ENGINEER-TO-ARCHITECT AND DURABLE THINKING SKILLS

Fundamentals of Software Architecture

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

Overview

Software architecture is the set of structural decisions that are expensive to change: how a system is partitioned, which qualities it must guarantee, and how its parts communicate. What makes it hard is that there are no right answers, only tradeoffs, and the consequences of a structural decision often stay hidden until the system is under load, under change, or under attack. The first law of software architecture, in the words of Ford and Richards, is that everything is a tradeoff.

This is a hands-on, practitioner course. It builds the discipline from its foundations up: first what architecture actually is and the laws that govern it, then the architecture characteristics that drive every structural decision, then modularity and components as the raw material of structure, then the major architecture styles, and finally how to make and record sound decisions. We deliberately go deep on the reasoning that transfers to any system rather than surveying every style and tool; where this course teaches the discipline, *From Developer to Architect* covers the career and leadership side of the role. Every module ends with a lab, and each module builds on the one before.

Who Should Attend

- Developers who want a rigorous grounding in software architecture
- Senior engineers and technical leads whose design decisions now shape whole systems
- New architects who learned the role by accident and want to fill in the foundations

Prerequisites

- Solid professional software development experience
- Experience working on at least one system big enough to have structural problems
- Comfort discussing designs with diagrams

What You Will Learn

- Explain what software architecture is and apply the laws of architecture, starting with "everything is a tradeoff"
- Identify and prioritize architecture characteristics from business requirements
- Analyze modularity using coupling, cohesion, and component granularity
- Compare the major architecture styles and match a style to a problem
- Make structural decisions through explicit tradeoff analysis and record them as ADRs
- Diagram and present an architecture so others can understand and challenge it

Course Outline

Day one: the raw material of architecture

- What Architecture Is
  - Structure, characteristics, decisions, and design principles: a working definition

  - The laws of software architecture: everything is a tradeoff, and why over how
  - Architecture versus design, and why the line matters less than the thinking
  - Lab: analyze a case-study system and separate its architectural decisions from its design decisions
- Architecture Characteristics
  - The "-ilities": deriving characteristics from business and technical requirements
  - Explicit versus implicit characteristics, and why you cannot have them all
  - Prioritizing and measuring: from vague quality goals to testable statements
  - Lab: extract and rank the top architecture characteristics from a case-study business brief
- Modularity and Components
  - Coupling, cohesion, and connascence as instruments for measuring structure
  - Identifying components and choosing granularity
  - Partitioning: technical versus domain, and what each optimizes for
  - Lab: partition the case-study system into components two different ways and compare the coupling

Day two: styles and decisions

- Architecture Styles: the Monolithic Family
  - Layered architecture: the default, and what it quietly costs
  - Modular monolith and microkernel
  - Rating styles against architecture characteristics
  - Lab: choose and defend a monolithic style for the case-study system using a characteristics scorecard
- Architecture Styles: the Distributed Family
  - Service-based, event-driven, and microservices in overview
  - The fallacies of distributed computing and what distribution really costs
  - When distribution is justified, and when it is fashion
  - Lab: decide whether the case-study system justifies a distributed style, and defend the answer either way
- Making and Recording Decisions
  - Tradeoff analysis in practice: comparing options against prioritized characteristics
  - Architecture Decision Records: capturing context, decision, and consequences
  - Diagramming with the C4 model, and presenting an architecture for challenge
  - Lab: write an ADR for your style decision, diagram the architecture, and defend both in a review

Extended Version

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

- A deeper pass through the distributed styles, including sagas and eventual consistency at a design level
- Architecture fitness functions: keeping characteristics true over time
- More practice deriving characteristics from messy, realistic requirements
- A capstone architecture kata: teams design a system from a fresh brief, produce ADRs and diagrams, and defend the design in review