Enterprise Architecture Comprehensive Curriculum

A Professional Development Program for
Software Engineers Transitioning to Enterprise Architecture

Program Duration: 9–12 Months

Format: Self-Paced with Milestone Assessments

Target Audience: Software Engineers, Technical Leads, Solution Architects

Version 1.0

December 1, 2025

Contents

1 Program Overview

1.1 Executive Summary

This comprehensive curriculum is designed for software engineers and technical professionals seeking to transition into Enterprise Architecture (EA) roles. The program synthesizes theoretical foundations with practical application, drawing from the most influential texts in the field while emphasizing real-world implementation.

Enterprise Architecture is not merely an IT diagramming exercise—it is a strategic business discipline that bridges technology capabilities with organizational objectives. This curriculum prepares practitioners to operate effectively across all organizational levels, from the "engine room" of technical implementation to the "penthouse" of executive strategy.

1.2 Program Objectives

Upon completion of this curriculum, participants will be able to:

Strategic Competencies

- Articulate the business value of Enterprise Architecture to executive stakeholders
- Design operating models that align IT capabilities with business strategy
- Evaluate and select appropriate EA frameworks for organizational contexts
- Create governance structures that enable architectural evolution

Technical Competencies

- Apply enterprise application patterns to system design
- Design integration architectures using proven messaging patterns
- Structure applications for testability, maintainability, and scalability
- Create architectural documentation that serves multiple stakeholder audiences

Organizational Competencies

- Navigate organizational politics and change management
- Design team structures that support target architectures (Conway's Law)
- Communicate effectively with technical and non-technical stakeholders
- Lead architecture review boards and governance processes

1.3 Target Audience

This curriculum is designed for professionals who meet the following criteria:

> **Primary Audience:**

- Software Engineers with 5+ years of development experience
- Technical Leads seeking to expand their architectural perspective
- Solution Architects transitioning to enterprise-level roles
- IT Managers responsible for technology strategy

Prerequisites:

- Proficiency in at least one programming language
- Experience with software development methodologies (Agile, Waterfall)
- Familiarity with database design and data modeling concepts
- Basic understanding of networking and distributed systems
- Experience participating in architectural discussions or reviews

1.4 Curriculum Structure

The curriculum is organized into four progressive phases, each building upon the previous:

Phase	Focus Area	Duration	Modules
I	Foundational & Strategic Enterprise Architecture	8–10 weeks	3
II	The Practicing Architect	6–8 weeks	3
III	Technical Architecture Mastery	10–12 weeks	4
IV	Organizational Architecture & Synthesis	6–8 weeks	3

Table 1: Curriculum Phase Overview

2 Phase I: Foundational & Strategic Enterprise Architecture

▷ Duration: 8–10 Weeks Level: Foundational to Intermediate

2.1 Phase Overview

Phase I establishes the conceptual foundations of Enterprise Architecture, positioning it firmly as a business discipline rather than a purely technical one. Participants will learn to think strategically about IT capabilities and their alignment with organizational objectives.

Phase I Learning Themes:

- EA as a strategic business enabler
- Operating models and their implications for IT
- Comprehensive EA frameworks and methodologies
- Business model alignment and value stream thinking

2.2 Module 1.1: Enterprise Architecture as Strategy

▷ Duration: 3 Weeks Prerequisites: None

2.2.1 Required Reading

Enterprise Architecture as Strategy: Creating a Foundation for Business Execution by Jeanne W. Ross, Peter Weill, David C. Robertson (MIT CISR)

2.2.2 Module Description

This foundational module introduces Enterprise Architecture as a strategic capability for organizational success. Originating from MIT's Center for Information Systems Research (CISR), the concepts presented here form the vocabulary and mental models that enable architects to communicate effectively with executive leadership.

The module emphasizes that EA is fundamentally about creating a "foundation for business execution"—the stable, high-quality technology platform that enables rapid business innovation while maintaining operational excellence.

2.2.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Define and distinguish operating models:** Articulate the four operating model archetypes (Diversification, Coordination, Replication, Unification) and their implications for IT investment decisions.
2. **Assess organizational maturity:** Evaluate an organization's position on the EA maturity spectrum and identify appropriate next steps for advancement.
3. **Create core diagrams:** Develop the four core EA diagrams (Business Process Integration, Shared Data, Standard Technology, Shared Services) appropriate to an operating model.

4. **Calculate IT engagement:** Apply the concepts of IT engagement and the relationship between foundation for execution and business agility.

5. **Communicate with executives:** Present EA concepts using business-appropriate language that resonates with C-level stakeholders.

2.2.4 Key Concepts and Topics

Concept	Description
Operating Model	The necessary level of business process integration and standardization for delivering goods and services to customers. The four types are Diversification, Coordination, Replication, and Unification.
Foundation for Execution	The IT infrastructure, applications, and data that enable a company's core business operations. A well-designed foundation increases agility while reducing costs.
Core Diagrams	Visual representations of the operating model: the high-level depiction of processes, data, technologies, and customer touchpoints that define how the organization operates.
Enterprise Architecture Maturity	The four stages of EA maturity: Business Silos, Standardized Technology, Optimized Core, and Business Modularity.
IT Engagement Model	The system of governance mechanisms that ensure IT investments align with enterprise architecture principles and business objectives.
Business Modularity	The ability to rapidly reconfigure business capabilities by mixing and matching standardized processes, data, and technology components.

Table 2: Key Concepts: Enterprise Architecture as Strategy

2.2.5 Practical Exercises

Exercise 1.1.1: Operating Model Assessment

Select an organization you are familiar with (your employer, a case study, or a well-documented company). Conduct an operating model assessment by:

1. Identifying the primary customer segments and value propositions
2. Mapping the degree of process standardization across business units
3. Assessing the level of data and process integration required
4. Classifying the organization into one of the four operating model archetypes
5. Documenting three implications of this classification for IT investment priorities

Deliverable: 3–5 page assessment document with supporting diagrams

Exercise 1.1.2: Core Diagram Development

Using the organization from Exercise 1.1.1, create a complete set of core diagrams:

1. Business Process Integration diagram showing key processes and their relationships
2. Shared Data diagram identifying master data entities and ownership
3. Standard Technology diagram depicting the technology stack
4. Customer/Supplier touchpoint diagram

Deliverable: Four diagrams with accompanying narrative (2 pages)

Exercise 1.1.3: Executive Presentation

Prepare a 15-minute executive presentation explaining:

1. The current operating model and its business implications
2. The maturity stage of the organization's EA
3. Recommended next steps for EA advancement
4. Expected business value from EA investments

Deliverable: Slide deck (10–12 slides) with speaker notes

2.2.6 Assessment Criteria

Assessment Component	Weight
Operating Model Assessment	30%
Core Diagram Development	35%
Executive Presentation	25%
Participation in Discussion Forums	10%

Table 3: Module 1.1 Assessment Weights

2.3 Module 1.2: Business Model Alignment

▷ Duration: 2 Weeks Prerequisites: Module 1.1

2.3.1 Required Reading

Business Model Generation by Alexander Osterwalder & Yves Pigneur

2.3.2 Module Description

Before architecting enterprise systems, architects must deeply understand how the business creates, delivers, and captures value. This module introduces the Business Model Canvas as a tool for aligning architectural decisions with business strategy.

The Business Model Canvas provides a shared language between business and technology stakeholders, enabling architects to ensure their designs support the organization's value creation mechanisms.

2.3.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Apply the Business Model Canvas:** Use all nine building blocks to map an organization's business model.
2. **Identify architectural implications:** Derive technology requirements from business model elements.
3. **Facilitate business model workshops:** Lead stakeholder sessions using the canvas methodology.
4. **Assess business model evolution:** Evaluate how digital transformation affects business model components.
5. **Connect value streams to capabilities:** Map value propositions to the IT capabilities required to deliver them.

2.3.4 Key Concepts and Topics

Concept	Description
Customer Segments	The different groups of people or organizations an enterprise aims to reach and serve. Different segments may require different value propositions and channels.
Value Propositions	The bundle of products and services that create value for a specific customer segment. This is the reason customers choose one company over another.
Channels	How a company communicates with and reaches its customer segments to deliver a value proposition.
Customer Relationships	The types of relationships a company establishes with specific customer segments, from personal assistance to self-service to automated services.
Revenue Streams	The cash a company generates from each customer segment (costs must be subtracted from revenues to create earnings).
Key Resources	The most important assets required to make a business model work. May be physical, financial, intellectual, or human.
Key Activities	The most important things a company must do to make its business model work.
Key Partnerships	The network of suppliers and partners that make the business model work.
Cost Structure	All costs incurred to operate a business model.

Table 4: Key Concepts: Business Model Canvas

2.3.5 Practical Exercises

Exercise 1.2.1: Business Model Canvas Mapping

Create a complete Business Model Canvas for an organization, addressing:

1. All nine building blocks with detailed descriptions
2. Relationships and dependencies between blocks
3. Key differentiators from competitors
4. Digital enablers for each building block

Deliverable: Completed canvas with narrative explanation (3 pages)

Exercise 1.2.2: Capability-to-Value Mapping

For the business model from Exercise 1.2.1:

1. Identify the top 10 IT capabilities required to deliver the value propositions
2. Map each capability to the business model building blocks it supports
3. Prioritize capabilities based on strategic importance
4. Identify capability gaps and investment opportunities

Deliverable: Capability map with prioritization matrix

2.4 Module 1.3: Comprehensive EA Frameworks

▷ Duration: 3–4 Weeks Prerequisites: Modules 1.1, 1.2

2.4.1 Required Reading

An Introduction to Enterprise Architecture (EA^3) by Scott A. Bernard

2.4.2 Module Description

This module provides a systematic treatment of Enterprise Architecture as a discipline, introducing the EA^3 Cube Framework. Unlike the strategic overview in Module 1.1, this module dives into the operational aspects of building and maintaining an EA program.

The EA^3 framework connects strategy to implementation through multiple architectural layers: Strategy, Business, Data, Systems, and Technology—with cross-cutting threads including Security, Standards, and Governance.

2.4.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Navigate the EA^3 Cube:** Explain all dimensions of the framework and their interrelationships.
2. **Develop EA artifacts:** Create documentation artifacts appropriate to each layer of the framework.

3. **Establish EA governance:** Design governance structures for architectural decision-making and compliance.
4. **Manage EA repositories:** Understand the requirements for EA tool selection and repository management.
5. **Compare EA frameworks:** Contrast EA^3 with other frameworks (TOGAF, Zachman, FEAF) and select appropriate approaches.

2.4.4 Key Concepts and Topics

Concept	Description
EA^3 Cube Framework	A three-dimensional framework organizing EA by documentation level (strategic, tactical, operational), component type (goals, initiatives, products, services, data, systems, infrastructure), and cross-cutting threads.
Strategic Layer	Goals, objectives, strategies, and initiatives that drive the organization. Connects EA to business planning.
Business Layer	Processes, activities, products, and services that deliver value. Includes organizational structure and roles.
Data Layer	Information entities, relationships, and data management practices. Master data and data governance.
Systems Layer	Applications, interfaces, and system configurations that automate business processes and manage data.
Technology Layer	Networks, platforms, infrastructure, and security controls that host systems and enable connectivity.
EA Governance	The processes, organizational structures, and policies that ensure EA alignment and compliance.
EA Repository	The tool or system used to store, manage, and analyze EA artifacts and their relationships.
Reference Models	Standard models that provide common vocabulary and structure for EA components (e.g., BRM, DRM, SRM, TRM).

Table 5: Key Concepts: EA^3 Framework

2.4.5 Practical Exercises

Exercise 1.3.1: Multi-Layer EA Documentation

Develop EA artifacts across all five layers for a defined scope:

1. Strategic: Goals hierarchy and strategic initiative roadmap
2. Business: Process model (Level 1–2) and capability map
3. Data: Conceptual data model and data dictionary excerpt
4. Systems: Application portfolio diagram and integration map
5. Technology: Infrastructure architecture and network diagram

Deliverable: Complete artifact set with traceability matrix

Exercise 1.3.2: EA Governance Design

Design an EA governance framework including:

1. Architecture Review Board charter and membership
2. Decision rights matrix (who decides what)
3. Architectural principles (8–10 principles with rationale)
4. Compliance assessment process
5. Exception handling procedures

Deliverable: Governance framework document (8–10 pages)

Exercise 1.3.3: Framework Comparison

Prepare a comparative analysis of EA frameworks:

1. Compare EA^3, TOGAF, Zachman, and FEAF
2. Identify strengths and weaknesses of each
3. Recommend which framework(s) are appropriate for different organizational contexts
4. Propose a hybrid approach if applicable

Deliverable: Comparison matrix and recommendation report (5–7 pages)

2.4.6 Phase I Capstone Project

Phase I Capstone: Strategic EA Assessment

Conduct a comprehensive EA assessment for a real or simulated organization:

1. Complete Business Model Canvas analysis
2. Determine operating model classification with justification
3. Assess current EA maturity level
4. Develop core diagrams aligned with operating model
5. Create EA artifact samples for each layer
6. Design governance framework
7. Produce executive roadmap for EA program advancement

Deliverable: Comprehensive EA Assessment Report (25–30 pages) with presentation

3 Phase II: The Practicing Architect

▷ Duration: 6–8 Weeks Level: Intermediate

3.1 Phase Overview

Phase II focuses on the human elements of Enterprise Architecture: how architects operate within organizations, navigate political landscapes, drive change, and communicate effectively with diverse stakeholders. This phase acknowledges that technical excellence alone is insufficient—successful architects must also be skilled organizational operators.

Phase II Learning Themes:

- The architect's role across organizational levels
- Communication and influence strategies
- Pragmatic EA practices that work in reality
- Managing legacy systems and technical debt

3.2 Module 2.1: The Software Architect Elevator

▷ Duration: 2–3 Weeks Prerequisites: Phase I

3.2.1 Required Reading

The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise by Gregor Hohpe

3.2.2 Module Description

Gregor Hohpe's influential work redefines the architect's role for the modern enterprise. The central metaphor—the "elevator"—captures the architect's need to move fluidly between the "penthouse" (executive strategy) and the "engine room" (technical implementation).

This module develops the soft skills and mental models essential for architectural effectiveness in complex organizational environments.

3.2.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Navigate organizational levels:** Communicate effectively with stakeholders from developers to C-suite executives.
2. **Drive architectural change:** Apply influence techniques to advance architectural initiatives without formal authority.
3. **Manage transformation:** Guide organizations through digital transformation journeys.
4. **Balance trade-offs:** Make and communicate architectural decisions involving competing concerns.

5. **Build architectural credibility:** Establish trust and influence through demonstrated expertise and relationship building.

3.2.4 Key Concepts and Topics

Concept	Description
The Elevator	The architect's ability to move between organizational levels, translating concerns and building bridges between business and technology.
Penthouse vs. Engine Room	The contrast between executive-level strategic thinking and hands-on technical work; architects must operate in both.
Selling Options	Positioning architectural investments as options that reduce future risk and enable future capabilities, rather than as fixed costs.
Architecture as a Lingua Franca	Using architecture as a common language to bridge organizational silos and enable cross-functional communication.
Transformation Through Architecture	Using architectural initiatives to drive organizational change and modernization.
The Architect's Toolbox	Mental models, communication techniques, and influence strategies for effective architectural practice.
Standing on Principles	Developing and consistently applying architectural principles to guide decision-making.
Feedback Loops	Establishing mechanisms to learn from architectural decisions and continuously improve.

Table 6: Key Concepts: The Software Architect Elevator

3.2.5 Practical Exercises

Exercise 2.1.1: Elevator Pitch Development

Create three versions of the same architectural initiative:

1. Executive summary for C-suite (1 paragraph, business outcomes focus)
2. Technical brief for development teams (1 page, implementation focus)
3. Business case for IT management (2 pages, cost/benefit analysis)

Deliverable: Three communication artifacts with reflection on translation challenges

Exercise 2.1.2: Influence Mapping

For a proposed architectural change:

1. Identify all stakeholders affected by the change
2. Map their interests, concerns, and influence
3. Develop an influence strategy for each stakeholder group

4. Plan engagement activities and communication cadence

Deliverable: Stakeholder map and influence strategy document

Exercise 2.1.3: Options Analysis

Present an architectural investment as an "option":

1. Define the current state and desired future capabilities
2. Identify the "option" being purchased (flexibility, scalability, etc.)
3. Quantify the cost of the option and the value it enables
4. Present risks of not purchasing the option

Deliverable: Options analysis presentation

3.3 Module 2.2: Pragmatic EA Practice

▷ Duration: 2–3 Weeks Prerequisites: Module 2.1

3.3.1 Required Reading

The Practice of Enterprise Architecture: A Modern Approach to Business and IT Alignment by Svyatoslav Kotusev

3.3.2 Module Description

This module provides a research-based, pragmatic perspective on EA practice. Kotusev's work challenges traditional EA orthodoxy by examining what actually works in real organizations, rather than what frameworks prescribe.

The module develops critical thinking about EA methodologies and helps participants distinguish between effective practices and EA "theater."

3.3.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Evaluate EA frameworks critically:** Distinguish between prescriptive frameworks and effective practice.
2. **Select appropriate EA artifacts:** Choose documentation types based on organizational needs and stakeholder requirements.
3. **Avoid EA anti-patterns:** Recognize and avoid common EA program pitfalls.
4. **Demonstrate EA value:** Measure and communicate the business impact of EA activities.
5. **Scale EA practices:** Adapt EA approaches to organizational size, maturity, and culture.

3.3.4 Key Concepts and Topics

Concept	Description
EA Artifacts Taxonomy	Classification of EA documentation types by purpose, audience, and use case. Not all artifacts are needed in all organizations.
EA Anti-Patterns	Common failures in EA practice: ivory tower architecture, analysis paralysis, over-documentation, disconnection from delivery.
Lightweight EA	Approaches to EA that emphasize value delivery over documentation completeness.
EA Effectiveness Metrics	Measures of EA program success beyond artifact production: decision quality, project success, business agility.
Stakeholder-Driven EA	Orienting EA activities around stakeholder needs rather than framework prescriptions.
EA Maturity Realism	Pragmatic assessment of EA maturity that acknowledges organizational constraints and culture.

Table 7: Key Concepts: Pragmatic EA Practice

3.3.5 Practical Exercises

Exercise 2.2.1: EA Artifact Audit

Conduct an audit of existing EA artifacts in an organization:

1. Inventory all EA documentation currently produced
2. Assess usage and value of each artifact type
3. Identify artifacts that are produced but not used
4. Recommend artifacts to eliminate, combine, or add

Deliverable: Artifact audit report with recommendations

Exercise 2.2.2: Anti-Pattern Analysis

Document three EA anti-patterns you have observed or studied:

1. Describe each anti-pattern in detail
2. Explain root causes and contributing factors
3. Propose remediation strategies
4. Develop prevention mechanisms

Deliverable: Anti-pattern catalog with remediation playbook

3.4 Module 2.3: Enterprise-Scale EA Implementation

▷ Duration: 2 Weeks Prerequisites: Module 2.2

3.4.1 Required Reading

A Practical Guide to Enterprise Architecture by James McGovern et al.

3.4.2 Module Description

This module addresses the unique challenges of EA in large, complex organizations with significant legacy system portfolios. It provides practical guidance on navigating organizational politics, managing technical debt, and executing large-scale transformation programs.

3.4.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Manage legacy systems:** Develop strategies for modernizing and integrating legacy applications.
2. **Navigate organizational politics:** Work effectively within complex political environments.
3. **Execute transition planning:** Create roadmaps for moving from current to target state architecture.
4. **Scale EA programs:** Organize EA functions for large, distributed organizations.
5. **Handle organizational resistance:** Overcome barriers to architectural change.

3.4.4 Practical Exercises

Exercise 2.3.1: Legacy Modernization Strategy

Develop a modernization strategy for a legacy system portfolio:

1. Assess the current legacy landscape (application portfolio analysis)
2. Apply the 6 Rs framework (Retain, Retire, Rehost, Replatform, Refactor, Replace)
3. Develop prioritized modernization roadmap
4. Create risk mitigation plan

Deliverable: Modernization strategy document with roadmap

Exercise 2.3.2: Political Navigation Plan

For a significant architectural initiative, develop a political navigation strategy:

1. Map organizational power structures and decision-makers
2. Identify potential allies and resistors
3. Develop coalition-building strategy
4. Plan for addressing resistance

Deliverable: Political strategy document

4 Phase III: Technical Architecture Mastery

▷ Duration: 10–12 Weeks Level: Advanced

4.1 Phase Overview

Phase III deepens technical architecture skills, translating high-level EA concepts into concrete system designs. This phase bridges the gap between enterprise strategy and implementation, equipping participants with patterns and principles for building robust, maintainable systems.

Phase III Learning Themes:

- Enterprise application design patterns
- Clean architecture principles and practices
- System integration and messaging patterns
- Practical pattern application

4.2 Module 3.1: Patterns of Enterprise Application Architecture

▷ Duration: 3–4 Weeks Prerequisites: Phase II

4.2.1 Required Reading

Patterns of Enterprise Application Architecture by Martin Fowler

4.2.2 Module Description

Martin Fowler's catalog of enterprise application patterns provides a vocabulary and toolkit for designing robust systems. This module covers architectural patterns for structuring applications, managing data, and handling the complexity inherent in enterprise systems.

4.2.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Apply layered architecture patterns:** Design applications using appropriate layering strategies.
2. **Select data source patterns:** Choose between Table Data Gateway, Row Data Gateway, Active Record, and Data Mapper.
3. **Implement domain logic patterns:** Apply Transaction Script, Domain Model, and Table Module patterns appropriately.
4. **Design for object-relational mapping:** Handle the impedance mismatch between objects and relational databases.
5. **Manage concurrency and sessions:** Apply patterns for handling concurrent access and web session state.

4.2.4 Key Concepts and Topics

Pattern Category	Key Patterns
Domain Logic	Transaction Script, Domain Model, Table Module, Service Layer
Data Source Architectural	Table Data Gateway, Row Data Gateway, Active Record, Data Mapper
Object-Relational Behavioral	Unit of Work, Identity Map, Lazy Load
Object-Relational Structural	Identity Field, Foreign Key Mapping, Association Table Mapping, Inheritance Mappers
Object-Relational Metadata	Metadata Mapping, Query Object, Repository
Web Presentation	Model View Controller, Page Controller, Front Controller, Template View, Transform View, Application Controller
Distribution	Remote Facade, Data Transfer Object
Offline Concurrency	Optimistic Offline Lock, Pessimistic Offline Lock, Coarse-Grained Lock, Implicit Lock
Session State	Client Session State, Server Session State, Database Session State
Base Patterns	Gateway, Mapper, Layer Supertype, Registry, Value Object, Money, Plugin, Separated Interface, Service Stub

Table 8: Key Patterns: Enterprise Application Architecture

4.2.5 Practical Exercises

Exercise 3.1.1: Pattern Selection Analysis

Given a set of application requirements, analyze and recommend appropriate patterns:

1. Define the application context and requirements
2. Evaluate applicable patterns for domain logic, data access, and presentation
3. Document trade-offs between pattern choices
4. Recommend a coherent pattern combination

Deliverable: Pattern selection rationale document

Exercise 3.1.2: Pattern Implementation

Implement core patterns in code:

1. Implement Repository pattern with Unit of Work
2. Create a Service Layer with Transaction Script
3. Implement Identity Map for caching
4. Demonstrate Lazy Loading behavior

Deliverable: Working code with documentation

Exercise 3.1.3: Pattern Refactoring

Refactor an existing application to apply enterprise patterns:

1. Analyze current architecture and identify weaknesses
2. Propose pattern-based improvements
3. Execute refactoring in incremental steps
4. Document improvements in maintainability and testability

Deliverable: Refactoring plan and implementation with before/after analysis

4.3 Module 3.2: Clean Architecture

▷ Duration: 2–3 Weeks Prerequisites: Module 3.1

4.3.1 Required Reading

Clean Architecture: A Craftsman's Guide to Software Structure and Design by Robert C. Martin

4.3.2 Module Description

Robert Martin's Clean Architecture provides principles for creating systems that are testable, independent of frameworks, and capable of long-term evolution. This module focuses on the dependency rule and the separation of concerns across architectural boundaries.

4.3.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Apply the Dependency Rule:** Design systems where source code dependencies point inward toward higher-level policies.
2. **Define architectural boundaries:** Identify and enforce boundaries between components at different abstraction levels.
3. **Separate concerns by layer:** Organize code into Entities, Use Cases, Interface Adapters, and Frameworks/Drivers.
4. **Design for testability:** Create architectures that support comprehensive automated testing.
5. **Manage framework dependencies:** Keep frameworks at arm's length to enable future technology changes.

4.3.4 Key Concepts and Topics

Concept	Description
The Dependency Rule	Source code dependencies must point inward, toward higher-level policies. Nothing in an inner circle can know anything about something in an outer circle.
Entities	Enterprise-wide business rules and data. The most stable and least likely to change.
Use Cases	Application-specific business rules that orchestrate data flow to and from entities.
Interface Adapters	Presenters, Controllers, and Gateways that convert data between use cases and external agencies.
Frameworks and Drivers	The outermost layer containing frameworks, tools, databases, and the web.
Screaming Architecture	The architecture should "scream" its purpose—a health care system should look like a health care system, not a Rails or Spring application.
Plugin Architecture	External tools and frameworks should plug into the business logic, not the other way around.
Humble Objects	Separating logic from infrastructure to enable testing.

Table 9: Key Concepts: Clean Architecture

4.3.5 Practical Exercises

Exercise 3.2.1: Clean Architecture Design

Design a system following Clean Architecture principles:

1. Define the domain entities and business rules
2. Specify use cases with input/output boundaries
3. Design interface adapters for persistence and presentation
4. Document framework and driver layer decisions
5. Verify the dependency rule is maintained

Deliverable: Clean Architecture design document with dependency diagrams

Exercise 3.2.2: Testability Assessment

Evaluate a codebase for Clean Architecture compliance:

1. Analyze dependency directions across the codebase
2. Identify coupling to frameworks and external systems
3. Assess testability of business logic
4. Recommend refactoring priorities

Deliverable: Code review report with refactoring recommendations

4.4 Module 3.3: Enterprise Integration Patterns

▷ Duration: 3–4 Weeks Prerequisites: Module 3.2

4.4.1 Required Reading

Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions by Gregor Hohpe, Bobby Woolf

4.4.2 Module Description

Enterprise systems rarely exist in isolation. This module provides a comprehensive pattern language for integrating heterogeneous systems through messaging. The patterns apply to ESB implementations, event-driven architectures, and modern microservices messaging.

4.4.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Select integration styles:** Choose between File Transfer, Shared Database, Remote Procedure Invocation, and Messaging.
2. **Design messaging systems:** Apply Message Channel, Message, Pipes and Filters, Message Router, Message Translator, and Message Endpoint patterns.
3. **Implement message routing:** Use Content-Based Router, Message Filter, Splitter, Aggregator, and Resequencer patterns.
4. **Handle message transformation:** Apply Envelope Wrapper, Content Enricher, Content Filter, and Claim Check patterns.
5. **Manage system integration:** Design for reliability, error handling, and system management in messaging solutions.

4.4.4 Key Concepts and Topics

Pattern Category	Key Patterns
Integration Styles	File Transfer, Shared Database, Remote Procedure Invocation, Messaging
Messaging Systems	Message Channel, Message, Pipes and Filters, Message Router, Message Translator, Message Endpoint
Messaging Channels	Point-to-Point Channel, Publish-Subscribe Channel, Datatype Channel, Invalid Message Channel, Dead Letter Channel, Guaranteed Delivery, Channel Adapter, Message Bridge
Message Construction	Command Message, Document Message, Event Message, Request-Reply, Return Address, Correlation Identifier, Message Sequence, Message Expiration

Pattern Category	Key Patterns
Message Routing	Content-Based Router, Message Filter, Dynamic Router, Recipient List, Splitter, Aggregator, Resequencer, Composed Message Processor, Scatter-Gather, Routing Slip, Process Manager, Message Broker
Message Transformation	Envelope Wrapper, Content Enricher, Content Filter, Claim Check, Normalizer, Canonical Data Model
Messaging Endpoints	Messaging Gateway, Messaging Mapper, Transactional Client, Polling Consumer, Event-Driven Consumer, Competing Consumers, Message Dispatcher, Selective Consumer, Durable Subscriber, Idempotent Receiver, Service Activator
System Management	Control Bus, Detour, Wire Tap, Message History, Message Store, Smart Proxy, Test Message, Channel Purger

Table 10: Key Patterns: Enterprise Integration

4.4.5 Practical Exercises

Exercise 3.3.1: Integration Architecture Design

Design an integration solution for a multi-system landscape:

1. Document the systems to be integrated and their interfaces
2. Select appropriate integration style(s)
3. Design the messaging topology (channels, routers, transformers)
4. Specify error handling and system management patterns

Deliverable: Integration architecture document with pattern catalog

Exercise 3.3.2: Saga Pattern Implementation

Design and implement a saga pattern for distributed transactions:

1. Define the business process requiring coordination
2. Design compensating transactions for each step
3. Implement orchestration or choreography approach
4. Handle failure scenarios and rollback

Deliverable: Saga design with implementation code

Exercise 3.3.3: Event-Driven Architecture Design

Create an event-driven architecture for a business domain:

1. Identify domain events and their triggers
2. Design event schemas and versioning strategy

3. Implement publish-subscribe topology
4. Address event ordering, deduplication, and replay

Deliverable: Event-driven architecture specification

4.5 Module 3.4: Technical Architecture Synthesis

▷ Duration: 2 Weeks Prerequisites: Modules 3.1–3.3

4.5.1 Module Description

This synthesis module integrates the patterns and principles from the preceding modules into coherent system architectures. Participants will practice combining application patterns, clean architecture principles, and integration patterns in realistic design scenarios.

4.5.2 Learning Objectives

Upon completing this module, participants will be able to:

1. **Synthesize multiple pattern languages:** Combine application and integration patterns cohesively.
2. **Design complete systems:** Create end-to-end architectures from requirements to deployment.
3. **Document architectures effectively:** Produce architectural documentation for multiple stakeholder audiences.
4. **Review and critique designs:** Evaluate architectural designs against quality attributes.

4.5.3 Phase III Capstone Project

Phase III Capstone: Technical Architecture Design

Design a comprehensive technical architecture for an enterprise system:

1. Requirements analysis and quality attribute prioritization
2. High-level system decomposition following Clean Architecture
3. Detailed component design using enterprise application patterns
4. Integration architecture for external system connectivity
5. Deployment architecture (cloud-native preferred)
6. Architecture decision records (ADRs) for key decisions
7. Architecture review presentation

Deliverable: Complete architecture documentation package (30–40 pages) with presentation

5 Phase IV: Organizational Architecture & Synthesis

▷ Duration: 6–8 Weeks Level: Advanced

5.1 Phase Overview

Phase IV addresses the critical relationship between organizational structure and system architecture. Conway's Law—"organizations design systems that mirror their communication structures"—implies that architects must understand and sometimes influence organizational design to achieve architectural goals.

Phase IV Learning Themes:

- Team topology design and Conway's Law
- Organizational design as architectural tool
- Curriculum synthesis and career development
- Continuous architectural practice

5.2 Module 4.1: Team Topologies

▷ Duration: 3 Weeks Prerequisites: Phase III

5.2.1 Required Reading

Team Topologies: Organizing Business and Technology Teams for Fast Flow by Matthew Skelton, Manuel Pais

5.2.2 Module Description

Team Topologies provides a model for organizing teams to achieve architectural objectives. By understanding stream-aligned, platform, enabling, and complicated-subsystem team types—and their interaction modes—architects can design organizations that support the target architecture.

5.2.3 Learning Objectives

Upon completing this module, participants will be able to:

1. **Apply Conway's Law strategically:** Use organizational design to enable desired system architecture.
2. **Classify team types:** Distinguish between stream-aligned, platform, enabling, and complicated-subsystem teams.
3. **Design team interactions:** Apply collaboration, X-as-a-Service, and facilitating interaction modes appropriately.
4. **Manage cognitive load:** Right-size teams and boundaries to optimize cognitive load.
5. **Evolve team structures:** Plan organizational evolution alongside architectural evolution.

5.2.4 Key Concepts and Topics

Concept	Description
Stream-Aligned Team	A team aligned to a single, valuable stream of work; capable of delivering user or customer value independently. The primary team type.
Platform Team	A team that enables stream-aligned teams to deliver work with substantial autonomy by providing self-service APIs, tools, and services.
Enabling Team	A team that assists other teams in adopting new technologies or practices, then moves on. Acts as a force multiplier.
Complicated-Subsystem Team	A team responsible for a system component requiring deep specialist knowledge that would be impractical for stream-aligned teams to develop.
Collaboration Mode	Close interaction between two teams working together on a common goal. High-bandwidth but cognitively expensive.
X-as-a-Service Mode	One team provides a service with a clear API/contract; consuming teams don't need to know implementation details.
Facilitating Mode	One team helps another team learn or adopt new skills. Temporary relationship.
Cognitive Load	The total amount of mental effort being used in working memory. Teams should not be overloaded.
Thinnest Viable Platform	A platform with just enough capability to accelerate stream-aligned teams, avoiding over-engineering.
Team API	The explicit definition of how other teams can interact with a team: communication channels, work requests, etc.

Table 11: Key Concepts: Team Topologies

5.2.5 Practical Exercises

Exercise 4.1.1: Team Topology Assessment

Assess an organization's current team structure:

1. Map existing teams to the four fundamental team types
2. Identify current interaction modes between teams
3. Analyze cognitive load distribution
4. Identify structural impediments to flow

Deliverable: Team topology assessment report

Exercise 4.1.2: Target Team Topology Design

Design a target team topology to support a target architecture:

1. Define the target architecture and its key boundaries

2. Design team structure aligned with architectural boundaries
3. Specify team types and sizes
4. Define interaction modes between teams
5. Create evolution path from current to target topology

Deliverable: Team topology design document with evolution roadmap

Exercise 4.1.3: Platform Team Charter

Create a charter for a platform team:

1. Define the platform's purpose and scope
2. Specify the thinnest viable platform offering
3. Design the Team API for platform consumers
4. Establish success metrics and feedback mechanisms

Deliverable: Platform team charter document

5.3 Module 4.2: Architectural Leadership

▷ Duration: 2 Weeks Prerequisites: Module 4.1

5.3.1 Module Description

This module integrates lessons from throughout the curriculum, focusing on the leadership dimensions of the architect role. Topics include building and leading architecture functions, mentoring other architects, and establishing architectural communities of practice.

5.3.2 Learning Objectives

Upon completing this module, participants will be able to:

1. **Build architecture functions:** Establish and grow architecture teams and practices.
2. **Mentor architects:** Develop other architects through coaching and guidance.
3. **Lead communities of practice:** Establish and facilitate architectural communities.
4. **Manage architectural debt:** Track, communicate, and systematically address architectural debt.
5. **Drive continuous improvement:** Establish feedback loops for architectural learning.

5.3.3 Practical Exercises

Exercise 4.2.1: Architecture Function Design

Design an architecture function for an organization:

1. Define the architecture function's mandate and scope
2. Specify roles and career paths (Solution Architect, Domain Architect, Enterprise Architect)
3. Design interaction model with delivery teams
4. Establish governance and decision-making processes

Deliverable: Architecture function design document

Exercise 4.2.2: Architectural Debt Register

Create a comprehensive architectural debt management approach:

1. Define architectural debt categories and severity levels
2. Create a debt register template
3. Establish debt identification and recording processes
4. Design debt retirement prioritization method
5. Create executive reporting dashboard

Deliverable: Architectural debt management framework

5.4 Module 4.3: Curriculum Capstone

▷ Duration: 2–3 Weeks Prerequisites: All Previous Modules

5.4.1 Module Description

The curriculum capstone integrates all learning into a comprehensive Enterprise Architecture engagement. Participants will demonstrate mastery across all curriculum domains through a realistic EA program design and execution plan.

5.4.2 Capstone Project

Final Capstone: Enterprise Architecture Program Design

Design a complete EA program for an organization, synthesizing learning from all phases:

Phase I Components:

- Business model and operating model analysis
- EA framework selection and customization
- Core diagram development

Phase II Components:

- Stakeholder engagement strategy
- Communication and influence plan

- Pragmatic artifact selection

Phase III Components:

- Reference architecture design
- Integration architecture specification
- Technical standards and guidelines

Phase IV Components:

- Target team topology design
- Architecture function organization
- Governance and continuous improvement

Synthesis:

- EA roadmap with 3-year horizon
- Business case and value proposition
- Executive presentation

Deliverable: Complete EA program proposal (50–60 pages) with executive presentation and peer review

A Complete Reading List

A.1 Required Texts

1. Ross, J.W., Weill, P., & Robertson, D.C. (2006). *Enterprise Architecture as Strategy: Creating a Foundation for Business Execution*. Harvard Business Review Press.
2. Osterwalder, A. & Pigneur, Y. (2010). *Business Model Generation*. John Wiley & Sons.
3. Bernard, S.A. (2012). *An Introduction to Enterprise Architecture (EA^3)*, 3rd Edition. AuthorHouse.
4. Hohpe, G. (2020). *The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise*. O'Reilly Media.
5. Kotusev, S. (2018). *The Practice of Enterprise Architecture: A Modern Approach to Business and IT Alignment*. SK Publishing.
6. McGovern, J., Ambler, S.W., Stevens, M.E., et al. (2004). *A Practical Guide to Enterprise Architecture*. Prentice Hall.
7. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.
8. Martin, R.C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.
9. Hohpe, G. & Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional.
10. Skelton, M. & Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press.

A.2 Supplementary Texts

1. Richards, M. & Ford, N. (2020). *Fundamentals of Software Architecture*. O'Reilly Media.
2. Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
3. Newman, S. (2021). *Building Microservices*, 2nd Edition. O'Reilly Media.
4. Rozanski, N. & Woods, E. (2011). *Software Systems Architecture*, 2nd Edition. Addison-Wesley.
5. TOGAF Standard, Version 9.2. The Open Group.

B Assessment Framework

B.1 Assessment Methods

Assessment Type	Weight	Description
Module Exercises	30%	Practical exercises completed for each module
Phase Capstones	40%	Major deliverables at the end of each phase
Final Capstone	20%	Comprehensive EA program design
Participation	10%	Discussion forums, peer reviews, presentations

Table 12: Assessment Weight Distribution

B.2 Grading Rubric

Grade	Criteria
Distinguished (90–100%)	Exceptional work demonstrating mastery; innovative application of concepts; professional-quality deliverables
Proficient (80–89%)	Strong understanding demonstrated; complete and well-organized deliverables; minor areas for improvement
Competent (70–79%)	Adequate understanding; deliverables meet requirements but lack depth or polish
Developing (60–69%)	Partial understanding; significant gaps in deliverables; requires additional work
Insufficient (<60%)	Insufficient evidence of learning; incomplete deliverables

Table 13: Grading Criteria

C Program Timeline

Week	Module	Key Activities
1–3	1.1	EA as Strategy reading; Operating model exercises
4–5	1.2	Business Model Canvas; Capability mapping
6–9	1.3	EA^3 Framework; Phase I Capstone
10–12	2.1	Software Architect Elevator; Communication exercises
13–15	2.2	Pragmatic EA Practice; Artifact audit
16–17	2.3	Enterprise-scale EA; Legacy modernization
18–21	3.1	Enterprise Application Patterns; Pattern implementation
22–24	3.2	Clean Architecture; Testability assessment
25–28	3.3	Integration Patterns; Event-driven design
29–30	3.4	Technical Synthesis; Phase III Capstone
31–33	4.1	Team Topologies; Organizational design
34–35	4.2	Architectural Leadership; Function design
36–38	4.3	Final Capstone preparation and presentation

Table 14: 38-Week Program Timeline

D Learning Resources

D.1 Online Communities

- The Architect Elevator Blog (https://architectelevator.com)
- Martin Fowler's Blog (https://martinfowler.com)
- InfoQ Architecture Topics (https://www.infoq.com/architecture-design/)
- EA Principals Community
- Team Topologies Community

D.2 Conferences

- O'Reilly Software Architecture Conference
- SATURN (SEI Architecture Technology User Network)
- QCon
- Gartner Enterprise Architecture Summit

D.3 Certifications (Optional)

- TOGAF 9 Certified
- AWS/Azure/GCP Solutions Architect
- Certified Enterprise Architect (CEA)
- ITIL Foundation