

Enterprise Architecture Framework Mapping

A Comprehensive Crosswalk Between Core Architectural Viewpoints, the Zachman Framework, and DoDAF

A Practical Guide for Enterprise and System Architects

Version 2.0

December 12, 2025

Contents

Executive Summary	3
1 Introduction	4
1.1 Purpose and Scope	4
1.2 Target Audience	4
1.3 Document Structure	4
2 Framework Foundations	5
2.1 Core Architectural Viewpoints	5
2.1.1 Conceptual Basis	5
2.1.2 Standard Core Viewpoints	5
2.2 The Zachman Framework	6
2.2.1 Framework Philosophy	6
2.2.2 The Two-Dimensional Matrix	6
2.2.3 Framework Characteristics	6
2.3 DoDAF (Department of Defense Architecture Framework)	7
2.3.1 Framework Purpose and Evolution	7
2.3.2 Viewpoint Families	7
2.3.3 Model Catalog	7
2.3.4 Data-Centric Approach	8
3 Comprehensive Framework Crosswalk	9
3.1 Understanding the Mapping Approach	9
3.2 The Mapping Table	9
4 Practical Interpretation Guidelines	16
4.1 Reading the Mapping Effectively	16
4.1.1 Zachman Framework Interpretation	16
4.1.2 DoDAF Interpretation	16
4.2 Common Mapping Patterns	16
4.2.1 Cross-Cutting Concerns	16
4.2.2 Viewpoint Progression	17
4.3 Framework Selection Guidance	17
4.3.1 When to Use Core Viewpoints	17
4.3.2 When to Use Zachman	17
4.3.3 When to Use DoDAF	17
5 Alignment with the 4+1 View Model	19
5.1 The 4+1 View Model Overview	19
5.2 Mapping to Core Viewpoints	19
5.3 Extended Viewpoints Beyond 4+1	19
5.4 4+1 Mapping to Zachman and DoDAF	20
6 Extended Examples and Use Cases	21
6.1 Example 1: E-Commerce Platform Architecture	21
6.1.1 Scenario	21
6.1.2 Viewpoint Application	21
6.2 Example 2: Healthcare Information Exchange	22
6.2.1 Scenario	22
6.2.2 Framework Selection and Rationale	22

6.2.3	Key Architectural Artifacts	22
6.2.4	Cross-Framework Translation	23
7	Limitations and Considerations	25
7.1	Mapping Limitations	25
7.1.1	Not Perfect One-to-One Correspondences	25
7.1.2	Context-Dependent Interpretation	25
7.2	Framework Evolution	25
7.3	Practical Application Challenges	25
7.3.1	Organizational Resistance	25
7.3.2	Tool Support Variability	26
7.4	When Not to Use Multiple Frameworks	26
8	Conclusion	27
8.1	Key Takeaways	27
8.2	Recommended Practices	27
8.3	Future Directions	27
8.4	Final Thoughts	28
Appendix A: Quick Reference Guide		29
Appendix B: Glossary		30
Appendix C: References and Further Reading		31

Executive Summary

Enterprise and system architects frequently encounter multiple architectural frameworks, each offering distinct perspectives and vocabularies for describing complex systems. This document provides a comprehensive mapping between three widely-adopted architectural approaches:

- **Core Architectural Viewpoints:** Concern-based slices aligned with ISO/IEC/IEEE 42010 standards
- **Zachman Framework:** A comprehensive taxonomy using stakeholder abstraction levels and interrogative aspects
- **DoDAF (Department of Defense Architecture Framework):** Enterprise-scale viewpoint sets with explicit model catalogs

This mapping enables architects to:

- Translate concepts and artifacts between frameworks
- Understand how different stakeholder concerns are addressed across methodologies
- Leverage existing documentation when transitioning between frameworks
- Ensure comprehensive coverage when using multiple frameworks concurrently

The document includes detailed explanations of each framework's philosophy, a comprehensive crosswalk table, practical interpretation guidelines, and alignment with the foundational 4+1 architectural view model.

1 Introduction

1.1 Purpose and Scope

This document serves as a reference guide for enterprise and system architects working across multiple architectural frameworks. In practice, organizations often adopt different frameworks at different levels of the enterprise, inherit legacy documentation using various methodologies, or need to communicate with stakeholders familiar with different architectural approaches.

The primary objectives are to:

1. Establish clear correspondences between core architectural viewpoints, Zachman Framework dimensions, and DoDAF viewpoint families
2. Explain the conceptual rationale behind each mapping
3. Provide practical guidance for translating architectural artifacts between frameworks
4. Identify areas of overlap, gaps, and unique coverage in each approach

1.2 Target Audience

This guide is intended for:

- Enterprise architects responsible for organization-wide architectural governance
- System architects designing complex technical systems
- Solution architects bridging business and technical domains
- Architecture review board members evaluating architectural documentation
- Technical leads transitioning between organizations with different architectural standards
- Consultants working across multiple client environments

1.3 Document Structure

The document is organized as follows:

Section 2 provides foundational concepts and framework overviews

Section 3 presents the detailed crosswalk mapping table

Section 4 offers practical interpretation guidelines

Section 5 aligns the mapping with the 4+1 view model

Section 6 includes extended examples and use cases

Section 7 discusses limitations and considerations

2 Framework Foundations

2.1 Core Architectural Viewpoints

2.1.1 Conceptual Basis

Core architectural viewpoints represent *concern-based slices* of an architecture description, as defined by ISO/IEC/IEEE 42010:2011 (and its successor ISO/IEC/IEEE 42010:2022). This international standard establishes architecture descriptions as artifacts created to address specific stakeholder concerns through defined viewpoints.

Key Principles:

- *Separation of Concerns*: Each viewpoint addresses a distinct set of architectural concerns
- *Stakeholder Focus*: Viewpoints are selected based on which stakeholders need specific information
- *Consistency Requirements*: Views must maintain consistency with one another where they overlap
- *Completeness*: The set of viewpoints should comprehensively address all significant concerns

2.1.2 Standard Core Viewpoints

While ISO/IEC/IEEE 42010 does not mandate specific viewpoints, the following core set has emerged as a de facto standard through widespread adoption:

Context Defines the system boundary, external entities, and environmental constraints. Addresses questions about system scope, external dependencies, and the relationship between the system and its operating environment.

Logical/Functional Describes the system's functional decomposition and logical organization. Focuses on what the system does, how functionality is organized, and the logical relationships between functional elements.

Process/Concurrency Captures runtime behavior, including concurrency, synchronization, performance characteristics, and scalability. Addresses dynamic aspects of system execution.

Development/Implementation Documents the organization of code, modules, libraries, and build processes. Addresses concerns about maintainability, team organization, and development tooling.

Information/Data Models data structures, information flow, and data lifecycle. Addresses questions about data consistency, persistence, and information architecture.

Operational Describes how the system is used, managed, and supported in its operational environment. Addresses business processes, operational workflows, and support procedures.

Deployment/Physical Shows the physical infrastructure, including hardware nodes, network topology, and runtime platform configuration. Addresses deployment concerns and environmental dependencies.

2.2 The Zachman Framework

2.2.1 Framework Philosophy

The Zachman Framework, developed by John Zachman in the 1980s and refined over subsequent decades, provides a comprehensive *taxonomy* for organizing enterprise architecture artifacts. Unlike prescriptive methodologies, Zachman offers a classification schema ensuring completeness of architectural coverage.

Fundamental Premises:

- Complex entities (enterprises, systems) can be described from multiple perspectives
- Each perspective operates at a different level of abstraction
- Universal interrogatives (What, How, Where, Who, When, Why) apply at every level
- The intersection of perspective and interrogative defines a unique architectural artifact

2.2.2 The Two-Dimensional Matrix

Rows (Perspectives/Abstraction Levels):

1. **Planner Perspective (Scope/Contextual)**: Executive summary level, establishing scope and high-level concepts
2. **Owner Perspective (Business/Conceptual)**: Business model level, defining business entities and processes
3. **Designer Perspective (System/Logical)**: Architect's view, showing logical system design
4. **Builder Perspective (Technology/Physical)**: Engineer's view, detailing physical implementation
5. **Subcontractor Perspective (Detailed Specifications)**: Individual component specifications
6. **Functioning Enterprise**: The actual operating system (as-built, as-operated)

Columns (Interrogatives/Aspects):

1. **What (Data)**: Entities, attributes, data structures
2. **How (Function)**: Processes, activities, transformations
3. **Where (Network)**: Locations, geography, network topology
4. **Who (People)**: Roles, organizations, responsibilities
5. **When (Time)**: Events, schedules, timing, cycles
6. **Why (Motivation)**: Goals, strategies, business rules

2.2.3 Framework Characteristics

- *Completeness*: The 6×6 matrix ensures no architectural concern is overlooked
- *Neutrality*: Framework-agnostic, accommodating any methodology or notation
- *Classification*: Provides a taxonomy rather than a process
- *Independence*: Each cell represents an independent perspective

- *Recursion:* Can be applied at enterprise, system, or component levels

2.3 DoDAF (Department of Defense Architecture Framework)

2.3.1 Framework Purpose and Evolution

DoDAF originated from the need to standardize architectural descriptions across the U.S. Department of Defense, enabling interoperability and integration across complex defense systems and enterprises. The framework has evolved through multiple versions:

- **DoDAF 1.0 (2003):** Initial standardization of architectural views
- **DoDAF 1.5 (2007):** Enhanced viewpoints and model definitions
- **DoDAF 2.0 (2009):** Major restructuring with data-centric approach
- **DoDAF 2.02 (Current):** Refined models and improved guidance

2.3.2 Viewpoint Families

DoDAF organizes architectural models into eight distinct viewpoint families:

AV (All Viewpoint) Provides overarching scope, summary, and information about the architecture description itself. Includes metadata and architecture roadmaps.

CV (Capability Viewpoint) Describes capabilities required to support mission objectives, independent of how they are implemented. Links operational concepts to capabilities.

DIV (Data and Information Viewpoint) Models data structures, information exchange requirements, and data relationships. Ensures semantic interoperability.

OV (Operational Viewpoint) Describes operational concepts, activities, and information exchanges from the operator's perspective. Focuses on mission/business processes.

PV (Project Viewpoint) Documents program/project relationships to capabilities and architectural elements. Tracks implementation timeline and dependencies.

StdV (Standards Viewpoint) Identifies technical and operational standards governing implementation. Ensures compliance and reuse.

SV (Systems Viewpoint) Describes system functions, resources, and interconnections. Focuses on technical system implementation.

SvcV (Services Viewpoint) Models service-oriented architectures, including service functions, resources, and compositions. Emphasizes service reuse and choreography.

2.3.3 Model Catalog

Each viewpoint family contains specific numbered models (e.g., OV-1, SV-4, DIV-2). These models define:

- Specific diagrammatic or tabular formats
- Required and optional elements
- Relationships to other models
- Stakeholder audiences
- Typical use cases

2.3.4 Data-Centric Approach

DoDAF 2.0 introduced a fundamental shift toward data-centricity:

- Models are manifestations of underlying architectural data
- Data is captured in the DoDAF Meta Model (DM2)
- Multiple models can be generated from the same data
- Tool-independence through data standardization
- Enhanced consistency across models

3 Comprehensive Framework Crosswalk

3.1 Understanding the Mapping Approach

The crosswalk table maps core architectural viewpoints to corresponding elements in Zachman and DoDAF. Key considerations:

- **Zachman Mapping:** Each core viewpoint typically spans multiple Zachman cells, as Zachman classifies by stakeholder level and interrogative aspect, while core viewpoints represent integrated concern areas.
- **DoDAF Mapping:** Core viewpoints map to one or more DoDAF viewpoint families and their constituent models. DoDAF's model catalog provides explicit specifications for each architectural artifact.
- **Many-to-Many Relationships:** Mappings are not always one-to-one. A single core viewpoint may involve multiple Zachman interrogatives or DoDAF models, and vice versa.

3.2 The Mapping Table

Table 1: Comprehensive Crosswalk: Core Viewpoints \leftrightarrow Zachman \leftrightarrow DoDAF

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Context	<p>Why, Who, Where (plus <i>What/How</i> at boundary) <i>Rationale:</i> Context establishes motivation (Why), identifies external stakeholders (Who), and defines environmental positioning (Where). Boundary definition touches on What entities and How processes interact at the edge.</p>	<p>Planner, Owner (and <i>Designer</i> for system boundary detail) <i>Rationale:</i> Context operates primarily at executive (Planner) and business (Owner) abstraction levels, with Designer-level detail when specifying precise system boundaries.</p>	<p>AV + CV + high-level OV: AV-1 (Overview & Summary), AV-2 (Integrated Dictionary); CV-1 (Vision), CV-2 (Capability Taxonomy); OV-1 (High-Level Operational Concept) <i>Rationale:</i> AV models provide architectural scope and context; CV models establish capability requirements; OV-1 describes the high-level operational environment.</p>

(Continued on next page)

(Continued from previous page)

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Logical / Functional	<p>How + What (and <i>Who</i> for responsibility allocation) <i>Rationale:</i> Functional viewpoints focus on processes/transformations (How) and the entities they manipulate (What). Role assignments (Who) indicate functional responsibility.</p>	<p>Owner → Designer (and <i>Builder</i> for technology-realized functions) <i>Rationale:</i> Spans from business-level functional decomposition (Owner) through logical design (Designer) to technology-specific function realization (Builder).</p>	<p>OV + SvcV/SV: OV-5a (Operational Activity Decomposition Tree), OV-5b (Operational Activity Model); SvcV-4 (Services Functionality Description), SV-4 (Systems Functionality Description) <i>Rationale:</i> OV-5 models capture operational/business functions; SvcV-4 and SV-4 describe service and system functional decomposition respectively.</p>

(Continued on next page)

(Continued from previous page)

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Process / Concurrency (runtime behavior, scaling, performance)	When + How (often <i>Where</i> for distribution) <i>Rationale:</i> Process viewpoints emphasize timing/sequencing (<i>When</i>) and behavioral dynamics (<i>How</i>). Distributed systems add network topology (<i>Where</i>) considerations.	Designer → Builder (and <i>Functioning Enterprise</i> for "as-operated") <i>Rationale:</i> Process views exist primarily at design and engineering levels, with operational runtime characteristics emerging in the <i>Functioning Enterprise</i> perspective.	OV + SvcV/SV (+ measures): OV-6b (Operational State Transition), OV-6c (Operational Event-Trace); SvcV-10b (Services State Transition), SvcV-10c (Services Event-Trace); SV-10b (Systems State Transition), SV-10c (Systems Event-Trace); SvcV-7 (Services Measures), SV-7 (Systems Measures) <i>Rationale:</i> State transition and event-trace models capture dynamic behavior; measures models document performance characteristics.

(Continued on next page)

(Continued from previous page)

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Development / Implementation (code/module organization, build)	<p>How (implementation) + Who (team/work allocation) <i>Rationale:</i> Development viewpoints focus on implementation approach (How at technical level) and team/organizational structure (Who delivers what).</p>	<p>Builder → Subcontractor <i>Rationale:</i> Operates at the most concrete technical levels, from component design (Builder) to detailed specifications (Subcontractor).</p>	<p>PV + StdV (+ evolution): PV-1 (Project Portfolio Relationships), PV-2 (Project Timelines), PV-3 (Project to Capability Mapping); StdV-1 (Standards Profile), StdV-2 (Standards Forecast); SvcV-8 (Services Evolution), SV-8 (Systems Evolution); SvcV-9 (Services Technology & Skills Forecast), SV-9 (Systems Technology & Skills Forecast) <i>Rationale:</i> PV models track project organization and dependencies; StdV models govern technical standards; evolution models show development roadmaps.</p>

(Continued on next page)

(Continued from previous page)

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Information / Data	What (data) <i>Rationale:</i> Data viewpoints focus exclusively on entities, attributes, and relationships—the "What" interrogative in its purest form.	Owner → Designer → Builder (→ <i>Subcontractor</i> for specs) <i>Rationale:</i> Data architecture spans all abstraction levels: conceptual data models (Owner), logical models (Designer), physical schemas (Builder), to implementation DDL (Subcontractor).	DIV (+ glossary): DIV-1 (Conceptual Data Model), DIV-2 (Logical Data Model), DIV-3 (Physical Data Model); AV-2 (Integrated Dictionary) <i>Rationale:</i> DIV models provide the complete data architecture progression from conceptual through physical; AV-2 ensures consistent terminology.
Operational (business/mission usage, support/monitoring in environment)	How, Who, Where, When, Why <i>Rationale:</i> Operational viewpoints integrate multiple aspects: processes (How), actors (Who), locations (Where), timing (When), and objectives (Why).	Owner + Functioning Enterprise <i>Rationale:</i> Combines business-level operational concepts (Owner) with actual operational reality (Functioning Enterprise as-operated state).	OV (+ capability trace): OV-1 (High-Level Operational Concept), OV-2 (Operational Resource Flow), OV-4 (Organizational Relationships), OV-5a/5b (Operational Activities), OV-6a/6b/6c (Operational Sequences/States/Events); CV-6 (Capability to Operational Activities Mapping) <i>Rationale:</i> OV models comprehensively describe operational concepts, activities, and relationships; CV-6 traces capabilities to operations.

(Continued on next page)

(Continued from previous page)

Core Architectural Viewpoint	Zachman Aspect Emphasis (Columns)	Zachman Perspective Emphasis (Rows)	DoDAF Viewpoint(s) + Representative Models
Deployment / Physical (runtime infrastructure, nodes, networks)	Where + How (and <i>Who</i> for ops roles) <i>Rationale:</i> Deployment emphasizes physical location/topology (Where) and technical implementation approach (How), with operational roles (Who) for system management.	Builder (and <i>Functioning Enterprise</i> for "deployed reality") <i>Rationale:</i> Primarily an engineering perspective (Builder) describing physical implementation, realized in the <i>Functioning Enterprise</i> as actual deployed infrastructure.	SV + SvcV + StdV: SV-1 (Systems Interface Description), SV-2 (Systems Resource Flow); SvcV-1 (Services Context), SvcV-2 (Services Resource Flow); StdV-1 (Standards Profile), StdV-2 (Standards Forecast) <i>Rationale:</i> SV and SvcV models describe system/service deployment and interconnections; StdV ensures technical standards compliance.

4 Practical Interpretation Guidelines

4.1 Reading the Mapping Effectively

4.1.1 Zachman Framework Interpretation

When mapping core viewpoints to Zachman:

1. **Expect Multiple Appearances:** A single core viewpoint typically appears across multiple Zachman cells because:
 - Core viewpoints integrate multiple concerns
 - Zachman classifies by two independent dimensions (perspective and aspect)
 - Different stakeholders need the same concern addressed at different abstraction levels
2. **Emphasis vs. Exclusivity:** Listed Zachman aspects and perspectives represent *primary emphasis*, not exclusive coverage. Secondary aspects often provide supporting context.
3. **Abstraction Level Variation:** The same architectural concern exists at every Zachman perspective level, but with different detail and audience:
 - *Planner:* High-level concepts and scope
 - *Owner:* Business-level requirements and models
 - *Designer:* Logical system design
 - *Builder:* Physical implementation specifications
 - *Subcontractor:* Detailed component specs
 - *Functioning Enterprise:* Actual operating reality

4.1.2 DoDAF Interpretation

When mapping core viewpoints to DoDAF:

1. **Viewpoint Families as Containers:** Each DoDAF viewpoint family (AV, CV, DIV, etc.) contains multiple specific models. The mapping indicates which families are relevant.
2. **Representative Models:** Listed models are *representative examples*, not exhaustive. Consult the complete DoDAF model catalog for all applicable models within a viewpoint family.
3. **Model Relationships:** DoDAF models are interconnected. When using models from one viewpoint, consider dependencies and consistency requirements with models from other viewpoints.
4. **Data-Centric Foundation:** Remember that DoDAF 2.0+ models are views of underlying architectural data. Consistency emerges from maintaining a coherent data model (DM2).

4.2 Common Mapping Patterns

4.2.1 Cross-Cutting Concerns

Some architectural concerns naturally span multiple viewpoints:

Security Appears in Context (security requirements), Logical (security functions), Process (authentication flows), Development (security libraries), Information (access control models), Operational (security procedures), and Deployment (security zones).

Scalability Primarily Process viewpoint, but impacts Logical (scalable design patterns), Deployment (infrastructure capacity), and Development (performance-optimized implementation).

Integration Context (external interfaces), Logical (integration layers), Information (data exchange formats), Deployment (integration infrastructure), and Operational (integration procedures).

Mapping Approach: For cross-cutting concerns, identify the primary viewpoint for each aspect, then add references to supporting viewpoints where necessary.

4.2.2 Viewpoint Progression

Certain viewpoint sequences represent natural refinement progressions:

- **Context → Logical:** From system boundary to internal functional organization
- **Logical → Development:** From logical design to physical code organization
- **Logical → Deployment:** From logical components to deployed physical nodes
- **Information (Conceptual) → Information (Logical) → Information (Physical):** Data architecture refinement

4.3 Framework Selection Guidance

4.3.1 When to Use Core Viewpoints

Best for:

- Software-intensive system architectures
- Agile/iterative development environments
- Projects requiring lightweight documentation
- Communicating with technical stakeholders
- ISO/IEC/IEEE 42010 compliance requirements

4.3.2 When to Use Zachman

Best for:

- Enterprise-wide architecture initiatives
- Ensuring comprehensive architectural coverage
- Multi-stakeholder environments requiring tailored perspectives
- Architecture assessment and gap analysis
- Tool-agnostic architecture documentation

4.3.3 When to Use DoDAF

Best for:

- Large-scale enterprise or system-of-systems architectures
- Government/defense sector projects
- Programs requiring standardized model formats

- Architectures emphasizing operational capabilities
- Data-centric architecture management
- Projects requiring detailed traceability

5 Alignment with the 4+1 View Model

5.1 The 4+1 View Model Overview

Philippe Kruchten's 4+1 View Model, introduced in 1995, represents one of the most influential architectural view models. It defines four primary views plus scenarios:

- Logical View** Describes functional requirements—what the system provides to end users
- Process View** Captures dynamic aspects—concurrency, distribution, performance, scalability
- Development View** Describes static organization of software in the development environment
- Physical View** Depicts the system from a system engineer's perspective—mapping software to hardware
- Scenarios ("+1")** Illustrates the architecture through use cases, validating and driving the other views

5.2 Mapping to Core Viewpoints

The 4+1 model provides a valuable sanity check for the core viewpoint set:

4+1 View	Core Viewpoint Correspondence
Logical View	Logical/Functional viewpoint (functional decomposition and organization)
Process View	Process/Concurrency viewpoint (runtime behavior, synchronization, performance)
Development View	Development/Implementation viewpoint (code organization, modules, build processes)
Physical View	Deployment/Physical viewpoint (hardware nodes, network topology, platform)
Scenarios/Use Cases	Operational viewpoint (provides usage context and validation scenarios)

Table 2: 4+1 View Model Alignment with Core Viewpoints

5.3 Extended Viewpoints Beyond 4+1

The core viewpoint set extends 4+1 with additional concerns:

Context Viewpoint Not explicitly present in 4+1, but critically important for understanding system boundaries, external dependencies, and environmental constraints. The Context viewpoint frames all other views.

Information/Data Viewpoint While data appears in the Logical View of 4+1, modern systems often require a dedicated data architecture view addressing data modeling, data flow, persistence strategies, and information lifecycle management.

This extension reflects the evolution of architectural practice since the mid-1990s:

- Increased emphasis on system boundaries and integration
- Growing importance of data architecture in data-intensive systems
- Recognition that data concerns often crosscut functional boundaries

- Need for explicit data governance in enterprise contexts

5.4 4+1 Mapping to Zachman and DoDAF

4+1 View	Primary Zachman Emphasis	Primary DoDAF Models
Logical View	How + What (Designer level)	OV-5a/5b, SvcV-4, SV-4
Process View	When + How (Designer/Builder)	OV-6b/6c, SvcV-10b/10c, SV-10b/10c
Development View	How + Who (Builder/Subcontractor)	PV-1/2/3, StdV-1/2
Physical View	Where + How (Builder)	SV-1/2, SvcV-1/2
Scenarios	All aspects (Owner + Functioning)	OV-1, CV-6

Table 3: 4+1 Alignment with Zachman and DoDAF

6 Extended Examples and Use Cases

6.1 Example 1: E-Commerce Platform Architecture

6.1.1 Scenario

A mid-sized retailer is building a modern e-commerce platform. The architecture team needs to document the system for multiple stakeholders: executives, business analysts, developers, operations teams, and security auditors.

6.1.2 Viewpoint Application

Context System context diagram showing customer-facing web/mobile interfaces, payment gateway integration, inventory management system integration, shipping carrier APIs, and enterprise data warehouse connections.

Zachman: Planner/Owner levels across Why (business drivers), Who (customers, partners), Where (online channels, fulfillment centers)

DoDAF: AV-1 (architectural overview), OV-1 (operational concept)

Logical/Functional Functional decomposition into: catalog management, search and discovery, shopping cart, order management, payment processing, customer management, inventory management, and recommendation engine.

Zachman: How (business processes), What (business entities), Who (role assignments) at Owner/Designer levels

DoDAF: OV-5b (operational activities), SvcV-4 (service functions)

Process/Concurrency Sequence diagrams for: user registration/authentication, product search with caching, checkout process with transaction management, order fulfillment workflow, and concurrent inventory updates.

Zachman: When (timing), How (dynamic behavior) at Designer/Builder levels

DoDAF: SvcV-10c (service event traces), SV-10c (system event traces)

Development/Implementation Module structure showing: frontend (React), API layer (Node.js), business logic (Java microservices), data access (Spring Data), shared libraries, and CI/CD pipeline configuration.

Zachman: How (implementation), Who (team organization) at Builder level

DoDAF: PV-2 (project timeline), StdV-1 (technology standards)

Information/Data Entity-relationship diagram covering: customers, products, orders, payments, inventory, with data flow diagrams showing: customer data → analytics, order data → fulfillment, inventory updates → catalog.

Zachman: What (data) across Owner (conceptual), Designer (logical), Builder (physical) levels

DoDAF: DIV-1 (conceptual data model), DIV-2 (logical data model), DIV-3 (physical schema)

Operational Business processes: customer journey mapping, support ticket workflows, promotional campaign management, fraud detection procedures, and monitoring/alerting procedures.

Zachman: All aspects at Owner + Functioning Enterprise levels

DoDAF: OV-2 (operational resource flows), OV-4 (organizational relationships)

Deployment/Physical Cloud infrastructure on AWS: load-balanced web tier, containerized microservices on EKS, managed RDS databases, ElastiCache for session/caching, S3 for static assets, CloudFront CDN, and multi-region active-passive disaster recovery.

Zachman: Where (topology), How (infrastructure) at Builder + Functioning Enterprise levels

DoDAF: SV-1 (system interfaces), SV-2 (system resource flows)

6.2 Example 2: Healthcare Information Exchange

6.2.1 Scenario

A regional health information exchange (HIE) enables secure sharing of patient records across hospitals, clinics, laboratories, and pharmacies. The architecture must address complex regulatory requirements (HIPAA), interoperability standards (HL7, FHIR), and diverse stakeholder needs.

6.2.2 Framework Selection and Rationale

Primary Framework: DoDAF

Rationale:

- Large-scale system-of-systems architecture
- Multiple independent organizations
- Stringent regulatory and standards requirements
- Need for capability-based planning (CV models)
- Complex operational workflows (OV models)
- Service-oriented architecture emphasis (SvcV models)

Secondary Framework: Core Viewpoints

Used for internal system development within participating organizations, bridging to DoDAF models as needed.

6.2.3 Key Architectural Artifacts

1. Capability Requirements (DoDAF CV-2, Core Context)

- Patient identity management
- Consent and authorization management
- Clinical data exchange
- Provider directory services
- Audit and compliance reporting

2. Operational Concept (DoDAF OV-1, Core Context + Operational)

- HIE operational model diagram
- Participating organization types and roles

- Information exchange patterns
- Governance structure

3. Operational Activities (DoDAF OV-5b, Core Logical/Functional + Operational)

- Patient record request workflow
- Emergency break-glass access procedure
- Consent revocation process
- Data quality validation activities

4. Service Architecture (DoDAF SvcV-1, SvcV-4, Core Logical/Functional)

- FHIR-based RESTful API services
- HL7 v2 message handling services
- Master Patient Index (MPI) service
- Terminology translation service

5. Information Model (DoDAF DIV-2, Core Information/Data)

- Patient demographic model
- Clinical document model (aligned with FHIR resources)
- Consent/authorization model
- Provenance and audit model

6. Standards Profile (DoDAF StdV-1, Core Development/Implementation)

- HL7 FHIR R4
- HL7 v2.5.1 (for legacy integration)
- IHE profiles (PIX, PDQ, XDS)
- SNOMED CT and LOINC terminologies
- OAuth 2.0 and OpenID Connect for authentication
- TLS 1.3 for transport security

7. System Deployment (DoDAF SV-1, SV-2, Core Deployment/Physical)

- Secure cloud infrastructure
- Data centers with HIPAA-compliant hosting
- Network security zones and segmentation
- Redundant systems for high availability

6.2.4 Cross-Framework Translation

When a hospital (using Core Viewpoints internally) integrates with the HIE (documented in DoDAF):

Hospital Internal	Translation Activity	HIE Framework
Logical view of EMR system	Map EMR functions to HIE operational activities	OV-5b activities
Data model of patient records	Transform to FHIR resource model	DIV-2 information model
Deployment view of EMR infrastructure	Define integration nodes and interfaces	SV-1 system interfaces
Development standards	Align with HIE standards profile	StdV-1 standards

Table 4: Cross-Framework Translation Example

7 Limitations and Considerations

7.1 Mapping Limitations

7.1.1 Not Perfect One-to-One Correspondences

The mapping between frameworks is *approximate and interpretive*, not mathematically precise:

- **Conceptual Differences:** Each framework emerged from different domains (software engineering, enterprise architecture, defense systems) with distinct philosophical foundations.
- **Granularity Mismatches:** Frameworks operate at different levels of detail. DoDAF models are highly specific; Zachman cells are more abstract; core viewpoints vary in scope.
- **Overlapping Concerns:** Real architectural concerns don't divide cleanly into orthogonal categories. Security, for example, truly is a cross-cutting concern that appears everywhere.

7.1.2 Context-Dependent Interpretation

The "best" mapping often depends on:

- *System type:* Embedded systems vs. enterprise applications vs. system-of-systems
- *Organization culture:* Engineering-driven vs. business-driven organizations
- *Project phase:* Early conceptual design vs. detailed implementation
- *Stakeholder needs:* What information each stakeholder actually requires

7.2 Framework Evolution

All three frameworks continue to evolve:

- **ISO/IEC/IEEE 42010:** Updated in 2022 with enhanced guidance on architecture framework definition and digital twin architectures
- **Zachman Framework:** John Zachman and the Zachman Institute continue to refine interpretations and expand application domains
- **DoDAF:** DoD Architecture Framework continuously updates in response to technological changes and lessons learned

Users of this mapping should:

- Consult current framework documentation
- Participate in framework communities
- Adapt mappings to reflect new framework versions
- Contribute lessons learned back to the architecture community

7.3 Practical Application Challenges

7.3.1 Organizational Resistance

Introducing or translating between frameworks may encounter:

- Stakeholder preference for familiar approaches
- Perceived overhead of additional documentation

- Learning curves for new frameworks
- Tool and training investments

Mitigation strategies:

- Start with pilot projects
- Focus on high-value architectural artifacts
- Provide concrete examples from similar projects
- Emphasize benefits (communication, consistency, completeness)

7.3.2 Tool Support Variability

Architecture tools vary widely in framework support:

- Some tools specialize in one framework (e.g., DoDAF-specific tools)
- General-purpose modeling tools may support multiple frameworks
- Custom frameworks may require tool customization
- Inconsistent import/export between tools

Recommendations:

- Evaluate tool support for required frameworks before selection
- Consider framework-agnostic core data models
- Maintain architecture data in standardized formats (XML, JSON)
- Develop or acquire transformation scripts for cross-framework translation

7.4 When Not to Use Multiple Frameworks

Multiple frameworks may be counterproductive when:

- Projects are small or short-duration
- Stakeholder community is homogeneous
- Regulatory/contractual requirements mandate a single framework
- Team lacks architectural maturity
- Tool infrastructure is limited

In such cases, select the *single best-fit framework* and use it consistently, rather than attempting to maintain multiple parallel representations.

8 Conclusion

8.1 Key Takeaways

1. **Complementary, Not Competitive:** Core viewpoints, Zachman, and DoDAF are complementary frameworks, each offering distinct advantages for different contexts.
2. **Translation is Possible:** While not perfectly one-to-one, meaningful mappings exist that enable translation of architectural concepts and artifacts between frameworks.
3. **Context Matters:** The most appropriate framework(s) depend on organizational context, system characteristics, stakeholder needs, and project constraints.
4. **Pragmatic Application:** Successful architecture practice often involves selective use of framework elements rather than rigid, comprehensive application.
5. **Evolutionary Approach:** Architectural documentation can start simple and evolve to more sophisticated frameworks as projects mature and organizational needs grow.

8.2 Recommended Practices

1. **Start with Stakeholder Needs:** Identify what information stakeholders actually need before selecting frameworks and viewpoints.
2. **Maintain Consistency:** When using multiple frameworks, ensure architectural decisions are consistently represented across frameworks.
3. **Document Framework Usage:** Explicitly state which frameworks you're using, how you're mapping between them, and any adaptations you've made.
4. **Leverage Standards:** Align with relevant standards (ISO/IEC/IEEE 42010, industry-specific standards) to ensure broader applicability.
5. **Tool-Independent Data:** Maintain architecture data in framework-agnostic formats to preserve flexibility and enable tool migration.
6. **Continuous Learning:** Architecture frameworks and practices evolve. Engage with the architecture community and stay current with framework updates.

8.3 Future Directions

Several trends are shaping the evolution of architectural frameworks:

- **Digital Twin Integration:** Architecture frameworks increasingly address digital twin concepts, blurring lines between design-time and runtime architectural representations.
- **AI/ML Systems Architecture:** Emerging patterns for documenting machine learning systems, data pipelines, and AI governance.
- **Cloud-Native Patterns:** Framework adaptations addressing microservices, containers, serverless, and cloud-native architectures.
- **Cybersecurity Integration:** Security architecture increasingly integrated throughout frameworks rather than treated as separate concern.
- **Sustainability and Green Architecture:** Growing emphasis on architectural decisions' environmental impact.
- **Continuous Architecture:** Frameworks adapting to DevOps, continuous delivery, and evolutionary architecture practices.

8.4 Final Thoughts

Architecture frameworks are tools, not ends in themselves. The ultimate goal is creating and communicating effective architectures that serve stakeholder needs and enable successful system delivery and operation. This mapping guide aims to help architects navigate the framework landscape, leveraging the strengths of multiple approaches while avoiding unnecessary complexity.

Whether you work primarily with one framework or regularly translate between them, understanding these correspondences enhances your ability to communicate architectural concepts, learn from diverse architectural practices, and adapt your approach to varied organizational contexts.

Appendix A: Quick Reference Guide

Core Viewpoint Summary

Context System boundary, external entities, environment

Logical/Functional Functional decomposition, logical organization

Process/Concurrency Runtime behavior, performance, scalability

Development/Implementation Code organization, build processes, teams

Information/Data Data models, information flow, persistence

Operational Business processes, operational workflows, support

Deployment/Physical Infrastructure, network topology, hardware

Zachman Framework Quick Reference

Perspectives (Rows): Planner, Owner, Designer, Builder, Subcontractor, Functioning Enterprise

Interrogatives (Columns): What (Data), How (Function), Where (Network), Who (People), When (Time), Why (Motivation)

DoDAF Viewpoint Families

AV All Viewpoint—architecture overview and summary

CV Capability Viewpoint—mission capabilities

DIV Data and Information Viewpoint—data models

OV Operational Viewpoint—business/mission operations

PV Project Viewpoint—implementation projects

StdV Standards Viewpoint—technical standards

SV Systems Viewpoint—system implementation

SvcV Services Viewpoint—service-oriented architecture

Common Model Numbers

- **Overview:** AV-1, OV-1, CV-1
- **Activities/Functions:** OV-5a/5b, SvcV-4, SV-4
- **Sequences/Dynamics:** OV-6b/6c, SvcV-10b/10c, SV-10b/10c
- **Data Models:** DIV-1 (conceptual), DIV-2 (logical), DIV-3 (physical)
- **Deployment:** SV-1, SV-2, SvcV-1, SvcV-2
- **Standards:** StdV-1, StdV-2
- **Projects:** PV-1, PV-2, PV-3

Appendix B: Glossary

Architecture Description Work product used to express an architecture (ISO/IEC/IEEE 42010)

Architecture Framework Conventions, principles, and practices for the description of architectures

Architecture View Work product expressing the architecture from the perspective of specific concerns

Architecture Viewpoint Specification for constructing and using an architecture view

Concern Interest in a system relevant to one or more stakeholders

DoDAF Department of Defense Architecture Framework

DoDAF Meta Model (DM2) Data model underlying DoDAF 2.0 architecture descriptions

ISO/IEC/IEEE 42010 International standard for architecture description

Interrogative Question word (What, How, Where, Who, When, Why) in Zachman Framework

Model Specific numbered diagram or table type in DoDAF (e.g., OV-1, SV-4)

Perspective Row in Zachman Framework representing stakeholder abstraction level

Stakeholder Individual, team, or organization with interests in or concerns about the architecture

Viewpoint Family Collection of related models in DoDAF (e.g., OV, SV, DIV)

Zachman Cell Intersection of a perspective (row) and interrogative (column) in Zachman Framework

Appendix C: References and Further Reading

Standards

- ISO/IEC/IEEE 42010:2022, *Systems and software engineering — Architecture description*
- IEEE 1471-2000, *Recommended Practice for Architectural Description of Software-Intensive Systems* (superseded by 42010)

DoDAF Resources

- U.S. Department of Defense CIO, *DoD Architecture Framework (DoDAF) Version 2.02*, <https://dodcio.defense.gov/Library/DoD-Architecture-Framework/>
- DoDAF Meta Model (DM2), https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_background/
- DoDAF Viewpoints and Models, https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_viewpoints/

Zachman Framework Resources

- Zachman International, *The Zachman Framework*, <https://zachman-feac.com/>
- LeanIX, *The Zachman Framework – A Definitive Guide*, <https://www.leanix.net/en/wiki/ea/zachman-framework>
- Ardoq, *What is the Zachman Framework?*, <https://www.ardoq.com/knowledge-hub/zachman-framework>

4+1 View Model

- Kruchten, P. (1995), *The 4+1 View Model of Architecture*, IEEE Software, 12(6), 42-50.
- Available at: <https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>

General Architecture Resources

- Bass, L., Clements, P., & Kazman, R. (2021), *Software Architecture in Practice*, 4th Edition, Addison-Wesley
- Clements, P., et al. (2010), *Documenting Software Architectures: Views and Beyond*, 2nd Edition, Addison-Wesley
- Rozanski, N., & Woods, E. (2011), *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*, 2nd Edition, Addison-Wesley
- The Open Group, *TOGAF Standard*, <https://www.opengroup.org/togaf>

Online Communities and Resources

- IMT AG Blog, *Architecture documentation with viewpoints*, <https://www.imt.ch/en/expert-blog-detail/architecture-documentation-with-viewpoints-en>
- SEI Software Architecture Resources, <https://www.sei.cmu.edu/our-work/software-architecture/>
- Architecture & Governance Magazine, <https://www.architectureandgovernance.com/>