has been a significant rise in the adoption of **microservices**. An increasing number of software teams are currently either engaged in developing microservices or considering them. Numerous publications on the subject offer various patterns, and multiple technologies pledge to simplify microservice development. Nonetheless, eight out of ten organizations that have adopted microservices are encountering significant issues that prevent them from meeting their initial expectations. The sources of these problems are seldom purely technical.

The key to success in the development of microservice systems is understanding that *microservices* are not just an *architectural style*. A microservice is a software component with an independent lifecycle. It can be built by different teams, at different times, using different technologies, and delivered independently into production. Achieving that kind of independence requires not only technical decisions. It touches all areas of software development, including organization structure, product, and project management.

This chapter introduces you to patterns at the organization and product level that help solve this problem by setting the right structure and direction to ensure success in microservices development.

Structure

In this chapter, we will cover the following topics:

- Microservices Adoption Goals
    - Scalability
    - Productivity
    - Time to Market
    - Innovation
- Incremental Delivery
- Development Model
    - Agile Workshop
    - Software Factory
- Organization Structure
    - Feature Delivery Teams
    - Platform Teams
    - Integration Team
- Microservices Adoption Process
- Antipatterns

Objectives

After studying this chapter, you should be able to set clear goals for microservice adoption, define an appropriate organizational structure, adopt an incremental delivery model, and assign clear roles and responsibilities to your team. Furthermore, this chapter explores different development models and introduces the Software Factory development model, which facilitates substantial increases in development productivity and cost reduction.

Microservices adoption goals

The concept of microservices extends beyond the technical realm, encompassing a **divide and conquer** strategy that empowers users to overcome the mounting complexity of software. However, embarking on a microservices journey is not a simple task. Success requires the involvement of both technical and non-technical teams, and necessitates a strong rationale linked to the organization's overall business vision.

Problem

Microservices have been a prominent topic in recent years, with numerous success stories shared by industry giants such as Amazon, Google, and Facebook (as depicted in *Figure 1.1*). Such achievements have inspired others to follow suit and adopt this approach to development. However, this decision is often taken solely by the technical team, lacking clear justification or support from management or other stakeholders.

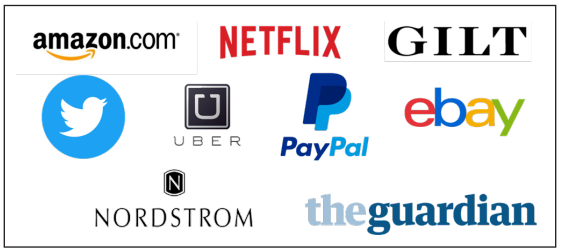

***Figure 1.1**: Large companies that shared their success stories about microservice adoption*

Often, when teams opt to develop microservices, they tend to handle the technical aspects correctly. Their code appears to be structured as microservices and may function as such. However, when it is time to work on a subsequent release, the team is overwhelmed with the extensive amount of work they must complete. Then, after several months of hard labor, they eventually managed to produce a long-awaited release.

This problem happens because teams continue to operate with a monolithic mindset and follow monolithic processes, even though microservices provide new possibilities. To fully leverage the advantages of microservices, it is critical to start with well-defined objectives that are aligned with broader business goals, approved by management, and reinforced by other teams. Collaboration is essential to achieving success.

Scalability

Microservices were initially championed by major internet corporations such as Facebook, Netflix, and Amazon as a solution to the challenges of scaling monolithic systems that were inefficient at serving millions of users simultaneously. Large components consumed excessive resources, with many of them underutilized. By dividing their monolithic systems into smaller, independent chunks, they were able to improve resource utilization and scale each component separately.

Although most organizations do not require such extensive scalability, it remains the primary factor for selecting microservices, as per the responses of people when asked why they opted for this approach.

Nonetheless, it is important to recognize a few key points:

- Microservices are not the only way to achieve scalability. Besides, many organizations just do not need that kind of scale. It's crucial to understand that while microservices offer one approach to scaling, there is a common misconception that they are the only option available.
- Regarding scalability, computational issues are not always the primary bottleneck in the system. Bottlenecks can manifest in various areas, not exclusively in microservices. In many cases, it has been observed that the most significant bottlenecks arise at the database level.

Although scalability is often considered the primary objective, it may only be relevant to a small group of software organizations, specifically those that anticipate exponential growth. An example is SaaS companies, which aim to safeguard their code investments by ensuring that they can handle heavy loads should success arrive.

It is also important to understand that when scalability is identified as a target, it is critical to establish a specific metric for it. Additionally, it is necessary to investigate all areas of the architecture that may contain bottlenecks, not only the microservices' backend.

Productivity

The second most popular reason for adopting microservices is probably the desire for higher development productivity. This belief has been reinforced by success stories from larger companies, leading people to view microservices as a guaranteed way to improve productivity.

Unfortunately, many who have adopted microservices have been surprised by the discovery that their productivity has significantly decreased. In addition to writing regular code, they now spend a great deal of time coding communication, troubleshooting difficult issues, and building and maintaining multiple CICD pipelines. What used to be a single software release has now become many, making their lives much more difficult. Almost without exception, the stories behind such cases involve a distributed monolith that stems from an old mindset, inefficient organizational structures, and monolithic development practices.

To enhance development productivity, microservices can be a valuable tool. However, to fully capitalize on their potential, the organization and development model must undergo a transformation, and the ability to compartmentalize the work across microservices should be used to its fullest extent. In addition,

- Product releases should be incremental, delivering a few features that require changes in a small number of microservices.
- Those microservices that have been modified should be the only ones eligible for development, testing, and release.

- Developers should focus only on assigned microservices and not spend their time and mental energy thinking about the entire system.
- DevOps engineers should assemble the system from the microservices, treating them as black boxes.
- Microservice implementations should be standardized and templated, so every new microservice gets to be a close copy of others. In this manner, developers do not need to think about how they should write the code.

Productivity can be set as a goal to deliver more features with limited resources, in order to achieve business growth and market domination. However, it is crucial for the organization to have clarity on this matter. This includes an understanding of past and current productivity levels, identifying bottlenecks that cause productivity to decline, and defining the precise development process required to achieve higher productivity.

Time to Market

In today's saturated market, vendors that deliver a new idea can quickly capture a big portion of the market, and those who come after them usually have a very hard time to battle uphill. That's why Time to Market can be extremely important for software companies that experience high competition.

Although microservices can potentially reduce Time to Market significantly, their implementation often fails to achieve this goal due to a lack of alignment among team members and uncertainty about how to achieve it. While microservices are intended to provide a solution, the reality can be different, resulting in a larger and more complex system with numerous components to be released and integrated, ultimately slowing down the release cycle. The root cause of this issue is typically the result of a monolithic mindset, development practices, and organizational structure that leads to a distributed monolith.

To release faster, the team should adopt the Incremental Delivery model. This involves selecting a small set of features that can be released independently and only affect a limited number of components. The team can then develop, test, and release these features while leaving the rest of the system unchanged. Importantly, product management should also be involved in defining a small set of features with high business value that will incentivize customers to purchase or upgrade (refer to the Incremental Delivery Pattern).

Innovation

Innovation is another popular goal for microservices adoption that we discuss in this chapter. It can be related to functional innovation: delivering new features quickly. Or it could be a technological innovation: using the latest technologies, integrating scanners, AI, etc.