PACT: A Reference Viewpoint Taxonomy for Software-Intensive Systems

Huiwen Han *

Posted Date: 9 January 2026

doi: 10.20944/preprints202601.0720.v1

Keywords: architecture viewpoints; architecture documentation; viewpoint taxonomy; ISO/IEC/IEEE 42010; enterprise architecture

 doi:10.20944/preprints202601.0720.v1

Article

PACT: A Reference Viewpoint Taxonomy for Software-Intensive Systems

Huiwen Han

Beijing, China; hanhuiwen@gmail.com

Abstract

Architecture viewpoints play a central role as an abstraction mechanism for structuring, communicating, and reviewing software architectures by separating concerns and addressing diverse stakeholder needs [1,8,9,11]. However, in both industrial practice and academic research, viewpoint definitions are often fragmented, inconsistently expressed, or narrowly scoped, which limits comparability, reuse, and long-term architectural evolution [10,14,15]. Existing architecture frameworks and standards, including TOGAF, C4, and ISO/IEC/IEEE 42010, either emphasize processes and notations or deliberately avoid prescribing concrete viewpoint sets [1,4,33]. While this flexibility supports broad applicability, it also leaves practitioners without a reusable reference taxonomy that systematically consolidates architectural concerns encountered in modern software-intensive systems [8,10,11]. This paper introduces PACT (Practical Architecture Viewpoint Taxonomy), a reference-level taxonomy of architecture viewpoints that consolidates recurring architectural concerns observed across standards, established viewpoint models, and industrial practice. PACT defines 52 viewpoints spanning business, application, integration, data, security, infrastructure, governance, and operations concerns. Each viewpoint is specified using a unified definition template that captures its primary concern, key questions, stakeholders, abstraction focus, and scope, enabling systematic comparison, selection, and reuse [1,11]. PACT is explicitly aligned with the conceptual model of ISO/IEC/IEEE 42010, while remaining method-, notation-, and tool-independent, enabling reuse across heterogeneous architectural practices without imposing process or documentation lock-in [1]. It is intended as a reference taxonomy rather than a prescriptive framework, supporting enterprise architecture governance and system design practices [6,7], as well as academic analysis of architectural knowledge and viewpoints [15,18]. A case-based evaluation and industrial illustrations illustrate how PACT supports more systematic concern coverage, improved clarity, and structured architecture reviews across heterogeneous systems. The taxonomy is designed to be extensible, providing a stable reference for future research and evolving architectural practices [10,32].

Keywords: architecture viewpoints; architecture documentation; viewpoint taxonomy; ISO/IEC/IEEE 42010; enterprise architecture

1. Introduction

Software-intensive systems have grown increasingly complex, spanning enterprise-scale business processes, distributed application architectures, data-intensive pipelines, and cloud-native operational environments [20,22,24]. As a result, architecture documentation has become a critical instrument for reasoning about system structure, behavior, risks, and organizational responsibilities [1,11,14]. Architecture viewpoints are widely recognized as the primary abstraction mechanism for addressing diverse stakeholder concerns in such systems [1,8,11].

Despite their importance, architecture viewpoints are applied inconsistently in practice [10,14,15].Many projects rely on ad hoc or tool-driven sets of diagrams, while others adopt framework-specific artifacts without a clear conceptual foundation [4,5,14]. This fragmentation makes architecture descriptions difficult to compare, reuse, and evolve across projects, organizations, and

 doi:10.20944/preprints202601.0720.v1

domains [10,11,14,15]. In particular, enterprise architecture concerns, solution-level technical design, and software development architectures are often documented using disconnected or incompatible models [4,5,8,33].

International standards such as ISO/IEC/IEEE 42010 establish a rigorous conceptual basis for architecture descriptions by defining stakeholders, concerns, viewpoints, and views [1,8]. However, the standard deliberately avoids prescribing concrete viewpoint sets. Similarly, widely used frameworks and models such as TOGAF and C4 either emphasize process and governance or focus on a limited subset of structural views [4,8,33]. As a result, practitioners and researchers lack a reusable, standardized, and reusable taxonomy of architecture viewpoints that reflects the breadth of modern software-intensive systems.

This paper addresses this gap by introducing PACT (Practical Architecture Viewpoint Taxonomy), a reference taxonomy of architecture viewpoints designed for long-term reuse and evolution. PACT defines 52 viewpoints organized across eight architectural layers, covering business, application, integration, data, security, infrastructure, governance, and operations concerns. Each viewpoint is specified using a unified template, enabling consistency, comparability, and extensibility.

PACT is not a development method or documentation framework. Instead, it serves as a stable reference that can be mapped to existing architecture artifacts, enterprise governance processes, and software design practices. By providing a reference and standardized viewpoint taxonomy aligned with ISO/IEC/IEEE 42010, this work aims to support architectural reasoning in both industrial and academic contexts, and to provide a durable foundation for future research and practice.

2. Background and Related Work

2.1. Architecture Frameworks and Standards

A variety of architecture frameworks and standards have been proposed to support the description, analysis, and governance of complex software systems[2]. Among the most influential are TOGAF, the Zachman Framework, the C4 model, and ISO/IEC/IEEE 42010 [1,4,5,33].

TOGAF provides a reference enterprise architecture methodology, including an Architecture Development Method (ADM), layered views, and a catalogue of artifacts [4,6] Its primary strength lies in governance processes and artifact organization. However, TOGAF does not define explicit, reusable viewpoint constructs in the sense of ISO/IEC/IEEE 42010; artifacts are described primarily as deliverables tied to ADM phases rather than as analytical instruments for architectural reasoning [1,4]. As a result, its artifact guidance tends to be prescriptive and organization-centric, with significant variability across adaptations and limited support for systematic viewpoint reuse [7].

The Zachman Framework offers a two-dimensional classification scheme based on interrogatives (what, how, where, who, when, why) and abstraction levels [5]. While Zachman effectively structures architectural information, it does not define concrete viewpoint semantics, templates, or activation logic. Consequently, Zachman functions more as a conceptual organizing matrix than as a systematic viewpoint taxonomy suitable for consistent reuse or comparison across systems [6,8].

The C4 model focuses on documenting software system structure through a fixed, four-level set of diagrams: Context, Container, Component, and Code [33]. Its simplicity and clarity have made it popular in agile and development teams. However, by design, C4 addresses primarily software structural concerns; it does not systematically incorporate business, data, security, governance, or operational perspectives and is not intended as a reference viewpoint taxonomy [9,10].

ISO/IEC/IEEE 42010 defines architecture descriptions in terms of stakeholders, concerns, viewpoints, and views, establishing a shared vocabulary and a rigorous conceptual foundation for architectural description [1]. Importantly, the standard deliberately avoids prescribing a specific set of viewpoints or rules for their selection. Instead, it provides a meta-model intended to support consistency and traceability across architecture descriptions. As a result, while ISO/IEC/IEEE 42010

 doi:10.20944/preprints202601.0720.v1

ensures conceptual coherence, it leaves unanswered the practical question of which viewpoints should be defined, organized, and reused across different system contexts [1,11].

Taken together, these frameworks illustrate a persistent tension in architectural practice: there is abundant guidance on what artifacts exist and how they may be structured, but limited guidance on how viewpoints should be systematically defined, selected, and reused across heterogeneous systems [8,10,14]. This observation motivates the need for a reference taxonomy that provides both breadth of coverage and conceptual consistency.

2.2. Viewpoint Taxonomies and Related Research

Prior research and industrial frameworks have proposed sets of architecture views or viewpoints tailored to specific domains or quality concerns. For example, work on safety-critical systems emphasizes process, hazard, and assurance views, while service-oriented and microservice architectures focus on interaction, decomposition, and deployment views [9,20,21]. Studies in data-intensive and cloud-native environments further highlight the importance of data flow, pipeline, and infrastructure views [22,24].

Despite this diversity, existing taxonomy efforts exhibit several recurring limitations.

First, many proposed viewpoint sets are domain-specific, addressing only a limited class of systems such as embedded, safety-critical, or enterprise systems, which restricts their applicability across architecture domains [10,12].

Second, viewpoint definitions are frequently informal or implicit, lacking rigorous templates or well-defined semantics, which complicates systematic reuse, comparison, and evolution [11,14].

Third, few approaches propose a unified viewpoint definition template that explicitly supports extensibility and long-term architectural knowledge management [15,18].

Finally, most existing taxonomies do not explicitly address how viewpoint sets should evolve in response to modern architectural paradigms, including cloud-native platforms, large-scale data pipelines, AI-enabled systems, and highly dynamic operating environments [24,38,40].

Consequently, there is currently no widely accepted, systematic, and reusable viewpoint taxonomy that serves as a stable reference across both industrial practice and architectural research [10,14,15].

Prior work on viewpoint taxonomies and architecture documentation also frequently overlooks a practical concern in industry: the trade-off between documentation effort and analytical effectiveness. Empirical studies show that extensive artifact sets often lead to over-documentation, stakeholder fatigue, and reduced review effectiveness when documentation effort is not aligned with decision-making needs [14,16]. While some approaches enumerate large numbers of artifacts, they provide limited guidance on which artifacts deliver the highest analytical value relative to the effort required to produce them.

In contrast, the taxonomy presented in this paper is explicitly designed to support value-driven artifact selection, where viewpoints are mapped to a curated set of commonly used artifacts prioritized for their analytical leverage and manageable modeling effort [11,16]. This enables organizations to balance architectural coverage with efficiency in documentation and review practices.

2.3. Positioning and Gap Analysis

In summary, existing approaches to architectural viewpoints and documentation tend to fall into one of three categories:

- **Conceptual standards without concrete viewpoint sets**, such as ISO/IEC/IEEE 42010 [1];
- **Prescriptive artifact models without unified abstraction**, such as TOGAF and C4 [4,33]; or
- **Domain-specific taxonomy efforts** with limited generality and reuse potential [10,12].

The taxonomy proposed in this paper is positioned at the intersection of these approaches. It provides a systematic, standardized, and extensible set of architecture viewpoints explicitly aligned

 doi:10.20944/preprints202601.0720.v1

with the conceptual model of ISO/IEC/IEEE 42010, while remaining grounded in practical architectural concerns observed across heterogeneous systems [1,14]. Unlike prior taxonomy efforts, the proposed approach adopts a unified viewpoint definition template, supports cross-domain coverage, and is designed for long-term reuse, comparison, and evolution of architectural knowledge [11,15,18].

3. Problem Characterization and Motivation

3.1. Limitations of Current Architecture Documentation Practices

Despite the widespread recognition of architecture viewpoints in both standards and textbooks [3], current architecture documentation practices continue to exhibit several persistent problems [1,9,11].

First, architectural coverage is often incomplete. Many architecture descriptions concentrate primarily on structural software components, while business capabilities, data assets, security concerns, operational constraints, and governance mechanisms are documented inconsistently or omitted entirely [8,10,14]. This selective coverage creates architectural blind spots that frequently surface only at late stages of development or during system operation, when remediation is costly and disruptive [16,31].

Second, the expression of architecture viewpoints is highly inconsistent across projects and organizations. Similar concerns may be addressed using different diagram types, terminology, and levels of abstraction, even within the same enterprise [11,14]. Viewpoints are often implicit rather than explicitly defined, which makes it difficult for stakeholders to understand the intent, scope, and assumptions underlying a particular architectural view [1,15].

Third, comparability and reuse of architectural knowledge are limited. In the absence of standardized viewpoint definitions, architecture descriptions cannot be systematically compared across systems or reused across projects [15,18]. This limitation significantly reduces the effectiveness of enterprise architecture governance, architectural reviews, and empirical analysis of architectural practices [7,16].

3.2. Industrial Pain Points

These limitations become particularly acute in large-scale, distributed, and heterogeneous environments commonly found in contemporary enterprises and platform-based systems [6,22].

Cross-team communication degrades when stakeholders lack a shared understanding of which viewpoints are relevant and what architectural questions each viewpoint is intended to address [10,14]. Enterprise architecture reviews become inefficient and subjective, as reviewers must infer missing viewpoints, reconcile incompatible documentation styles, or rely on personal experience rather than shared analytical structures [7,16].

The emergence of AI- and data-intensive systems further amplifies these problems. Concerns related to data pipelines, data lineage, model lifecycle management, regulatory compliance, and operational risk are rarely captured through traditional software-centric viewpoints that focus primarily on components and interactions [24,38,39]. As a result, critical architectural risks in such systems often remain implicit or are addressed in isolated documents disconnected from the main architecture description [40].

In practice, these issues manifest as increased review cycles, delayed risk identification, and reduced confidence in architectural decisions, particularly in environments subject to regulatory scrutiny or rapid technological change [14,16,28].

3.3. Motivation and Objectives

The motivation of this work is to address the above challenges by providing an industrial-grade, academically citable taxonomy of architecture viewpoints aligned with established architectural

 doi:10.20944/preprints202601.0720.v1

standards and practices [1,9,11]. Rather than prescribing a fixed documentation process or methodology, PACT is conceived as a reference taxonomy. It offers a common structural foundation that architects, organizations, and researchers can adopt, tailor, and extend according to local contexts and constraints.

The objectives of this work are threefold.

First, to improve coverage of architectural concerns across modern software-intensive systems, including business, application, data, security, infrastructure, governance, and operational dimensions [6,10,24].

Second, to enable consistency and comparability of architecture descriptions through a unified viewpoint definition template, supporting clearer communication and more systematic review [11,15].

Third, to support long-term evolution and reuse of architectural knowledge, ensuring that viewpoint definitions remain relevant as technologies, organizational structures, and development practices continue to evolve [18,30,31].

By consolidating architecture viewpoints into a coherent and extensible taxonomy, this paper aims to strengthen both architectural practice and empirical research, providing a stable foundation upon which future frameworks, tools, and studies can build [14,18].

4. Meta-Model of the PACT Taxonomy

This section introduces the meta-model underlying PACT (Practical Architecture Viewpoint Taxonomy).

The meta-model defines how viewpoints are structured, identified, and related, providing a stable foundation for extensibility, comparison, and long-term reuse, consistent with established principles of architecture description and documentation [1,11].

4.1. Viewpoints as First-Class Architectural Abstractions

In PACT, a viewpoint is treated as a first-class architectural abstraction, consistent with the conceptual model defined in ISO/IEC/IEEE 42010 [1].

A viewpoint specifies *what architectural concern is addressed*, independently of *how that concern is represented* in documentation artifacts.

This distinction between concerns and representations has long been emphasized in software architecture literature, where viewpoints are understood as analytical lenses rather than diagram types or notations [8,9,11,13]. Accordingly, PACT deliberately separates:

- **Viewpoints**, which define architectural concerns, and the questions used to reason about them, from
- **Artifacts**, which provide concrete representations such as diagrams, matrices, tables, or structured descriptions.

This separation avoids conflating conceptual intent with specific tools, modeling languages, or notations—a problem frequently observed in industrial documentation practices [14]. By decoupling viewpoints from artifacts, the taxonomy remains stable even as documentation styles, tooling ecosystems, and architectural practices evolve over time [15,18].

4.2. Unified Viewpoint Definition Template

All viewpoints in PACT—both core and extended—are defined using a unified definition template to ensure consistency, comparability, and controlled evolution. The use of explicit templates for viewpoint definition is a well-established practice in architecture documentation and evaluation approaches, including view-based documentation methods and architecture knowledge management frameworks [10,11,18].

Each viewpoint in PACT is specified by the following attributes:

- **Viewpoint Name**

 doi:10.20944/preprints202601.0720.v1

A stable and descriptive identifier.

- **Primary Concern**

  The architectural concern the viewpoint is intended to address.

- **Key Questions Answered**

  The principal questions that the viewpoint enables stakeholders to reason about.

- **Primary Stakeholders**

  Roles or organizational actors for whom the viewpoint is primarily relevant.

- **Structural / Behavioral Focus**

  Whether the viewpoint emphasizes static structure, dynamic behavior, or both, reflecting common distinctions in architectural analysis [8,9].

- **Logical / Physical Scope**

  Whether the viewpoint addresses conceptual/logical aspects, physical realization, or both, consistent with established abstraction dimensions in architecture modeling [10,36].

This template enables viewpoints to be compared systematically, reviewed consistently, and extended incrementally without redefining foundational concepts. Defining viewpoints through a fixed, concern-driven template supports consistent interpretation and comparison across projects and organizations, without prescribing notations or artifact forms [1,14].

4.3. Layered Organization of Viewpoints

To manage scale and complexity, PACT organizes viewpoints into a set of conceptual layers.

Layers do not imply strict architectural tiers, implementation order, or development phases. Instead, they group viewpoints according to their dominant concern and abstraction focus, a practice commonly adopted in enterprise and software architecture frameworks to improve comprehensibility and navigability [5,6,10].

The layered organization supports:

- navigability for practitioners dealing with large viewpoint sets,
- selective adoption in industrial contexts where only subsets of concerns are relevant, and
- controlled expansion of the taxonomy as new architectural concerns emerges.

This approach reflects empirical observations that large, unstructured collections of views reduce usability and adoption in practice [14,16].

4.4. Prefix-Based Viewpoint Identification Scheme

Instead of linear numbering (e.g., 1–52), PACT adopts a prefix-based identification scheme for viewpoints.

Each viewpoint identifier consists of:

- a layer prefix, and
- a local numeric index within that layer.

Prefix-based identification schemes are commonly used in large classification systems to improve extensibility and semantic clarity, particularly when taxonomies are expected to evolve over time [6,35]. In PACT, this scheme avoids renumbering when viewpoints are added or refined and makes the architectural role of each viewpoint explicit at a glance.

The current PACT taxonomy comprises the following layers and prefixes:

- **B** — Business / Organization
- **A** — Context & Application
- **C** — Container / Component / Technical Service
- **IP** — Integration & Processing
- **D** — Data Architecture
- **SCR** — Security, Compliance & Risk
- **AGD** — Architecture Governance & Decision

 doi:10.20944/preprints202601.0720.v1

- **DIN** — Deployment / Infrastructure / Networking
- **OR** — Operations & Reliability

For example, *C3* denotes the third viewpoint in the Container / Component layer, while *AGD2* denotes the second viewpoint related to architectural governance and decision-making.

4.5. Architecture Governance & Decision as a Meta-Layer

A distinctive feature of PACT is the explicit introduction of the **Architecture Governance & Decision (AGD)** layer.

Unlike other layers, which primarily describe architectural structures or behaviors, AGD viewpoints focus on:

- architectural decisions and their rationale,
- trade-offs among competing quality attributes,
- evolution and change impact, and
- organizational and economic alignment.

Prior research has shown that architectural decisions and their rationale are critical to long-term system sustainability yet are often underrepresented or informally captured in architecture documentation [17–19,32]. AGD viewpoints therefore operate at a *meta-level*: they reason about architectural structures and viewpoints rather than being structural views themselves.

To reflect this role, AGD is positioned as a conceptual overlay layer in the taxonomy. It can be applied across all other layers, providing governance context and decision rationale for business, application, data, security, infrastructure, and operational viewpoints. This design aligns with established architecture evaluation and decision-centric approaches such as ATAM and architectural knowledge management [17,18].

4.6. Core and Extensible Viewpoints

PACT defines:

- a **core set of viewpoints**, capturing stable and recurring architectural concerns observed across domains, and
- an **extensible structure**, allowing additional viewpoints to be introduced using the same template and prefix-based identification scheme.

New viewpoints can be added by:

- introducing new identifiers within an existing layer, or
- defining a new layer when emerging concerns cannot be coherently integrated elsewhere.

This extensibility reflects the widely recognized need for architecture descriptions to evolve alongside technologies, organizational structures, and system contexts, without fragmenting into incompatible or tool-specific viewpoint sets [1,15,30].

4.7. Summary

The PACT meta-model establishes:

- viewpoints as reusable, concern-driven architectural abstractions,
- a unified and explicit viewpoint definition template,
- a layered conceptual organization, and
- a prefix-based identification scheme that supports long-term evolution.

By explicitly modeling architectural governance and decision-making as a dedicated meta-layer, PACT extends beyond descriptive architecture documentation and supports reflective, decision-oriented architectural reasoning, addressing gaps identified in both industrial practice and existing architectural frameworks [14,18,19].

 doi:10.20944/preprints202601.0720.v1

5. Full Viewpoint Taxonomy

This section presents the full architecture viewpoint taxonomy, which constitutes the core contribution of this paper. The taxonomy operationalizes the viewpoint abstraction introduced in Section 4 by instantiating **52 architecture viewpoints**, each defined using a unified template and organized into coherent architectural layers. The notion of organizing architectural knowledge through explicitly defined viewpoints is well established in software architecture research and standards [1,8,11].

5.1. Core Contribution and Design Principles

PACT is designed according to three guiding principles, derived from limitations observed in existing architecture documentation practices and prior research on viewpoint-based architectural description [11,14,18].

Broad and Structured Coverage

The taxonomy spans the full spectrum of concerns observed in modern software-intensive systems, including business, application, integration, data, security, infrastructure, governance, and operational concerns. Prior studies have shown that architecture documentation often overemphasizes software structure while underrepresenting data, security, and operational viewpoints, leading to blind spots in architectural reasoning [14,16]. By explicitly including these concern areas, PACT is intended to serve as a common reference across heterogeneous system types, including data-intensive, cloud-native, and AI-enabled systems [24,38,39].

Explicit Alignment with ISO/IEC/IEEE 42010

Each viewpoint is explicitly defined in terms of stakeholder concerns, scope, and analytical focus, directly reflecting the conceptual model of ISO/IEC/IEEE 42010 [1]. While the standard deliberately avoids prescribing concrete viewpoint sets, PACT complements it by providing a reusable instantiation of viewpoints that remains independent of notations, modeling languages, or tools. This approach follows established recommendations for separating architectural concerns from representation mechanisms [1,11].

Evolvability and Reusability

The taxonomy is not intended as a fixed or closed list. Viewpoints are designed to be reusable across projects, composable within architecture reviews, and extensible over time as new architectural concerns emerge. Prior work on architectural knowledge management and decision documentation highlights the importance of evolvable abstractions that can accommodate system growth and technological change without fragmenting architectural descriptions [15,18,30].

Together, these principles position the taxonomy as a **reference taxonomy** rather than a prescriptive framework, suitable for enterprise architecture governance, system design, and academic research [6,7,10].

5.2. Full Viewpoint Taxonomy

Table 1 presents the full PACT viewpoint taxonomy, organized using the layered structure and prefix-based identification scheme defined in Section 4.

Each viewpoint is defined using the unified template, ensuring consistency, comparability, and extensibility, consistent with best practices in view-based architecture documentation [11,14].

The taxonomy is intended as a **reference viewpoint set**: projects may selectively activate viewpoints based on scope, risk, and documentation effort, without compromising conceptual completeness. This selective adoption principle reflects empirical findings that value-driven documentation improves review effectiveness and stakeholder engagement [14,16].

 doi:10.20944/preprints202601.0720.v1

This table presents the complete set of architecture viewpoints defined by the PACT (Practical Architecture Viewpoint Taxonomy).

Viewpoints are organized into conceptual layers and identified using a prefix-based scheme that reflects their dominant architectural concern.

Each viewpoint is defined using a unified template, enabling systematic comparison, selective adoption, and long-term evolution across heterogeneous software-intensive systems.

Table 1. PACT — Full Architecture Viewpoint Taxonomy (52 Viewpoints).

Business / Organization Layer (B)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
B1	Business Capability View	Capability structure	What business capabilities exist and which are critical?	Executives, EA	Structural	Logical
B2	Business Process View	End-to-end workflows	How do business processes execute across domains?	Product, Operations	Behavioral	Logical
B3	User / Actor View	Users and roles	Who interacts with the system and in what roles?	Product, Security	Structural	Logical
B4	Organizational Ownership View	Accountability & governance	Who owns systems, data, and decisions?	Executives, EA	Structural	Logical
B5	Interaction & Influence View	Cross-domain influence	How do organizational units influence each other?	Product, Architecture	Structural	Logical

Context & Application Layer (A)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
A1	System Context View (C4 L1)	System boundary	What is inside and outside the system?	Product, Development	Structural	Logical
A2	Application Functional View	Functional decomposition	How are application functions grouped?	Product, Development	Structural	Logical
A3	Application Interaction View	Application collaboration	How do applications interact and depend on each other?	Development, Integration	Behavioral	Logical
A4	User Role–Application Mapping View	Access responsibility	Which roles can access which applications?	Product, Security	Structural / Behavioral	Logical

Container / Component / Technical Service Layer (C)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
C1	Container View (C4 L2)	Runtime containers	What are the main runtime units?	Solution Architects, Dev	Structural	Logical
C2	Component View (C4 L3)	Internal modularity	How are components structured within containers?	Solution Architects, Dev	Structural	Logical
C3	Technical Service View	Service responsibility	What capability boundary does each service own?	Dev, Architects	Structural	Logical

 doi:10.20944/preprints202601.0720.v1

C4	Technical Stack View	Technology choices	Which languages, frameworks, and platforms are used?	Dev, Operations	Structural	Physical
C5	Communication & Protocol View	Interaction mechanisms	How do components communicate?	Dev, SRE	Structural / Behavioral	Logical / Physical

Integration & Processing Layer (IP)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
IP1	Application Integration View	Business-level integration	How do applications exchange data?	Product, Integration	Behavioral	Logical
IP2	Technical Integration View	Technical constraints	What latency, throughput, or coupling constraints exist?	Dev, SRE	Behavioral	Physical
IP3	System Processing View	Execution semantics	How are requests processed end-to-end?	Dev, Operations	Behavioral	Logical
IP4	Activity View	Action paths	What actions and decisions occur during execution?	Architects, Dev	Behavioral	Logical
IP5	Process Flow View	Workflow execution	How do workflows orchestrate services?	Product, SRE	Behavioral	Logical
IP6	Sequence Interaction View	Call ordering	What is the interaction sequence?	Dev, Architects	Behavioral	Logical

Data Architecture Layer (D)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
D1	Business Data Flow View	Business data movement	How does data support business processes?	Product, Operations	Behavioral	Logical
D2	Data Derivation & Evolution View	Data transformation	How is data derived and transformed?	Data Engineering, ML	Behavioral	Logical / Physical
D3	Data Lineage View	Traceability	Where does data originate and propagate?	Compliance, Data Eng	Behavioral	Logical / Physical
D4	Data Asset View	Ownership & value	What data assets exist and who owns them?	Data Office	Structural	Physical
D5	Conceptual Data Entity View	Business entities	What core business entities exist?	Data Architects, Business	Structural	Logical
D6	Data Quality View	Quality assurance	Is data accurate, complete, and timely?	Data Stewards	Structural / Behavioral	Logical / Physical
D7	Data Compliance Flow View	Regulatory paths	Does data usage comply with regulations?	Compliance, Product	Behavioral	Logical / Physical
D8	Data Pipeline View	Technical pipelines	How is data processed technically?	Data Engineering	Behavioral	Physical
D9	Data Sovereignty View	Residency constraints	Where is data stored and restricted?	Compliance, EA	Structural	Physical

 doi:10.20944/preprints202601.0720.v1

D10	Logical Data Model View	Logical schema	How are entities logically related?	Dev, Data Architects	Structural	Logical
D11	Physical Data Model View	Physical schema	How is data physically implemented?	DBA, Dev	Structural	Physical

Security, Compliance & Risk Layer (SCR)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
SCR1	Security Control View	Control mechanisms	What controls enforce security policy?	Security, DevOps	Structural / Behavioral	Logical / Physical
SCR2	Threat & Risk Modeling View	Threat analysis	What threats and risks exist?	Security, Risk	Structural / Behavioral	Logical / Physical
SCR3	Attack Path View	Adversarial traversal	How could an attacker move through the system?	Security, IR	Behavioral	Physical
SCR4	Service Authentication View	Service trust	How do services authenticate each other?	Security, Dev	Structural / Behavioral	Logical / Physical
SCR5	User Authentication View	User identity	How are users authenticated?	Security, Product	Structural / Behavioral	Logical / Physical
SCR6	Authorization & Access Control View	Access enforcement	Who is authorized to do what?	Security, Compliance	Structural / Behavioral	Logical
SCR7	Compliance Boundary View	Regulatory scope	Which regulations apply and where?	Compliance, EA	Structural	Logical / Physical

Architecture Governance & Decision Layer (AGD) ***(Meta-layer)***

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
AGD1	Architecture Decision View	Decision rationale	What architectural decisions were made and why?	Architects, EA	Structural	Logical
AGD2	Evolution & Change Impact View	Change analysis	What is impacted by a proposed change?	Architects, Product	Behavioral	Logical
AGD3	Quality Attribute Trade-off View	Trade-off reasoning	How are quality attributes balanced?	Architects, SRE	Structural / Behavioral	Logical
AGD4	Cost & Value View	Economic impact	What are the costs and expected value?	Executives, Finance	Structural	Logical
AGD5	Team–Architecture Alignment View	Organizational fit	Are teams aligned with the architecture?	Engineering Managers	Structural	Logical

Deployment / Infrastructure / Networking Layer (DIN)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
DIN1	Application Deployment Location View	Deployment topology	Where are applications deployed?	EA, Infrastructure	Structural	Physical

 doi:10.20944/preprints202601.0720.v1

DIN2	Network Topology View	Network segmentation	How is the network structured?	Network, Security	Structural	Physical
DIN3	Infrastructure Composition View	Infrastructure layout	What infrastructure components exist?	Dev, Operations	Structural	Physical
DIN4	Resource & Capacity View	Scaling & capacity	How are resources allocated and scaled?	SRE, Infrastructure	Structural	Physical

Operations & Reliability Layer (OR)

ID	Viewpoint Name	Primary Concern	Key Questions Answered	Primary Stakeholders	Focus	Scope
OR1	Operations View	Incident handling	How are incidents managed operationally?	Operations, Dev	Behavioral	Physical
OR2	Resilience & DR View	Recovery strategy	What are RPO/RTO objectives?	Infrastructure, Executives	Structural / Behavioral	Physical
OR3	Reliability & SLO View	Reliability targets	What service levels are required?	Dev, SRE	Structural / Behavioral	Logical
OR4	Monitoring & Observability View	System visibility	How is system health observed?	Dev, SRE	Structural	Logical
OR5	Operational Feedback View	Learning loops	How do operations inform evolution?	Architects, Operations	Behavioral	Logical

Note: Viewpoints are grouped by architectural layer for readability; the layered structure is conceptual and does not imply implementation order or strict architectural tiers. The prefix-based identifiers (e.g., B, A, C, IP, D, SCR, AGD, DIN, OR) indicate the primary concern domain of each viewpoint and support extensibility without renumbering. The taxonomy is intended as a reference viewpoint set: projects may activate a subset of viewpoints based on scope, risk, and documentation effort, without compromising conceptual completeness.

5.2.1. Summary

The PACT taxonomy provides integrated yet modular coverage of business, technical, data, governance, security, deployment, and operational concerns commonly encountered in modern software-intensive systems. By making these concerns explicit as viewpoints, the taxonomy addresses gaps identified in prior architecture documentation studies, particularly in the areas of data governance, security, and operational reasoning [14,24,27].

The prefix-based structure supports selective adoption, systematic review, and long-term evolution, making PACT suitable as a reference taxonomy for both research and industrial practice.

5.3. Reuse, Mapping, and Evolution

Each viewpoint in the taxonomy is designed to be reusable and composable, consistent with established view-based documentation approaches [11]:

- A viewpoint may be realized by one or more architecture artifacts (e.g., diagrams, matrices, catalogs).
- A single artifact may support multiple viewpoints, enabling efficient architecture reviews and reducing documentation redundancy [14].
- Viewpoints can be selectively combined to address specific stakeholder concerns or review objectives, supporting architecture evaluation and governance activities [17,18].

The taxonomy explicitly supports extension and evolution. New viewpoints can be added by instantiating the same unified template, and existing viewpoints can be refined or versioned without disrupting the overall structure. This design aligns with prior research emphasizing the need for

 doi:10.20944/preprints202601.0720.v1

architecture descriptions to evolve alongside systems and organizations [18,30,32], making PACT suitable as a long-term reference framework for both industrial practice and academic research.

6. Evaluation: Case-Based Assessment of the Taxonomy

6.1. Evaluation Objectives and Method

The objective of this evaluation is to assess the **practical applicability, coverage, and review effectiveness** of the proposed architecture viewpoint taxonomy in an industrially representative setting.

Given the nature of the contribution—a **reference-level taxonomy** rather than an algorithmic or tool-based technique—this study adopts a **case-based qualitative evaluation approach**. Such approaches are well established in systems and software engineering research for evaluating architectural frameworks, models, and documentation practices, particularly when the primary contribution is conceptual, structural, or classificatory rather than computational [15–17].

The evaluation focuses on the following research questions:

- **RQ1:** To what extent does the taxonomy provide broad and structured coverage of architectural concerns in a modern software-intensive system?
- **RQ2:** How does taxonomy-driven documentation compare with conventional documentation approaches in terms of concern visibility and review effectiveness?
- **RQ3:** Does the taxonomy support more structured and efficient enterprise architecture (EA) review and cross-team communication?

To address these questions, the taxonomy is applied to a representative industrial system and compared against a widely used baseline documentation approach, following established comparative evaluation practices for architecture descriptions [1,18]. Given the reference-level nature of the contribution, the evaluation emphasizes qualitative applicability and review effectiveness rather than quantitative performance metrics, consistent with prior evaluations of architectural frameworks and documentation approaches [6,18,22].

6.2. Case Description: A Microservice-Based E-Commerce Platform

The evaluation is conducted using a **hypothetical but industry-realistic microservice-based e-commerce platform**, designed to reflect common characteristics of contemporary enterprise systems, as commonly adopted in architectural evaluation studies where proprietary industrial systems cannot be disclosed [16,19].

The system includes:

- Core business domains such as user management, product catalog, order processing, payment, and recommendation.
- A microservice architecture deployed on a cloud-native infrastructure.
- Integration with external systems, including payment gateways and logistics providers.
- Both transactional and analytical data flows, including real-time event processing and batch-oriented reporting pipelines.
- Security, compliance, and operational requirements typical of enterprise environments, such as identity management, data protection, and availability targets.

This type of system is representative of many real-world projects encountered in enterprise IT landscapes and is frequently used as an evaluation surrogate in architecture research due to its richness of concerns and cross-cutting requirements [20,21]. It therefore provides a suitable context for evaluating the breadth and applicability of PACT.

6.3. Viewpoint Activation and Artifact Selection

Using PACT, viewpoints were selected and activated based on the system's characteristics and risk profile.

 doi:10.20944/preprints202601.0720.v1

Rather than documenting all 52 viewpoints exhaustively, only those relevant to the identified concerns were instantiated, following the principle of **risk- and context-driven viewpoint selection**, as advocated by ISO/IEC/IEEE 42010 and prior architecture documentation research [1,18].

Activated viewpoints spanned multiple architectural layers, including:

- Business and organizational concerns (e.g., business capability and process viewpoints),
- Application and integration concerns (e.g., container, interaction, and integration viewpoints),
- Data architecture concerns (e.g., data flow, lineage, and governance viewpoints),
- Security and risk concerns (e.g., authentication, threat modeling, and attack path viewpoints),
- Deployment, infrastructure, and operational concerns (e.g., deployment, network, and observability viewpoints).

For each activated viewpoint, one or more architecture artifacts were produced using the artifact definitions described in Appendix B (e.g., Technical Architecture Diagram, Application Collaboration Diagram, Data Pipeline Diagram, Authentication and Service Flow Diagram).

This ensured a **clear and traceable mapping between viewpoints and concrete architectural representations**, consistent with the separation of viewpoints and views prescribed by ISO/IEC/IEEE 42010 [1]. Actionability is achieved through the explicit mapping between viewpoints and concrete artifacts, allowing the same viewpoint to be realized through different representations depending on project context [11,18].

6.4. Comparative Analysis with Conventional Architecture Documentation Approaches

To contextualize the evaluation results, the taxonomy-driven documentation approach was compared with three widely adopted approaches in industrial practice:

- **C4 model**: developer-oriented structural documentation [33],
- **ISO/IEC/IEEE 42010–aligned descriptions**: standards-based conceptual architecture documentation [1],
- **TOGAF-oriented documentation**: enterprise architecture governance and process frameworks [4].

These approaches represent three dominant paradigms for architecture documentation and review in contemporary practice. The qualitative comparison summarized in **Table 2** focuses on viewpoint definition, concern coverage, visibility of cross-cutting concerns, and review effectiveness.

Comparison with C4-Based Documentation

C4 documentation provides a concise and accessible representation of software system structure and is widely adopted in development teams [33].

In the evaluated e-commerce case, C4 diagrams effectively captured software decomposition and deployment structure, supporting localized technical reasoning.

However, concerns related to data governance, security boundaries, operational responsibilities, and regulatory compliance were addressed only implicitly or not at all. This limitation is consistent with prior observations that developer-oriented architectural views tend to emphasize structural aspects while underrepresenting cross-cutting concerns [11,14].

The taxonomy-driven approach complements C4 by introducing explicit viewpoints for these concerns, without invalidating or replacing existing C4 diagrams. In practice, several C4 artifacts were reused to support multiple viewpoints, reflecting established practices of multi-view reuse in architecture documentation [18].

Comparison with ISO/IEC/IEEE 42010–Aligned Descriptions

ISO/IEC/IEEE 42010 establishes a rigorous conceptual model for architecture descriptions based on stakeholders, concerns, viewpoints, and views [1].

In principle, the evaluated system could be documented in compliance with the standard by defining project-specific viewpoints.

 doi:10.20944/preprints202601.0720.v1

In practice, however, the absence of a reference-level viewpoint taxonomy resulted in largely ad hoc viewpoint selection and definition. Different architects emphasized different concerns, leading to inconsistent coverage and limited comparability across projects—an issue also reported in empirical studies of architecture documentation practices [14,18].

PACT can therefore be interpreted as an operational complement to ISO/IEC/IEEE 42010: it instantiates the standard's abstract concepts through a reusable and structured set of viewpoints. In the evaluation case, this reduced ambiguity in viewpoint selection and enabled more consistent architecture reviews without constraining architectural freedom.

Comparison with TOGAF-Oriented Documentation

TOGAF provides extensive guidance on enterprise architecture processes, governance mechanisms, and artifact catalogs [4].

When applied to the case system, TOGAF-inspired documentation emphasized organizational alignment, capability mapping, and high-level solution structure.

While effective at the enterprise level, TOGAF artifacts did not sufficiently address runtime behavior, data pipelines, security flows, and operational risk—concerns that are critical for modern cloud-native and data-intensive systems [21,24].

Moreover, the lack of a uniform viewpoint abstraction made it difficult to trace concerns consistently across business, application, and infrastructure layers.

In contrast, the taxonomy-driven approach applies the same viewpoint abstraction across all layers, enabling clearer traceability between enterprise concerns, technical design decisions, and operational implications during EA review.

Table 2. Qualitative Comparison of Architecture Documentation Approaches in the Case-Based Evaluation.

Aspect	C4-Only Documentation	ISO/IEC/IEEE 42010 (Conceptual)	TOGAF-Oriented Documentation	Taxonomy-Driven Approach
Viewpoint definition	Fixed, limited	Abstract, project-specific	Implicit in artifacts	Explicit and standardized
Concern coverage	Structural focus	Depends on viewpoint selection	Enterprise-centric	Integrated and balanced
Data & security visibility	Limited	Inconsistent	Partial	Explicit and systematic
EA review effectiveness	Local	Conceptually sound but variable	Governance-oriented	Structured and risk-aware
Cross-project comparability	Low	Low–Medium	Medium	High

Note. The comparison is based on the case-based qualitative evaluation described in Section 6. It reflects relative differences in viewpoint explicitness, concern coverage, visibility of cross-cutting concerns, and review effectiveness, rather than quantitative performance measurements.

Overall, the evaluation suggests that PACT does not replace existing frameworks or standards. Instead, it connects and operationalizes them by providing a shared viewpoint vocabulary that spans enterprise architecture, system design, and software development practice.

6.5. Evaluation Summary

This case-based evaluation demonstrates that PACT:

- Enables broader and more systematic coverage of architectural concerns without requiring exhaustive documentation.
- Improves the visibility of non-functional and cross-domain risks, particularly in security, data governance, and operations.
- Supports more focused and efficient architecture reviews, reducing iteration cycles caused by late discovery of missing concerns.

 doi:10.20944/preprints202601.0720.v1

The evaluation does not aim to prove optimality or quantitative superiority. Instead, it provides qualitative evidence—consistent with established evaluation practices for architectural frameworks [15,16]—that the taxonomy is practically usable, review-oriented, and well aligned with industrial architecture governance practices.

These results support the taxonomy's suitability as a reusable reference for both enterprise architecture and system design contexts.

7. Using the Taxonomy in Industrial Practice

While the primary contribution of this paper is a standardized and reusable taxonomy of architecture viewpoints, its value is ultimately realized through practical use in industrial settings. Prior studies have emphasized that the effectiveness of architecture documentation and viewpoints must be assessed in terms of their applicability to real projects, governance processes, and cross-team communication rather than theoretical completeness alone [14,34].

This section explains how the taxonomy can be applied in industrial practice, focusing on artifact mapping, enterprise architecture (EA) governance, and cross-team communication. Examples from AI-intensive, data-intensive, and cloud-native systems illustrate its extensibility. The taxonomy is not intended for exhaustive instantiation; viewpoints are activated selectively based on project context and risk profile, enabling focused documentation rather than increased documentation volume [14,18,25].

7.1. Fragmentation of Architecture Practices in Industrial Settings

PACT does not prescribe a development process, documentation workflow, or review procedure. Instead, it provides a reference taxonomy that can be mapped to existing artifacts, tools, and governance practices, consistent with recommendations to separate architectural concerns from specific notations and methods [1,11].

In industrial practice, architecture work rarely exists within a single coherent framework. Empirical studies of enterprise and software architecture practice consistently report that organizations operate across multiple, partially disconnected architectural perspectives, each addressing different concerns, stakeholders, and time horizons [6,7,14].

At the enterprise level, enterprise architecture (EA) frameworks such as TOGAF emphasize long-term alignment, capability planning, organizational governance, and cross-domain consistency [4,6]. These perspectives are essential for strategic decision-making and portfolio-level coordination, but they often remain abstract and loosely connected to concrete system and software design activities [7].

At the standards level, ISO/IEC/IEEE 42010 provides a rigorous conceptual foundation for architecture descriptions by defining stakeholders, concerns, viewpoints, and views [1]. However, the standard deliberately avoids prescribing concrete viewpoint sets. As a result, while it offers conceptual clarity, it does not resolve how architectural concerns should be consistently instantiated, compared, or reused across projects—an issue noted in practice-oriented analyses of standards-based architecture documentation [14,18].

At the system and software development level, solution architecture and software architecture approaches—including the C4 model and related diagramming practices—focus primarily on system structure, component decomposition, and implementation-oriented views [8,9,33]. These approaches are effective for development teams, but they typically underrepresent enterprise concerns, data governance, security boundaries, and operational accountability [10,11].

In real-world projects, these perspectives coexist rather than replace one another. A single system may be described simultaneously using:

- EA capability maps and governance artifacts [4,6],
- ISO/IEC/IEEE 42010–inspired architecture descriptions [1], and
- C4-style system and software diagrams [33].

 doi:10.20944/preprints202601.0720.v1

However, because these approaches rely on different abstractions, terminologies, and scopes, they are rarely integrated into a unified architectural description. This fragmentation leads to recurring problems that have been widely reported in both industry surveys and empirical research:

- architectural concerns are unevenly covered across documentation sets [14,37],
- assumptions made at one level (e.g., enterprise or security) are not visible at another (e.g., software design) [7,18],
- architecture reviews focus on artifact presence rather than concern coverage [17], and
- cross-project comparison and reuse remain difficult [11,14].

The viewpoint taxonomy proposed in this paper is motivated by this structural gap. Rather than introducing yet another framework, it provides a shared abstraction layer of architecture viewpoints that can be applied consistently across enterprise architecture, IT solution design, and software development architecture. By decoupling architectural concerns from specific methodologies, notations, or tools, the taxonomy enables otherwise fragmented practices to be aligned, compared, and reviewed within a common conceptual structure [1,18].

7.2. Supporting Enterprise Architecture Governance and Reviews

In EA governance, architecture reviews often suffer from inconsistent expectations, implicit criteria, and subjective judgment, as documented in studies of architecture evaluation and governance practice [17,19].

The taxonomy provides a shared reference frame that can be used to structure governance processes without prescribing a fixed documentation set.

Typical applications include:

- **Review scoping:** Selecting a subset of relevant viewpoints based on project type, scale, or risk profile, consistent with risk-driven architecture evaluation principles [17].
- **Expectation alignment:** Making explicit which viewpoints must be addressed for a given review stage, improving transparency for project teams and reviewers [6,7].
- **Gap identification:** Identifying missing or weakly covered viewpoints during reviews, independent of specific diagram formats or modeling tools [11,14].

By grounding governance discussions in viewpoints rather than artifacts, EA teams can focus reviews on architectural concerns and decision quality, rather than on document completeness or notation preferences—a shift recommended in architecture evaluation literature [17,18].

7.3. Enabling Cross-Team Communication

Large organizations typically involve multiple teams with different vocabularies, responsibilities, and priorities. Prior work on architecture knowledge and documentation has shown that the lack of a shared architectural vocabulary is a major barrier to effective communication and reuse [15,18].

The taxonomy provides a common language for discussing architecture across domains such as business, development, data, security, and operations:

- Business stakeholders can focus on capability, process, and ownership viewpoints [6].
- Development teams can reason about container, component, integration, and processing viewpoints [8,9].
- Security and compliance teams can focus on risk, control, authentication, and data governance viewpoints [26,28].

Because each viewpoint is defined using a consistent template, teams can communicate architectural intent and assumptions more precisely, reducing misinterpretation, coordination overhead, and rework—an outcome aligned with findings in architecture knowledge management research [15,18].

 doi:10.20944/preprints202601.0720.v1

7.4. Extensibility for AI-Intensive Systems

AI-intensive systems introduce architectural concerns such as model lifecycle management, data lineage, feedback loops, and operational risk. These concerns have been widely discussed in recent software engineering for machine learning literature [38–40].

Such concerns can be addressed within PACT by extending or specializing existing viewpoints rather than redefining the taxonomy:

- Extending the Data Evolution / Derivation View to include feature engineering and model training pipelines [24,38].
- Specializing the Data Lineage View to trace training data, model versions, and inference outputs [39].
- Combining Threat & Risk Modeling and Monitoring & Observability Views to reason about model drift and bias detection [27,40].

This demonstrates how the taxonomy supports emerging system types while avoiding the fragmentation that has been observed in ad hoc AI architecture documentation practices [40]. These extensions reuse and specialize existing core viewpoints rather than introducing isolated AI-specific views, preserving conceptual consistency while accommodating emerging system types [21,23].

7.5. Applicability to Data-Intensive Systems

For data-intensive systems, architectural reasoning requires explicit separation between:

- business-level data flows,
- technical data pipelines, and
- governance and compliance concerns.

This separation aligns with established principles in data-intensive system design [24,26]. By selecting and combining relevant data and integration viewpoints, teams can reason about data quality, ownership, and regulatory compliance alongside system functionality. Such structured documentation has been shown to improve traceability and auditability without excessive documentation overhead [24,28].

7.6. Adoption in Cloud-Native Architectures

Cloud-native systems emphasize dynamic deployment, elasticity, and operational resilience. These characteristics are well documented in contemporary microservice and container-based architecture literature [20–22].

The taxonomy accommodates these characteristics through viewpoints addressing deployment location, infrastructure composition, resource allocation, reliability, and observability. Organizations can incrementally adopt these viewpoints to:

- reason about deployment strategies across regions and environments [22],
- align operational responsibilities with architectural decisions [21], and
- support continuous evolution without destabilizing governance structures [30,31].

7.7. Summary

The taxonomy is designed to be adopted incrementally, mapped flexibly to existing artifacts, and extended systematically. By focusing on architectural concerns rather than documentation formats, it supports EA governance, architecture reviews, and cross-team communication across a wide range of system types. This practical applicability reinforces the taxonomy's role as a reusable reference for both industrial practice and architectural research [14,18,34].

8. Discussion and Implications (With Citations)

This section discusses the implications of the proposed viewpoint taxonomy in terms of coverage, standards alignment, evolution potential, and its value for future research and practice.

 doi:10.20944/preprints202601.0720.v1

Beyond presenting a static catalog, the taxonomy is intended as a durable architectural reference that supports both industrial application and academic reuse. By grounding the taxonomy in established standards such as ISO/IEC/IEEE 42010 [1], classical viewpoint models (e.g., Kruchten's 4+1 view model [8]), and contemporary industrial practices [14,34], PACT is positioned as a reference taxonomy rather than a prescriptive framework or development method.

The evaluation highlights a structural gap that has been repeatedly observed in industrial practice and empirical studies. Enterprise architecture frameworks emphasize abstraction, long-term alignment, and organizational governance [4,6], while software architecture approaches prioritize system structure and implementation guidance [9,10]. In parallel, solution-level technical architectures attempt to address integration and deployment concerns, often without a consistent abstraction model [14]. Existing frameworks such as TOGAF [4], ISO/IEC/IEEE 42010 [1], and C4 [33] each address parts of this spectrum, but they are rarely integrated into a coherent, end-to-end architectural documentation approach [7,14,18].

The proposed viewpoint taxonomy does not replace these frameworks. Instead, it provides a unifying abstraction layer that enables enterprise architecture concerns, IT solution design, and software development architecture to be expressed, compared, and reviewed within a consistent viewpoint structure. By decoupling architectural concerns from specific notations or methodologies, the taxonomy supports long-term enterprise alignment while remaining practical for system design and software development activities, an approach consistent with recommendations in architecture knowledge and documentation research [11,18].

8.1. Coverage Across Architectural Concerns

A central design objective of the taxonomy is integrated yet structured coverage of architectural concerns encountered in modern software-intensive systems. The 52 viewpoints span eight abstraction layers, covering:

business and organizational concerns, including capabilities, processes, ownership, and decision structures [6];

application and technical architecture, including system boundaries, functional decomposition, containers, components, and technology stacks [8,9];

integration and processing behavior, capturing interactions, workflows, and execution semantics [10,21];

data architecture, addressing data flows, evolution, lineage, quality, governance, and sovereignty [24,26];

security and risk, including authentication, threat modeling, and attack paths [27,29];

deployment and infrastructure, encompassing location, networking, resource allocation, and runtime environments [22];

governance, operations, and reliability, covering operational processes, resilience, observability, and architectural governance [17,21].

This breadth directly addresses a common shortcoming in existing architecture documentation practices, where certain domains—particularly data governance, security, and operations—are underrepresented or treated implicitly [14,37]. By making these concerns first-class viewpoints, the taxonomy enables more balanced and complete architectural reasoning across heterogeneous systems.

The resulting set of 52 viewpoints reflects consolidation rather than enumeration: viewpoints were derived by synthesizing recurring architectural concerns observed across standards-based architecture descriptions, established viewpoint models, and industrial practice, rather than by exhaustively cataloging all possible perspectives [1,6,14].

8.2. Alignment with ISO/IEC/IEEE 42010

The taxonomy is explicitly aligned with the conceptual model defined in ISO/IEC/IEEE 42010 [1]. In particular:

 doi:10.20944/preprints202601.0720.v1

each viewpoint is defined independently of specific notations or artifacts, consistent with the standard's separation between viewpoints and views [1];

viewpoints are framed around stakeholder concerns, ensuring traceability between architectural representation and decision needs [1,11];

the taxonomy supports multiple views per viewpoint and multiple viewpoints per view, preserving flexibility in architectural description [8,11].

While ISO/IEC/IEEE 42010 deliberately avoids prescribing a fixed set of viewpoints, this paper complements the standard by providing a concrete, reusable instantiation of its abstract concepts. The taxonomy can thus be interpreted as a reference library of viewpoints that operationalizes the standard without constraining architectural freedom, addressing a gap noted in practice-oriented analyses of standards-based architecture documentation [14,18].

8.3. Evolution and Extensibility Potential

The taxonomy is designed to evolve alongside architectural practice. Its extensibility is supported by several structural properties:

template consistency, enabling new viewpoints to be added without redefining foundational semantics [11];

layered organization, allowing emerging concerns to be integrated without disrupting existing viewpoints [6,10];

composable viewpoints, supporting specialization and domain adaptation [18].

Emerging technologies such as AI-driven systems, platform ecosystems, and cyber-physical systems introduce new architectural concerns that have proven difficult to capture using traditional software-centric views [38–40]. These concerns can be accommodated within PACT by refining existing viewpoints (e.g., data evolution, runtime interaction, or risk modeling) or by introducing specialized viewpoints that inherit the same conceptual structure. This supports controlled evolution rather than the ad hoc proliferation of diagrams and concepts reported in prior studies [14,40].

8.4. Implications for Research

From a research perspective, the taxonomy provides a stable reference point for:

comparative studies of architectural practices across domains and organizations [37];

empirical investigations into which viewpoints are most effective in exposing specific classes of architectural risk [17,18];

longitudinal studies examining how architectural concerns evolve over time [30,31].

Because the taxonomy is explicitly aligned with an international standard and defined using a consistent template, it supports cumulative research and facilitates comparison across studies. Future work can reference, extend, or specialize subsets of the taxonomy without redefining foundational concepts, addressing a long-standing challenge in architecture research [11,18].

8.5. Implications for Practice

For practitioners, the taxonomy offers:

a shared vocabulary for cross-team and cross-discipline communication [15,18];

a structured basis for architecture reviews and governance discussions [17,19];

a reusable reference for onboarding, training, and documentation standardization [14,34].

Importantly, the taxonomy does not prescribe how architecture should be documented but clarifies what concerns should be considered. This distinction enables organizations to adapt the taxonomy to their tools, processes, and maturity levels while maintaining conceptual consistency, a key recommendation in industrial architecture practice literature [6,7,14].

 doi:10.20944/preprints202601.0720.v1

8.6. Balancing Documentation Effort and Architectural Value

A key practical challenge in architecture documentation is achieving an appropriate balance between effort invested and value delivered. Prior studies have shown that exhaustive documentation can impose significant production and review overhead without proportional benefit [14,17]. PACT addresses this challenge by coupling viewpoints with a value-driven artifact strategy, prioritizing artifacts with high analytical leverage and cross-viewpoint applicability [11,18].

By explicitly recognizing the cost of architectural documentation, the taxonomy encourages focused, risk-proportionate artifacts that support architectural decision-making without unnecessary burden, consistent with principles of architecture evaluation and governance [17,19].

8.7. Citation and Reference Value

As an integrated and standardized catalog of architecture viewpoints, the taxonomy is intended to function as a reference artifact for both research and practice. Its value lies not only in immediate application, but also in its potential to be cited and reused as:

a baseline for future viewpoint- or domain-specific studies [37];

a reference framework for evaluating architecture documentation completeness [14,18];

a conceptual bridge between architectural theory and industrial practice [1,11].

By providing coverage, structure, and explicit alignment with established standards, this work contributes a durable and evolvable reference to the architecture literature, addressing long-standing calls for reusable architectural abstractions in both academia and industry [11,14,18]. Rather than introducing new architectural concepts, the contribution of PACT lies in the systematic unification and normalization of viewpoints across frameworks that traditionally operate at different abstraction levels and are rarely integrated in practice [1,11,12,18].

9. Limitations and Future Work

Despite its broad and structured coverage, PACT has several limitations that should be explicitly acknowledged.

First, the taxonomy is **conceptual rather than prescriptive**. While it defines which architectural concerns can be represented, it does not mandate which viewpoints must be used in a given project or how they should be combined. This design choice intentionally prioritizes flexibility, reuse, and cross-context applicability, consistent with the principles underlying ISO/IEC/IEEE 42010 [1]. However, as observed in prior studies on architecture frameworks, non-prescriptive reference models may require complementary organizational guidance to ensure consistent adoption in practice [6,14]. PACT is intentionally tool-agnostic, allowing it to be mapped onto existing documentation tools and practices; dedicated tool support is therefore treated as an extension opportunity rather than a prerequisite for applicability [12,18,37].

Second, although the taxonomy spans a broad range of concerns commonly encountered in modern software-intensive systems, it does not claim completeness for all possible domains. Specialized areas such as safety-critical systems, real-time embedded systems, and highly regulated domain-specific architectures often introduce concerns and constraints that require dedicated viewpoints or specialized extensions [41,42]. Similar limitations have been reported in other general-purpose architecture frameworks, which are typically extended or profiled for domain-specific use rather than applied unchanged [4,6].

Third, the taxonomy is currently **documentation-oriented**, focusing on viewpoints and their conceptual intent rather than on executable or code-level representations. While this abstraction is appropriate for architectural reasoning and governance, contemporary software practices increasingly emphasize infrastructure-as-code, continuous delivery, and automated compliance checking [21,22]. Bridging the gap between conceptual architecture viewpoints and runtime or code-level artifacts remains an open challenge in architecture research and practice [17,30].

 doi:10.20944/preprints202601.0720.v1

These limitations suggest several directions for future work. In particular, the taxonomy could be extended and strengthened through:

- **empirical studies** evaluating viewpoint selection, coverage, and reuse across different domains and organizational contexts, building on prior empirical architecture research [18,37];
- **tool support** for viewpoint selection, consistency checking, and traceability across views and artifacts, as advocated in architecture knowledge management literature [11,17];
- **integration with architectural decision records (ADRs)** to explicitly link viewpoints with design rationale and trade-off decisions [19,32];
- **periodic evolution and profiling** to accommodate emerging technologies, regulatory requirements, and architectural paradigms, following practices recommended for long-lived reference frameworks [6,14].

Taken together, these limitations do not undermine the core contribution of PACT as a reference taxonomy. Rather, they reflect deliberate design choices aligned with its intended role: providing broad and structured concern coverage (Section 8.1), operational alignment with ISO/IEC/IEEE 42010 (Section 8.2), and controlled extensibility over time (Section 8.3), without prescribing methods, tools, or artifacts [1,6,14].

By explicitly separating stable viewpoint abstractions from evolving practices and technologies, PACT is positioned to support both industrial governance and cumulative research. The identified directions for future work therefore represent natural extensions of the taxonomy's reference-oriented design, reinforcing its long-term value as an evolvable architectural knowledge base rather than a fixed documentation standard [18,37].

10. Summary and Conclusion

This paper presented a standardized and reusable taxonomy of architecture viewpoints that unifies and normalizes architectural concerns across enterprise, system, and software architecture levels. By defining 52 viewpoints using a unified template and organizing them into a coherent abstraction structure, the taxonomy addresses long-standing challenges in architectural documentation, including inconsistent coverage, poor comparability, and limited reuse [14,37].

As discussed in Section 8, this coverage explicitly spans business, application, integration, data, security, infrastructure, and operational concerns, elevating cross-cutting domains such as data governance and security to first-class architectural viewpoints.

Aligned with the ISO/IEC/IEEE 42010 conceptual model, the taxonomy operationalizes the notion of viewpoints without constraining architectural freedom [1]. Rather than replacing existing standards or frameworks, PACT instantiates the standard's abstract concepts through a reusable reference taxonomy, enabling consistent viewpoint selection and review across heterogeneous systems and organizational contexts. Positioning PACT explicitly as a taxonomy rather than a framework is a deliberate design choice to maximize reuse, longevity, and compatibility with existing methods, avoiding methodological lock-in while supporting cumulative research and practice [6,14,37].

From an industrial perspective, the taxonomy provides a shared viewpoint abstraction that supports selective adoption, structured reviews, and cross-team communication without increasing documentation burden. From a research perspective, it establishes a stable baseline for comparative studies, empirical evaluation, and systematic extension. Its layered organization and template-driven structure support controlled evolution, allowing emerging concerns—such as those found in AI-intensive or data-centric systems—to be incorporated without fragmenting the architectural description.

Importantly, this work positions architecture viewpoints not as ad hoc diagrams, but as reusable analytical instruments grounded in stakeholder concerns. As such, PACT positions architecture viewpoints as reusable analytical abstractions rather than ad hoc diagrams, offering a durable

 doi:10.20944/preprints202601.0720.v1

reference that can be cited, extended, and adapted as architectural practices and system types continue to evolve [1,14,37].

Funding: This research received no external funding.

Acknowledgments: The author would like to thank the anonymous reviewers for their valuable comments.

Appendix A

Extended Viewpoint Definitions for the PACT Taxonomy

This appendix provides the **normative extended definitions** of all architecture viewpoints included in PACT (Practical Architecture Viewpoint Taxonomy).

Each viewpoint elaborates the corresponding entry summarized in **Table 1** and is defined using the **unified viewpoint definition template** introduced in **Section 4**.

The purpose of this appendix is to complement Section 5.2 by providing additional **intent, rationale, and usage guidance** for each viewpoint, thereby supporting consistent interpretation, comparison, and long-term reuse across projects and organizations.

No additional viewpoints, attributes, or classification dimensions are introduced in this appendix beyond those defined in the core taxonomy.

Viewpoints are defined **independently of architecture artifacts**, in accordance with the conceptual separation established by ISO/IEC/IEEE 42010.

Architecture artifacts (e.g., diagrams, matrices, catalogs) are specified separately and mapped to viewpoints through a consolidated mapping matrix in **Appendix C**.

This separation avoids over-specification, preserves methodological neutrality, and supports the controlled evolution of both viewpoints and artifacts over time.

Viewpoint Template (Applied Consistently)

Each viewpoint is described using the following fields:

- Viewpoint ID
- Viewpoint Name
- Primary Concern
- Intent / Rationale
- Key Questions Answered
- Primary Stakeholders
- Notes on Usage / Common Pitfalls

All fields correspond directly to the unified viewpoint definition template defined in Section 4.

A.1 Business / Organization Layer (B)

This section provides extended definitions for the Business / Organization viewpoints listed in Table 1 (B1–B5).

B1 — Business Capability View

- **Primary Concern**: Business capability structure
- **Intent / Rationale**:
- To provide a stable, technology-independent representation of what the organization is able to do. Capabilities serve as a long-term anchor for alignment across business, IT, and strategy.
- **Key Questions Answered**:
- What capabilities exist? Which are core or differentiating? Where are investment priorities?
- **Primary Stakeholders**:

 doi:10.20944/preprints202601.0720.v1

- Executives, Enterprise Architects
- **Notes**:
- Should remain stable across system redesigns; avoid mixing with process sequencing.

B2 — Business Process View

- **Primary Concern**: End-to-end business processes
- **Intent / Rationale**:
- To describe how capabilities are realized through ordered activities and decisions.
- **Key Questions Answered**:
- How does work flow across roles and systems?
- **Primary Stakeholders**:
- Product, Operations
- **Notes**:
- Focus on business intent, not technical orchestration.

B3 — User / Actor View

- **Primary Concern**: Users, roles, and external actors
- **Intent / Rationale**:
- To clarify who interacts with the system and in what capacity.
- **Key Questions Answered**:
- Who uses the system? Who initiates or receives interactions?
- **Primary Stakeholders**:
- Product, Security
- **Notes**:
- Distinct from authentication mechanisms (see SCR layer).

B4 — Organizational View

- **Primary Concern**: Ownership and responsibility
- **Intent / Rationale**:
- To establish accountability for systems, data, and decisions.
- **Key Questions Answered**:
- Who owns what? Who approves changes?
- **Primary Stakeholders**:
- Executives, EA
- **Notes**:
- Often missing in technical documentation, but critical for governance.

B5 — Interaction & Influence View

- **Primary Concern**: Cross-domain influence and decision coupling
- **Intent / Rationale**:
- To expose dependencies and influence paths between business domains.
- **Key Questions Answered**:
- How do decisions in one domain affect others?
- **Primary Stakeholders**:
- Product, Architects
- **Notes**:
- Helps anticipate organizational bottlenecks.

A.2 Context & Application Layer (A)

This section provides extended definitions for the Context / Application viewpoints listed in Table 1 (A1–A4).

 doi:10.20944/preprints202601.0720.v1

A1 — System Context View

- **Primary Concern**: System boundaries
- **Intent / Rationale**:
- To define what is inside the system and what lies in its environment.
- **Key Questions Answered**:
- What external systems and actors interact with the system?
- **Primary Stakeholders**:
- Product, Development
- **Notes**:
- Foundation for all downstream viewpoints.

A.2 Functional / Application View

- **Primary Concern**: Functional decomposition
- **Intent / Rationale**:
- To group system functionality independent of deployment or implementation.
- **Key Questions Answered**:
- What functions does the system provide?
- **Primary Stakeholders**:
- Product, Development
- **Notes**:
- Avoid conflating with component structure.

A3 — Application Interaction View

- **Primary Concern**: Inter-application responsibilities
- **Intent / Rationale**:
- To show how applications collaborate at a logical level.
- **Key Questions Answered**:
- How does responsibility flow across applications?
- **Primary Stakeholders**:
- Development, Product
- **Notes**:
- Often precedes technical integration design.

A4 — User Role Mapping View

- **Primary Concern**: Role-to-application mapping
- **Intent / Rationale**:
- To clarify authorization intent at a conceptual level.
- **Key Questions Answered**:
- Who can do what in which system?
- **Primary Stakeholders**:
- Product, Security
- **Notes**:
- Not a substitute for access control models.

A.3 Container / Component / Technical Service Layer (C)

This section provides extended definitions for the Container / Component / Technical Service viewpoints listed in Table 1 (C1–C5).

C1 — Container View

- **Primary Concern**: Runtime execution units
- **Intent / Rationale**:
- To describe deployable units and their responsibilities.
- **Key Questions Answered**:

 doi:10.20944/preprints202601.0720.v1

- What runs independently?
- **Primary Stakeholders**:
- Software Architects, Developers
- **Notes**:
- Logical, not physical deployment.

C2 — Component View

- **Primary Concern**: Internal structure
- **Intent / Rationale**:
- To explain how containers are decomposed internally.
- **Key Questions Answered**:
- How is complexity managed?
- **Primary Stakeholders**:
- Developers
- **Notes**:
- Avoid over-detailing early.

C3 — Technical Service View

- **Primary Concern**: Logical service responsibility
- **Intent / Rationale**:
- To define service boundaries independent of deployment.
- **Key Questions Answered**:
- What capability does each service provide?
- **Primary Stakeholders**:
- Architects, Developers
- **Notes**:
- Key for microservice clarity.

C4 — Technical Stack View

- **Primary Concern**: Technology selection
- **Intent / Rationale**:
- To make technology choices explicit and reviewable.
- **Key Questions Answered**:
- Which frameworks, platforms, and runtimes are used?
- **Primary Stakeholders**:
- Development, Operations
- **Notes**:
- Frequently required in EA reviews.

C5 — Communication / Protocol View

- **Primary Concern**: Communication mechanisms
- **Intent / Rationale**:
- To document interaction patterns and protocols.
- **Key Questions Answered**:
- How do components communicate?
- **Primary Stakeholders**:
- Development, SRE
- **Notes**:
- Critical for performance and reliability analysis.

A.4 Integration & Processing Layer (IP)

This section provides extended definitions for the Integration / Processing viewpoints listed in Table 1 (IP1–IP6).

 doi:10.20944/preprints202601.0720.v1

IP1 — Application Integration View

- **Primary Concern**: Business-level system integration
- **Intent / Rationale**:
- To describe how applications collaborate to support end-to-end business scenarios, independent of low-level technical constraints.
- **Key Questions Answered**:
- How do applications exchange information? Where are integration responsibilities assigned?
- **Primary Stakeholders**:
- Product Owners, Application Architects
- **Notes on Usage / Common Pitfalls**:
- Should focus on *responsibility and intent*, not protocol details.

IP2 — Technical Integration View

- **Primary Concern**: Technical integration constraints
- **Intent / Rationale**:
- To capture non-functional integration characteristics such as latency, throughput, and coupling.
- **Key Questions Answered**:
- What technical limitations or guarantees apply to integration?
- **Primary Stakeholders**:
- Developers, SRE
- **Notes**:
- Complements IP1; avoid duplicating business semantics.

IP3 — System Processing View

- **Primary Concern**: Execution semantics and orchestration
- **Intent / Rationale**:
- To explain how requests are processed and coordinated across system components.
- **Key Questions Answered**:
- How is work orchestrated? Where are control points?
- **Primary Stakeholders**:
- Developers, Operations
- **Notes**:
- Particularly important for distributed systems.

IP4 — Activity View

- **Primary Concern**: Action and decision flow
- **Intent / Rationale**:
- To describe logical execution paths including branching and decisions.
- **Key Questions Answered**:
- What actions occur and in what order?
- **Primary Stakeholders**:
- Architects, Developers
- **Notes**:
- Focus on logic, not timing.

IP5 — Process Flow View

- **Primary Concern**: Executable workflows
- **Intent / Rationale**:
- To represent operational workflows that can be executed or automated.
- **Key Questions Answered**:
- How does the workflow run at runtime?
- **Primary Stakeholders**:
- Product, SRE

 doi:10.20944/preprints202601.0720.v1

- **Notes**:
- Distinct from B2 (business process) in execution focus.

IP6 — Sequence Interaction View

- **Primary Concern**: Temporal interaction order
- **Intent / Rationale**:
- To make call ordering and dependencies explicit.
- **Key Questions Answered**:
- What is the exact sequence of interactions?
- **Primary Stakeholders**:
- Developers, Architects
- **Notes**:
- Use selectively to manage effort.

A.5 Data Architecture Layer (D)

This section provides extended definitions for the Data viewpoints listed in Table 1 (D1–D11).

D1 — Business Data Flow View

- **Primary Concern**: Business-level data movement
- **Intent / Rationale**:
- To show how data supports business processes.
- **Key Questions Answered**:
- How does data flow across business activities?
- **Primary Stakeholders**:
- Product, Operations
- **Notes**:
- Avoid premature technical detail.

D2 — Data Evolution / Derivation View

- **Primary Concern**: Data transformation logic
- **Intent / Rationale**:
- To explain how data is derived, enriched, or aggregated.
- **Key Questions Answered**:
- How is data transformed over time?
- **Primary Stakeholders**:
- Data Engineers, ML Engineers
- **Notes**:
- Crucial for analytics and AI systems.

D3 — Data Lineage View

- **Primary Concern**: Data traceability
- **Intent / Rationale**:
- To enable auditing and impact analysis by tracing data origins.
- **Key Questions Answered**:
- Where did this data come from?
- **Primary Stakeholders**:
- Compliance, Data Governance
- **Notes**:
- Often mandated by regulation.

D4 — Data Asset View

- **Primary Concern**: Data inventory and ownership
- **Intent / Rationale**:
- To treat data as a managed enterprise asset.
- **Key Questions Answered**:

 doi:10.20944/preprints202601.0720.v1

- What data assets exist and who owns them?
- **Primary Stakeholders**:
- Data Office
- **Notes**:
- Foundation for governance.

D5 — Data Entity View

- **Primary Concern**: Conceptual data entities
- **Intent / Rationale**:
- To define business concepts independent of storage.
- **Key Questions Answered**:
- What core entities exist?
- **Primary Stakeholders**:
- Data Architects, Business Analysts
- **Notes**:
- Should remain stable.

D6 — Data Quality View

- **Primary Concern**: Data quality characteristics
- **Intent / Rationale**:
- To expose quality expectations and metrics.
- **Key Questions Answered**:
- Is data accurate and complete?
- **Primary Stakeholders**:
- Data Stewards
- **Notes**:
- Often overlooked in architecture

D7 — Data Flow Compliance View

- **Primary Concern**: Regulatory data paths
- **Intent / Rationale**:
- To ensure compliance along data flows.
- **Key Questions Answered**:
- Is sensitive data handled correctly?
- **Primary Stakeholders**:
- Compliance, Legal
- **Notes**:
- Distinct from security controls.

D8 — Data Pipeline View

- **Primary Concern**: Technical data pipelines
- **Intent / Rationale**:
- To document execution stages and responsibilities.
- **Key Questions Answered**:
- How is data processed technically?
- **Primary Stakeholders**:
- Data Engineers
- **Notes**:
- Physical focus.

 doi:10.20944/preprints202601.0720.v1

D9 — Data Sovereignty View

- **Primary Concern**: Data residency
- **Intent / Rationale**:
- To show where data is stored geographically.
- **Key Questions Answered**:
- Where does data physically reside?
- **Primary Stakeholders**:
- EA, Compliance
- **Notes**:
- Increasingly critical globally.

D10 — Logical Data Model View

- **Primary Concern**: Logical data structure
- **Intent / Rationale**:
- To describe entity relationships.
- **Key Questions Answered**:
- How are entities related?
- **Primary Stakeholders**:
- Developers, Data Architects
- **Notes**:
- Platform-independent.

D11 — Physical Data Model View

- **Primary Concern**: Physical data implementation
- **Intent / Rationale**:
- To capture actual database schemas.
- **Key Questions Answered**:
- How is data stored?
- **Primary Stakeholders**:
- DBAs
- **Notes**:
- Most volatile data viewpoint.

A.6 Security, Compliance & Risk Layer (SCR)

This section provides extended definitions for the Security/Compliance /Risk viewpoints listed in Table 1 (SCR1–SCR7).

SCR1 — Security Control View

- **Primary Concern**: Security mechanisms and enforcement points
- **Intent / Rationale**:
- To make security controls explicit and reviewable across system boundaries.
- **Key Questions Answered**:
- What controls enforce security policies?
- **Primary Stakeholders**:
- Security Architects, DevOps
- **Notes on Usage / Common Pitfalls**:
- Should focus on *controls*, not threat enumeration (see SCR2).

 doi:10.20944/preprints202601.0720.v1

SCR2 — Risk / Threat Modeling View

- **Primary Concern**: Threat identification and risk assessment
- **Intent / Rationale**:
- To systematically analyze potential threats and vulnerabilities.
- **Key Questions Answered**:
- What threats exist and where are the risks?
- **Primary Stakeholders**:
- Security, Risk Management
- **Notes**:
- Often performed iteratively; not a one-time artifact.

SCR3 — Attack Path View

- **Primary Concern**: Adversarial traversal paths
- **Intent / Rationale**:
- To visualize how an attacker could move through the system.
- **Key Questions Answered**:
- How could an attack propagate?
- **Primary Stakeholders**:
- Security Operations, Incident Response
- **Notes**:
- Complements SCR2 by emphasizing sequence.

SCR4 — Service Authentication View

- **Primary Concern**: Service-to-service authentication
- **Intent / Rationale**:
- To document trust relationships among services.
- **Key Questions Answered**:
- How do services authenticate each other?
- **Primary Stakeholders**:
- Security, Developers
- **Notes**:
- Distinct from user authentication.

SCR5 — User Authentication View

- **Primary Concern**: User authentication mechanisms
- **Intent / Rationale**:
- To clarify how users are authenticated and identity is established.
- **Key Questions Answered**:
- How are users authenticated?
- **Primary Stakeholders**:
- Security, Product
- **Notes**:
- Authorization concerns belong to A4 / Governance.

SCR6 — Compliance Boundary View

- **Primary Concern**: Regulatory and compliance boundaries
- **Intent / Rationale**:
- To explicitly represent regulatory zones and constraints.
- **Key Questions Answered**:

 doi:10.20944/preprints202601.0720.v1

- Which regulations apply where?
- **Primary Stakeholders**:
- Compliance, Legal
- **Notes**:
- Often implicit; making it explicit reduces risk.

SCR7 — Security Monitoring View

- **Primary Concern**: Security observability
- **Intent / Rationale**:
- To ensure security-relevant events are observable.
- **Key Questions Answered**:
- How are security incidents detected?
- **Primary Stakeholders**:
- Security Operations
- **Notes**:
- Bridges security and operations layers

A.7 Architecture Governance & Decision Layer (AGD)

This section provides extended definitions for the Architecture Governance / Decision viewpoints listed in Table 1 (AGD1–AGD5).

Note:

This layer captures *meta-architectural* concerns and intentionally sits above functional and technical layers.

It enables long-term alignment, traceability, and evolution across organizational and technical boundaries.

AGD1 — Architecture Governance View

- **Primary Concern**: Governance structures and rules
- **Intent / Rationale**:
- To define how architectural decisions are guided and constrained.
- **Key Questions Answered**:
- How are architectural decisions governed?
- **Primary Stakeholders**:
- Executives, Enterprise Architects
- **Notes on Usage / Common Pitfalls**:
- Should not prescribe implementation details.

AGD2 — Architecture Decision View

- **Primary Concern**: Architectural decision records
- **Intent / Rationale**:
- To document significant architectural decisions and their rationale.
- **Key Questions Answered**:
- Why was this design chosen?
- **Primary Stakeholders**:
- Architects, Development Leads
- **Notes**:
- Critical for long-term system understanding.

 doi:10.20944/preprints202601.0720.v1

AGD3 — Evolution / Change Impact View

- **Primary Concern**: Architectural evolution
- **Intent / Rationale**:
- To assess the impact of change across viewpoints.
- **Key Questions Answered**:
- What breaks if something changes?
- **Primary Stakeholders**:
- Architects, Program Management
- **Notes**:
- Often missing until late-stage failures occur.

AGD4 — Quality Attribute Trade-off View

- **Primary Concern**: Quality trade-offs
- **Intent / Rationale**:
- To explicitly reason about competing quality attributes.
- **Key Questions Answered**:
- Which qualities are prioritized and why?
- **Primary Stakeholders**:
- Architects, Product Owners
- **Notes**:
- Prevents implicit trade-offs.

AGD5 — Cost & Value View

- **Primary Concern**: Economic implications
- **Intent / Rationale**:
- To relate architectural choices to cost and value.
- **Key Questions Answered**:
- What is the cost/value impact of decisions?
- **Primary Stakeholders**:
- Executives, Finance
- **Notes**:
- Bridges architecture and business strategy.

A.8 Deployment / Infrastructure / Networking Layer (DIN)

This section provides extended definitions for the Deployment / Infrastructure / Networking viewpoints listed in Table 1 (DIN1–DIN4).

DIN1 — Application Location View

- **Primary Concern**: Deployment location and environment distribution
- **Intent / Rationale**:
- To clarify where systems and components are deployed across environments and regions.
- **Key Questions Answered**:
- Where are systems deployed?
- **Primary Stakeholders**:
- Enterprise Architects, Infrastructure Teams
- **Notes on Usage / Common Pitfalls**:
- Should distinguish logical deployment intent from physical realization.

 doi:10.20944/preprints202601.0720.v1

DIN2 — Network Topology View

- **Primary Concern**: Network structure and segmentation
- **Intent / Rationale**:
- To make network boundaries and communication paths explicit.
- **Key Questions Answered**:
- How is the network structured?
- **Primary Stakeholders**:
- Network Engineers, Infrastructure Architects
- **Notes**:
- Often under-documented or conflated with application diagrams.

DIN3 — Deployment / Infrastructure View

- **Primary Concern**: Infrastructure composition
- **Intent / Rationale**:
- To document runtime infrastructure components and dependencies.
- **Key Questions Answered**:
- What infrastructure components exist?
- **Primary Stakeholders**:
- DevOps, Operations
- **Notes**:
- Should evolve with infrastructure-as-code artifacts.

DIN4 — Resource Allocation View

- **Primary Concern**: Capacity and scaling
- **Intent / Rationale**:
- To reason about resource provisioning and scalability.
- **Key Questions Answered**:
- What resources are allocated and how do they scale?
- **Primary Stakeholders**:
- SRE, Infrastructure Teams
- **Notes**:
- Enables proactive capacity planning.

A.9 Operations & Reliability Layer (OR)

This section provides extended definitions for the Operations/Reliability viewpoints listed in Table 1 (OR1–OR5).

OR1 — Operations View

- **Primary Concern**: Operational processes
- **Intent / Rationale**:
- To describe how systems are operated and supported.
- **Key Questions Answered**:
- How are incidents and routine operations handled?
- **Primary Stakeholders**:
- Operations Teams, DevOps
- **Notes on Usage / Common Pitfalls**:
- Often treated as out-of-scope in architecture documentation.

 doi:10.20944/preprints202601.0720.v1

OR2 — Incident & Escalation View

- **Primary Concern**: Incident handling and escalation paths
- **Intent / Rationale**:
- To clarify responsibilities during failures.
- **Key Questions Answered**:
- Who responds when things go wrong?
- **Primary Stakeholders**:
- Operations, Support
- **Notes**:
- Critical for high-availability systems.

OR3 — Resilience / Disaster Recovery View

- **Primary Concern**: Recovery strategies
- **Intent / Rationale**:
- To ensure systems can recover from failures.
- **Key Questions Answered**:
- What are RPO/RTO targets and recovery mechanisms?
- **Primary Stakeholders**:
- Infrastructure, Executives
- **Notes**:
- Should align with business continuity planning.

OR4 — Reliability / SLO View

- **Primary Concern**: Reliability objectives
- **Intent / Rationale**:
- To define and monitor reliability targets.
- **Key Questions Answered**:
- What SLOs apply?
- **Primary Stakeholders**:
- SRE, Development Teams
- **Notes**:
- Links architecture decisions to operational outcomes.

OR5 — Monitoring / Observability View

- **Primary Concern**: System visibility
- **Intent / Rationale**:
- To ensure system behavior is observable at runtime.
- **Key Questions Answered**:
- How is system health observed?
- **Primary Stakeholders**:
- DevOps, SRE
- **Notes**:
- Complements OR4 by providing evidence.

A.10 Appendix Summary

Appendix A serves as a normative definition library for the PACT taxonomy.

It elaborates the intent and usage of each viewpoint without prescribing specific architecture artifacts or notations.

 doi:10.20944/preprints202601.0720.v1

Artifact types and value-driven artifact selection are addressed separately in Appendix B and Appendix C.

This separation preserves conceptual clarity and supports independent evolution of viewpoints and artifacts, in alignment with ISO/IEC/IEEE 42010.

Appendix B. Artifact Mapping (Industrial-Oriented, Low-Risk)

This appendix provides a pragmatic mapping between the proposed architecture viewpoints and commonly used architecture artifacts in industrial practice. The purpose of this appendix is not to prescribe new documentation requirements, but to **reduce adoption risk** by showing how the taxonomy aligns with artifacts that organizations already produce.

The mapping is intentionally **lightweight, optional, and incremental**, enabling organizations to adopt the taxonomy without disrupting existing processes, tools, or governance models.

B.1 Design Principles for Artifact Mapping

The artifact mapping follows four principles:

1. **Non-intrusive**: No new mandatory artifacts are introduced.
2. **Tool-agnostic**: The mapping does not depend on specific tools (e.g., ArchiMate, UML, C4).
3. **Process-neutral**: Compatible with waterfall, agile, DevOps, and hybrid delivery models.
4. **Incremental adoption**: Organizations may start with a subset of viewpoints and artifacts.

Each viewpoint may map to one or more existing artifacts, and a single artifact may partially address multiple viewpoints.

B.2 Viewpoint-to-Artifact Mapping (Representative, Value-Driven)

To support practical adoption without imposing excessive documentation effort, this appendix presents a **value-driven mapping between architecture viewpoints and commonly used architecture artifact types**.

Rather than prescribing an exhaustive or mandatory documentation set, the mapping reflects **industrial practice-oriented trade-offs between analytical value and modeling effort**. Each artifact type represents a cost-effective deliverable that can support multiple viewpoints defined in the PACT taxonomy.

Table 2. Architecture Artifact Catalog (with IDs).

Artifact ID	Artifact Name	Purpose (High-Level)	Typical Contents	Supported Viewpoints (Representative)	Effort vs. Value
A1	Technical Architecture Diagram	Shows major technical components and their relationships	Services, modules, logical boundaries	Technical Service, Container, Application Interaction	⭐⭐⭐☆☆ high value, moderate effort
A2	Application Collaboration Diagram	Illustrates communication relationships between systems/services	Nodes, links, directionality	Application Interaction, Integration	⭐⭐⭐☆☆ high value, moderate effort
A3	Business Process Diagram	Represents business workflows and decision points	Activities, tasks, roles	Business Process, Governance	⭐⭐☆☆☆ moderate value, low effort
A4	Data Model Diagram	Defines key data entity relationships	Entities, relationships	Logical Data Model, Data Architecture	⭐⭐⭐☆☆ high value, moderate effort

 doi:10.20944/preprints202601.0720.v1

A5	Data Pipeline Diagram	Shows how data moves and transforms	Sources, ETL/ELT stages	Data Flow, Data Lineage, Data Evolution	⭐⭐⭐☆☆ high value, moderate effort
A6	Auth & Service Flow Diagram	Maps authentication, authorization, and service invocation	Identity providers, tokens, trust boundaries	User Authentication, Service Authentication, Security Control	⭐⭐⭐⭐☆ very high value, moderate effort
A7	System Process Flow Diagram	Shows runtime orchestration and execution paths	Sync/async calls, retries, errors	System Processing, Operations, Resilience	⭐⭐⭐⭐☆ very high value, moderate effort
A8	Resource List	Provides inventory of systems, owners, and environments	System names, ownership, environments	Deployment, Infrastructure, Governance	⭐⭐☆☆☆ moderate value, low effort
A9	Data Asset Matrix	Maps data assets to value, ownership, and sensitivity	Assets, classifications, consumers	Data Asset, Compliance, Data Governance	⭐⭐⭐☆☆ high value, moderate effort
A10	Data Compliance Diagram	Shows regulatory and control boundaries for data	Regulatory zones, control points	Data Compliance, Sovereignty	⭐⭐☆☆☆ moderate value, low effort

Note: Supported viewpoints are representative. A complete and systematic mapping between Viewpoint IDs and Artifact IDs is provided in **Appendix C**.

Artifact Selection Principles (Value-Driven Heuristics)

To ensure that architectural documentation effort remains proportionate to project risk and complexity, the following principles guide artifact selection:

5. Start with Core Artifacts
6. Artifacts such as A1 (Technical Architecture Diagram), A6 (Auth & Service Flow Diagram), and A7 (System Process Flow Diagram) provide high analytical leverage and cover multiple structural and behavioral concerns.
7. Add Data-Centric Artifacts Only When Data Is a Dominant Concern
8. A4 (Data Model Diagram) and A5 (Data Pipeline Diagram) should be prioritized when data movement, derivation, lineage, or compliance risks are central.
9. Use Lightweight Artifacts for Governance and Compliance Contexts
10. A8 (Resource List) and A10 (Data Compliance Diagram) expose ownership, responsibility, and regulatory boundaries with minimal modeling overhead.
11. Avoid Redundant Artifacts
12. When multiple artifacts convey overlapping concerns, teams should prioritize the artifact with higher analytical value for the target stakeholders rather than maximizing diagram count.

These principles reinforce the taxonomy's intent: **to guide architectural reasoning and review, not to enforce a checklist-driven documentation process**.

B.3 Typical Adoption Patterns in Industry

B.3.1 Minimal Adoption (Low Maturity)

- Use 5–8 core viewpoints (e.g., Capability, Application Landscape, Deployment, Security).
- Reuse existing diagrams without modification.
- Primary value: improved **review consistency** and **shared vocabulary**.

B.3.2 Selective Adoption (Medium Maturity)

- Introduce viewpoint-based labeling of artifacts.
- Use viewpoints to structure architecture review checklists.
- Primary value: improved **comparability** across projects.

 doi:10.20944/preprints202601.0720.v1

B.3.3 Systematic Adoption (High Maturity)

- Maintain a viewpoint catalog aligned with EA governance.
- Link viewpoints to architecture decisions (ADR).
- Primary value: **traceability**, **reuse**, and **long-term evolution**.

B.4 Support for Enterprise Architecture Governance

The artifact mapping supports EA governance by:

- Providing a **common reference model** for architecture reviews.
- Enabling reviewers to assess **coverage gaps** explicitly.
- Reducing subjective interpretation of "complete architecture documentation."

Rather than enforcing artifact quantity, the taxonomy shifts governance focus toward **concern coverage and stakeholder alignment**.

B.5 Mapping to ISO/IEC/IEEE 42010 Concepts

The artifact mapping is aligned with ISO/IEC/IEEE 42010 as follows:

- **Viewpoints** define the concerns to be addressed.
- **Artifacts** serve as concrete representations of views.
- Multiple artifacts may instantiate the same viewpoint.

This reinforces compliance with the standard while preserving flexibility in documentation practices.

B.6 Risk Mitigation and Practical Considerations

To minimize industrial adoption risk:

- Viewpoints are **recommendatory**, not mandatory.
- Artifact mapping is **context-sensitive**, not prescriptive.
- Organizations may omit viewpoints that are irrelevant to a given system.

This ensures the taxonomy remains practical for large enterprises, regulated industries, and fast-moving product teams alike.

Appendix C. Viewpoint–Artifact Mapping Matrix

This appendix provides a consolidated mapping between the PACT viewpoints and the controlled set of architecture artifact types defined in Appendix B.

The matrix clarifies which artifact types are typically used to realize or support each viewpoint, without coupling viewpoint definitions to specific notations or documentation formats.

Consistent with ISO/IEC/IEEE 42010, viewpoints are defined independently of views and artifacts.

The mapping is indicative rather than prescriptive, reflecting common industrial practice while allowing adaptation to local tools and standards.

Table C1. Viewpoint–Artifact Mapping Matrix.

Business / Organization Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
B1 Business Capability View	✓		✓							
B2 Business Process View			✓				✓			
B3 User / Actor View		✓				✓				
B4 Organizational View								✓		
B5 Interaction & Influence View	✓	✓								

 doi:10.20944/preprints202601.0720.v1

Context & Application Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
A1 Context View	✓	✓								
A2 Functional / Application View	✓									
A3 Application Interaction View		✓					✓			
A4 User Role View		✓				✓				

Container / Component / Technical Service Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
C1 Container View	✓									
C2 Component View	✓									
C3 Technical Service View	✓	✓								
C4 Technical Stack View	✓							✓		
C5 Communication / Protocol View	✓	✓					✓			

Integration & Processing Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
IP1 Application Integration View		✓					✓			
IP2 Technical Integration View	✓	✓								
IP3 System Processing View							✓			
IP4 Activity View			✓				✓			
IP5 Process Flow View			✓				✓			
IP6 Sequence Diagram View		✓					✓			

Data Architecture Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
D1 Business Data Flow View			✓							
D2 Data Evolution / Derivation View					✓					
D3 Data Lineage View					✓				✓	✓
D4 Data Asset View									✓	
D5 Data Entity View				✓						
D6 Data Quality View									✓	
D7 Data Flow Compliance View										✓
D8 Data Pipeline View					✓					
D9 Data Sovereignty View										✓
D10 Logical Data Model View				✓						
D11 Physical Data Model View				✓						

Security, Compliance & Risk Layer (SCR)										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
GCS1 Architecture Governance View	✓							✓		
GCS2 Security Control View	✓					✓				
GCS3 Risk / Threat Modeling View	✓					✓				✓
GCS4 Attack Path View	✓						✓			
GCS5 Service Authentication View						✓				
GCS6 User Authentication View						✓				
GCS7 Compliance View										✓

Deployment / Infrastructure / Networking Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
DIN1 Application Location View	✓							✓		
DIN2 Network Topology View	✓									
DIN3 Deployment / Infrastructure View	✓							✓		
DIN4 Resource View								✓		

 doi:10.20944/preprints202601.0720.v1

Operations & Reliability Layer										
Viewpoint ID	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
OR1 Operations View							✓	✓		
OR2 Resilience / DR View	✓							✓		
OR3 Reliability / SLO View								✓		
OR4 Monitoring / Observability View							✓	✓		
OR5 Incident & Recovery View							✓	✓		

References

1. ISO/IEC/IEEE, ISO/IEC/IEEE 42010:2011 – Systems and Software Engineering — Architecture Description, ISO/IEC/IEEE, 2011.
2. ISO/IEC, ISO/IEC 12207:2017 – Systems and Software Engineering — Software Life Cycle Processes, ISO/IEC, 2017.
3. ISO/IEC, ISO/IEC 15288:2015 – Systems and Software Engineering — System Life Cycle Processes, ISO/IEC, 2015.
4. The Open Group, TOGAF® Standard, 10th Edition, The Open Group, 2022.
5. J. A. Zachman, "A Framework for Information Systems Architecture," IBM Systems Journal, vol. 26, no. 3, pp. 276–292, 1987.
6. M. Lankhorst et al., Enterprise Architecture at Work, 4th ed., Springer, 2017.
7. D. Simon, K. Fischbach, and D. Schoder, "Enterprise architecture management and its role in corporate strategic management," Information Systems and e-Business Management, vol. 12, no. 1, pp. 5–42, 2014.
8. P. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, vol. 12, no. 6, pp. 42–50, 1995.
9. L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 4th ed., Addison-Wesley, 2021.
10. N. Rozanski and E. Woods, Software Systems Architecture, 2nd ed., Addison-Wesley, 2012.
11. P. Clements et al., Documenting Software Architectures: Views and Beyond, 2nd ed., Addison-Wesley, 2011.
12. C. Hofmeister, R. Nord, and D. Soni, "Applied Software Architecture," Addison-Wesley, 2000.
13. C. Hofmeister, R. Nord, and D. Soni, "A general model of software architecture design derived from five industrial approaches," Journal of Systems and Software, vol. 80, no. 1, pp. 106–126, 2007.
14. C. Lange and P. Avgeriou, "Documentation of software architecture: A survey," Journal of Systems and Software, vol. 82, no. 4, pp. 548–569, 2009.
15. P. Avgeriou, "Architectural knowledge management: Past, present, and future," IEEE Software, vol. 31, no. 6, pp. 66–73, 2014.
16. R. Kazman, J. Asundi, and M. Klein, "Quantifying the costs and benefits of architectural decisions,"Proceedings of the 23rd International Conference on Software Engineering (ICSE),pp. 297–306, 2001.
17. R. Kazman, L. Bass, M. Klein, and G. Abowd, "ATAM: Method for architecture evaluation," SEI Technical Report, CMU/SEI-2000-TR-004, 2000.
18. P. Kruchten, P. Lago, and H. van Vliet, "Building up and reasoning about architectural knowledge," Quality of Software Architectures, LNCS 4214, Springer, 2006.
19. J. Tyree and A. Akerman, "Architecture decisions: Demystifying architecture," IEEE Software, vol. 22, no. 2, pp. 19–27, 2005.
20. M. Fowler and J. Lewis, "Microservices: a definition of this new architectural term," martinfowler.com, 2014.
21. S. Newman, Building Microservices, 2nd ed., O'Reilly Media, 2021.
22. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," ACM Queue, vol. 14, no. 1, 2016.
23. C. Richardson, Microservices Patterns, Manning, 2018.
24. M. Kleppmann, Designing Data-Intensive Applications, O'Reilly Media, 2017.
25. E. Evans, Domain-Driven Design, Addison-Wesley, 2003.
26. N. Marz and J. Warren, Big Data: Principles and Best Practices of Scalable Realtime Data Systems, Manning, 2015.

 doi:10.20944/preprints202601.0720.v1

27. A. Shostack, Threat Modeling: Designing for Security, Wiley, 2014.
28. ISO/IEC, ISO/IEC 27001:2022 — Information Security Management Systems, ISO/IEC, 2022.
29. R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems,3rd ed., Wiley, 2020.
30. J. Bosch, Design and Use of Software Architectures, Addison-Wesley, 2000.
31. D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch," IEEE Software, vol. 12, no. 6, pp. 17–26, 1995.
32. R. de Boer et al., "Architectural knowledge: Getting to the core," Software Architecture, LNCS 5292, Springer, 2008.
33. S. Brown, The C4 Model for Software Architecture, https://c4model.com, 2018.
34. IEEE Computer Society, "Software Architecture in Industrial Practice," IEEE Software, special issues, multiple years.
35. P. Kruchten, "What colour is your architecture?" IEEE Software, vol. 23, no. 4, pp. 86–88, 2006.
36. H. van Vliet, Software Engineering: Principles and Practice, 3rd ed., Wiley, 2008.
37. A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar, "A comparative study of architecture knowledge management tools," Journal of Systems and Software, vol. 83, no. 3, pp. 352–370, 2010.
38. E. Breck et al., "The ML test score: A rubric for ML production readiness," IEEE Big Data, 2017.
39. D. Sculley et al., "Hidden technical debt in machine learning systems," NIPS, 2015.
40. M. Amershi et al., "Software engineering for machine learning," IEEE Software, vol. 36, no. 4, pp. 81–90, 2019.