Table of Contents

Copyright Information
Disclaimer
Table of Contents

Part 1: What Is Software Architecture?

The Definition of Software Architecture
    Software Engineering vs. Software Architecture
Architectural Characteristics
        Auditability
        Observability
        Scalability
        Responsiveness
        Fault Tolerance
        Extensibility
        Testability
        Performance
        Configurability
Non-Functional Requirements (NFR)
Architectural Tradeoffs
    Interoperability vs Scalability, Elasticity and Responsiveness
    Simplicity and Supportability vs Responsiveness
    Maintainability vs Resilience and Fault Tolerance
    Finding the Balance
    Being Conscious and Aware of Architectural Trade-offs
Long-Term Technical Decisions
Aligning Business With Technology
    Technology & Business: Core Principles
        Principle 1 - Technology Serves Business
        Principle 2 - Business Does Not Understand Technology
        Principle 3 - Technology Powers Business
There is ALWAYS an "Architecture"
Part 1: Key Takeaways

Part 2: Who Is a Software Architect?

Why Do You Need Software Architects?
Who is a Software Architect?
Types of Software Architect Roles

- Software Architect "Specializations" or "Sub-Roles"
  - Solutions Architect
  - Enterprise Architect
  - Technical Architect
  - Cloud Architect
  - Integration Architect
  - Infrastructure Architect
  - Business Architect
  - Application Architect
  - Data Architect
  - Network Architect
  - Domain Architect
  - Security Architect
- Roles Summary

Difference Between Staff/Principal Software Engineers and Software Architects
- Staff Software Engineer/Developer
- Distinguished or Principal Software Engineer/Developer
- Software Architect

The Skills of the Software Architect
- Negotiation 🤝
- Learning 📚
- Influencing 📣
- Communication and Socialization 💬
- Prioritization 🔼
- Facilitation 👋
- Persuasion and "Selling Ideas" 💰
- Coaching/Mentoring 👩‍🏫
- Problem-Solving 🎯
- Translating Business-Speak to Technology-Speak 💻
- Systems Thinking 🌐
- Critical and Creative Thinking 🤔
- Zooming In and Zooming Out 🔎
- Summary of Skills

The Path from a Software Developer to an Architect
- You Think of Edge Cases
- You Ask Whether the System Will "Scale"
- You Treat "One-Size-Fits-All" Advice With Skepticism
- You Question Existing Narratives
- You Frequently "Zoom In" and "Zoom Out"
- You Question Whether Current Technology Will Satisfy Current or Future Business Needs

    - You Create Channels of Communication Between Business And Technology
- Steps for Positioning Yourself as a Software Architect
- Part 2: Key Takeaways

Part 3: Blueprint for Success

- 💎 Principles of Software Architecture
    - Modularity
    - Loose Coupling/Decoupling
    - Separation of Concerns
    - Decoupling Using Asynchronous Flows
    - Favour Stateless over Stateful
- ⚙️ Technical Topics to Master (Or, At Least, Become Very Familiar With)
    - Event-Driven Architecture (EDA), Event Streaming, and Event Sourcing
    - Request Management
        - Throttling
        - Rate Limiting
        - Load Shedding
        - Back Pressure
        - Circuit Breaker
    - Distributed Systems and the Eight Fallacies of Distributed Computing
    - Big Data, Data Lakes, Data Lakehouses, ETL/ELT
    - Data Science, Artificial Intelligence, and Machine Learning
    - Message Brokers, Service Buses, and Event Streaming Platforms
    - Domain Driven Design
    - Networking
    - Databases - Relational vs NoSQL
    - Caching
    - Partitioning and Sharding
    - 12-Factor Applications
    - Zero Trust, Shift Left Security, and Security-First
    - APIs and System Integration
    - Cloud
    - Architectural Styles and Patterns
- 🧰 Tools of the Software Architect
    - Architecture Decision Records (ADR)
        - Sample ADR
    - Architectural Diagrams
        - Example - Good vs Not-So-Good Architectural Diagram
    - Non-Functional Requirements (NFR) and Architectural Characteristics Analysis
    - Decision Trees

- Current State VS Target State Analysis
  - Current State vs Target State - Example
- 🪨 Challenges of the Software Architect
  - Balancing Technical and Business Requirements
  - Keeping Current With Developments in the Industry
  - Identifying and Managing Trade-Offs
  - Aligning Teams and Stakeholders
  - Managing Internal Company Politics
  - Dealing with Personal Agendas
  - Bureaucracy and Organizational Red Tape
  - Balancing Different Types of Work and Constant Context Switching
  - Conveying Technical Information to Non-Technical Audiences
- ✔️ More Strategies for Success
- ❌ Software Architect Anti Patterns
  - The "Ivory Tower Architect"
  - The "Seagull Architect"
  - The "Glorified Developer Architect"
  - The "My-Way" Architect
  - One Last Word on Antipatterns
- ✍️ Wrapping Up
  - Top 3 Mistakes Made By Software Architects
    - Mistake #1 - Thinking Too Far Ahead Or Not Far Ahead Enough
    - Mistake #2 - Not Considering The Tradeoffs
    - Mistake #3 - Mixing Project Management With Architecture
  - Top Techniques that Help Software Architects Thrive
    - 🚀 Leading the Way
    - 🚀 Bridging Gaps Between the Technical and the Non-Technical
    - 🚀 Mentoring
    - 🚀 Knowing How To Dive Deep, But Not Too Deep
    - 🚀 Being Constantly Aware of Cost
    - 🚀 Recognizing that Everything is a Trade-Off
    - 🚀 Most Problems are People Problems

Part 3: Key Takeaways

Part 4: Software Architecture in the Age of AI

Preface

AI: A Quick Run-Through

- Artificial Intelligence (AI)
- Data
- Data Engineering
- Analytics and Business Intelligence (BI)

- Statistics
- Data Science
- Machine Learning (ML)
- Predictive AI
- Neural Networks
- Deep Learning (DL)
- Natural Language Processing (NLP)
- Generative AI
- Foundation Models
- Large Language Models (LLMs)
  - Tokenization
- Transformers
  - Context
- Prompting and Prompt Engineering
- Retrieval-Augmented Generation (RAG)
- Fine-Tuning
  - Training vs Inference
- AI Agents and Agentic AI
- AI Models vs AI Systems

Integrating AI Into Your Role

- Generating Architectural Alternatives
  - Task:
  - Validation Phase
- Drafting Architectural Decision Records
- Decision
- Constraints
- Documentation
- Task
  - 1. Context and Background
  - 2. Options Considered
  - 3. Trade-offs
  - 4. Consequences
- Format
  - 1. SRE Perspective
  - 2. Security Architect Perspective
  - 3. Finance Perspective
  - 4. Skeptical Senior Engineer Perspective

Failure Simulation

- Threat Modeling
- Context
- Task:

- Stress Test:
- Failure Simulation:
- Simulating an Architecture Review

AI Tools to Take Your Architecture to the Next Level

- 🧠Brainstorming Architecture
- 📐Architectural Diagramming
- 🎤 Meeting Note Taking & Analysis
- 📊Code Analysis
- 📖 Knowledge Management
- How These Tools Fit Together

Emerging Architectural Patterns

- AI Gateway
- Retrieval Augmented Generation (RAG)
- Prompt Chaining and Orchestration
- Tiered Model Usage
- Agentic Workflows, Protocols, and Tools
- Semantic Caching
- Guardrails and Safety Layers
- Human In The Loop AI Workflows
- Model as a Service

Rethinking Architectural Characteristics & Trade-Offs

- Cost
- Latency & Availability
- Accuracy
- Data Residency
- Privacy
- Observability & Explainability
- Ethics, Risk, and Accountability
- AI Architectural Trade-Offs Summary

The New Socio-Technical Reality

- Dealing with Politics, and False Expectations
- Everyone is an Architect
- Shadow AI And Uncontrolled Tool Usage

Summing it Up: Career Capital in the Age of AI

- AI Literacy, AI Fluency, and AI Expertise
- The Future of Technology Architects in the Era of AI

Parting Thoughts …

Bonus:

- 🚩 Red Flags in AI-Produced Designs
- 🔎 Sense-Check Checklist
- 🧠 Common Hallucinations in AI Architecture

🔎 Sense-Check Checklist
📁 Context Validation
🔎 Sense-Check Checklist
🌐 Non-Functional Requirements Verification
🔎 Sense-Check Checklist
🧩 Dependency and Integration Realism
🔎 Sense-Check Checklist
⚠️ Failure Mode Analysis
🔎 Sense-Check Checklist
⚙️ Operational Feasibility
🔎 Sense-Check Checklist
🪜 Decision Sequencing
🔎 Sense-Check Checklist
🎯 The Core Principle
⚠️ Common AI Architecture Traps: Summary
- ⭕ Over-Distribution
- ⭕ Over-Abstraction
- ⭕ Imaginary Capabilities
- ⭕ Ignored Constraints
- ⭕ Happy-Path Thinking
- ⭕ Premature Optimization
- ⭕ Static End-State Thinking