ENGINEER-TO-ARCHITECT AND DURABLE THINKING SKILLS

Software Architecture Patterns and Styles

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Every architecture style is a bet: layered architecture bets on simplicity, event-driven bets on responsiveness and decoupling, microservices bets on independent change. Teams get into trouble not because they pick obscure styles but because they pick familiar ones without understanding the bet they are making. Knowing the catalog is easy; knowing which tradeoffs each style buys you, and against which architecture characteristics, is the skill.

This is a hands-on, practitioner course. It builds the judgment first: a method for evaluating any style against the characteristics a system needs, so every style studied afterward lands in the same framework. Then it works through the styles in a deliberate gradient, from the monolithic family through the distributed family, ending with the honest craft of choosing, combining, and evolving styles in real systems. We go deep on the styles you will actually choose between rather than cataloging every named architecture in the literature. Every module ends with a design lab, and each module builds on the one before.

Who Should Attend

- Developers and technical leads who choose or defend system structure
- Architects who want a rigorous, tradeoff-driven command of the style catalog
- Engineers inheriting a system and trying to understand why it is shaped the way it is

Learners who want the broader discipline first, including architecture characteristics in depth, should take *Fundamentals of Software Architecture*.

Prerequisites

- Solid professional software development experience
- Basic familiarity with architecture characteristics (the "-ilities")
- Comfort sketching and discussing system diagrams

What You Will Learn

- Distinguish architecture styles from patterns and evaluate any style against prioritized characteristics
- Explain the layered, modular monolith, and microkernel styles and where each wins
- Design an event-driven architecture and reason about its failure modes
- Compare service-based and microservices styles and the distribution costs they share
- Judge which style fits a given problem, and combine styles into defensible hybrids
- Plan the evolution of a system from one style toward another

Course Outline

Day one: the framework and the monolithic family

- How to Judge a Style

  - Styles versus patterns, and why the vocabulary keeps them straight
  - Rating styles against architecture characteristics: a scorecard you will use all course
  - Topology, deployment, and data: the three questions to ask of any style
  - Lab: build a characteristics scorecard for a case-study system, ready to rate every style against
- Layered and Modular Monolith
  - Layered architecture: strengths, sinkholes, and why it is the accidental default
  - The modular monolith: domain partitioning without distribution
  - Enforcing module boundaries so the monolith stays modular
  - Lab: restructure the case-study system as a modular monolith and rate it on your scorecard
- Microkernel and Pipeline
  - Microkernel: a stable core with plug-ins, from IDEs to claims processing
  - Pipeline: composing work from filters and pipes
  - Recognizing when a specialized style beats a general one
  - Lab: identify which parts of the case-study system fit a microkernel or pipeline shape, and why

Day two: the distributed family and the choice

- Event-Driven Architecture
  - Brokers and mediators: two topologies with different guarantees
  - Asynchronous thinking: eventual consistency, ordering, and error handling
  - Where event-driven shines, and how it becomes untraceable spaghetti
  - Lab: design an event-driven flow for one workflow of the case study, including its failure paths
- Service-Based and Microservices
  - Service-based architecture: coarse services, shared database, honest middle ground
  - Microservices: independent deployment and data, at full distribution cost
  - The fallacies of distributed computing applied to both
  - Lab: rate service-based versus microservices for the case study and defend the winner
- Choosing, Combining, and Evolving
  - Matching style to problem: the scorecard as a decision tool
  - Hybrids: styles that combine well and combinations that fight
  - Evolving between styles without a rewrite
  - Lab: make the final style recommendation for the case study, record it as an ADR, and defend it in review

Extended Version

The three-day version keeps the same gradient and adds depth and a capstone:

- Space-based architecture and orchestration-driven SOA: the remaining distributed styles and what they teach
- Deeper event-driven practice: designing for replay, idempotency, and observability
- Style archaeology: reading an existing system's real style and its drift from the intended one
- A capstone architecture kata: teams pick and defend a style for a fresh brief against a challenge panel