Architecture as a Directed Object–Relation Graph: A Minimal and Complete Model of Complex Systems

Alexey A. Nekludoff

AstraVerge Research

E-mail: an@astraverge.org
ORCID: 0009-0002-7724-5762

25 February 2026

Abstract

Architectural descriptions of complex systems are traditionally centered around diagrams, viewpoints, and notational frameworks. This paper argues that such representations are not architectural primitives, but derived artifacts that do not belong to the ontological core of architecture.

We propose a minimal and complete architectural model in which any architecture is fully determined by a set of objects and a set of directed relations between them. Within this model, architectural analysis is formulated in terms of canonical graph operators, and all diagrams, views, and viewpoints are shown to be computable projections of a single underlying object–relation graph.

The separation between architectural ontology and representation eliminates diagram-level redundancy and ensures invariance of architectural reasoning under changes of notation, tooling, or visualization. The model is inspired by principles of the Architecture of Complex Systems (ACS) and is supported by a reference implementation, demonstrating that architectural reasoning can be carried out independently of visualization artifacts.

Contents

1 Introduction

Architectural descriptions of complex systems are commonly expressed through diagrams, layered views, and notations tailored to specific audiences. While such representations are useful for communication, they are often implicitly treated as primary architectural artifacts, rather than as derived means of access to an underlying structure.

This paper adopts a different position. We argue that diagrams do not constitute architecture itself, but represent derived views computed from a more fundamental architectural core. Treating diagrams as ontological entities introduces redundancy, encourages inconsistency between representations, and ties architectural reasoning to specific tools and notations.

Inspired by the Architecture of Complex Systems (ACS) [3], we propose a minimal architectural model in which architecture is defined solely by a set of objects and a set of directed relations between them. Within this framework, architectural analysis is expressed in terms of canonical graph operations, rather than through the manipulation or comparison of diagrams.

The central claim of this work is that *any architectural diagram can be expressed as a computable projection of a directed object–relation graph.* By separating architectural ontology from representational form, the model ensures that architectural reasoning remains invariant under changes of notation, tooling, or visualization.

The emphasis on structure over representation follows a long tradition in the study of complex systems, originating with Simons analysis of architectural complexity [6], and extends this perspective by making the separation between structure and representation explicit and operational.

2 Architectural Ontology

We define an architecture as a pair

$$\mathcal{A} = (O, R),$$

where O is a finite set of objects and $R \subseteq O \times O$ is a set of directed relations between objects.

2.1 Objects

Objects represent any distinguishable architectural entities, including physical components, software systems, services, interfaces, endpoints, or abstract functional units.

No assumptions are made about internal structure, scale, or implementation. Objects are treated as atomic from the perspective of architectural reasoning.

An object without relations does not participate in the architecture and is therefore architecturally irrelevant.

2.2 Relations

Relations are directed and represent flows, dependencies, interactions, or any form of architectural linkage between objects.

The model intentionally avoids semantic overloading of relations. Different interpretations (e.g. data flow, control flow, dependency) are treated as contextual refinements rather than ontological primitives.

Directionality is essential. Incoming and outgoing relations correspond to distinct architectural roles and must not be conflated.

The treatment of architecture as a relational structure aligns with early insights on modular decomposition, where relations, rather than component boundaries, determine architectural coherence [5].

3 Directed Relations and Reachability

Given the architectural graph $\mathcal{A} = (O, R)$, we define directed reachability as a fundamental operation.

For an object $o \in O$ and a non-negative integer d, the forward reachability set is defined as

$$T_d^+(o) = \{o' \in O \mid \exists \text{ a directed path } o \to o' \text{ of length } \leq d\}.$$

Similarly, the inverse reachability set is defined as

$$T_d^-(o) = \{o' \in O \mid \exists \text{ a directed path } o' \to o \text{ of length } \leq d\}.$$

These operators distinguish an objects role as a source of architectural influence from its role as a receiver. This separation is canonical and cannot be recovered from undirected or diagram-centric representations.

4 Architectural Views as Graph Projections

Architectural views and diagrams are commonly treated as primary artifacts of architectural description. In this work, we adopt a different stance: views are not ontological entities, but computable projections of the underlying architectural graph.

Let $\mathcal{A} = (O, R)$ be an architectural graph, where O is the set of objects and R the set of directed relations. An architectural view is defined as a function

$$V : O \times D \times \mathbb{N} \to \mathcal{P}(O \times O),$$

where $D \in \{+, -\}$ denotes the direction of analysis (outgoing or incoming), and the natural number parameter bounds the depth of traversal.

Given an object $o \in O$, a direction D, and a depth limit d, the resulting view $V(o, D, d)$ is the induced subgraph consisting of all relations whose endpoints lie within the corresponding reachability set.

Formally, for the outgoing direction ($D = +$), the view is defined as

$$V(o, +, d) = \{(u, v) \in R \mid u \in T_d^+(o)\},$$

while for the incoming direction ($D = -$), the view is defined as

$$V(o, -, d) = \{(u, v) \in R \mid v \in T_d^-(o)\}.$$

These definitions make explicit that the same architectural graph may give rise to infinitely many distinct views, depending on the choice of focal object, direction, and depth. No view introduces new architectural entities; each is fully determined by the underlying graph and the projection parameters.

This formulation establishes a strict separation between architectural ontology and representation. Views are neither stored nor maintained as independent structures, but are recomputed as needed from canonical operators. Consequently, architectural consistency is preserved by construction, and representational changes do not affect the underlying model.

By treating diagrams as projections rather than primitives, architectural reasoning becomes invariant under changes of notation, visualization style, or tooling. The architecture itself remains a single, stable graph, while views serve as contextual lenses through which specific aspects of the system are examined.

5 Paths and Structural Vulnerabilities

Beyond local reachability, architectural analysis often requires reasoning about connectivity between arbitrary objects and identifying structurally critical components. Within the proposed model, both tasks are expressed as properties of paths in the directed architectural graph.

5.1 Paths Between Objects

Let $\mathcal{A} = (O, R)$ be an architectural graph. A directed path from object $a \in O$ to object $b \in O$ is defined as a finite sequence of objects

$$(a = o_0, o_1, \ldots, o_k = b)$$

such that $(o_i, o_{i+1}) \in R$ for all $i < k$.

The existence of a path from a to b indicates a potential architectural dependency, influence, or flow between the two objects. Directionality is essential: a path from a to b does not imply a path from b to a.

Three canonical path-related questions arise naturally:

- *Existence*: does at least one directed path from a to b exist?
- *Minimality*: what is the shortest directed path from a to b?
- *Completeness*: what is the set of all directed paths from a to b under a given depth constraint?

All three questions are reducible to bounded reachability and path enumeration over the same underlying graph. No additional architectural constructs are required.

5.2 Structural Vulnerabilities

Architectural vulnerability is often discussed in operational or reliability terms. In contrast, we introduce a purely structural notion derived directly from the topology of the architectural graph.

Let $deg^+(o)$ denote the number of outgoing relations of object o, and $deg^-(o)$ the number of incoming relations. Given non-negative thresholds M and N, an object $o \in O$ is defined as a *structural vulnerability point* if

$$deg^+(o) \geq M \quad \text{and} \quad deg^-(o) \geq N.$$

Intuitively, such an object concentrates both architectural influence and architectural dependence. Its removal or failure may therefore fragment the graph or disrupt multiple independent paths.

This definition is independent of runtime metrics, failure probabilities, or service-level indicators. It characterizes vulnerability as an intrinsic architectural property rather than an operational condition.

Remark. Structural vulnerability points often (SVP) correspond, in operational practice, to what are informally referred to as *critical failure points* (CFP). However, the present definition is purely structural and does not presuppose the occurrence of failure.

5.3 Distinguishing Structure from Operation

The proposed notion of structural vulnerability does not predict failures. Instead, it identifies architectural concentrations that warrant special attention during design, review, or evolution of the system.

Operational monitoring and reliability engineering may mitigate or mask the effects of such vulnerabilities, but they cannot eliminate the underlying structural fact. Conversely, a system may be operationally stable while remaining architecturally fragile.

By separating structural vulnerability from operational behavior, the model enables architectural analysis to precede and inform implementation-level decisions, rather than being derived from them.

5.4 Formal Properties

Lemma 5.1 (Path–Reachability Equivalence)**.** *For any objects $a, b \in O$ and depth bound d, there exists a directed path from a to b of length at most d if and only if $b \in T_d^+(a)$.*

Proof sketch. By definition, $T_d^+(a)$ consists exactly of those objects reachable from a by a directed path of length at most d. The forward direction follows directly from the construction of the reachability set, while the reverse direction follows from path concatenation. □

Lemma 5.2 (Directional Non-Symmetry of Dependency)**.** *In a directed architectural graph, the existence of a path from a to b does not imply the existence of a path from b to a.*

Proof sketch. The claim follows from the directionality of relations. Unless the graph contains a directed cycle connecting b back to a, reverse reachability is not guaranteed. □

Lemma 5.3 (Existence of Canonical Shortest Paths)**.** *If there exists at least one directed path from a to b in $\mathcal{A}$, then there exists a shortest directed path whose length is minimal among all such paths.*

Proof sketch. Since all paths are finite sequences of relations and path length is a natural number, the set of path lengths from a to b, if non-empty, has a minimum. □

Lemma 5.4 (Structural Concentration)**.** *Let $o \in O$ be an object such that $deg^+(o) \geq M$ and $deg^-(o) \geq N$ for $M, N > 0$. Then o lies on at least $\min(M, N)$ distinct directed paths connecting different pairs of objects.*

Proof sketch. Each incoming relation contributes at least one predecessor, and each outgoing relation contributes at least one successor. By pairing incoming and outgoing relations, one obtains distinct directed paths passing through o. □

Lemma 5.5 (Structural Vulnerability Independence)**.** *Structural vulnerability, as defined by degree thresholds, is invariant under operational parameters such as load, failure rates, or runtime availability.*

Proof sketch. The definition of structural vulnerability depends solely on the graph topology, which is independent of runtime behavior. Operational parameters do not alter the existence or direction of relations. □

5.5 Formal Results

Theorem 5.6 (Minimality of the Object–Relation Model)**.** *Let $\mathcal{A} = (O, R)$ be an architectural graph, and let $\mathcal{V}$ denote the family of all bounded, directed views defined as projections $V(o, D, d)$ over $\mathcal{A}$ (Section 4), together with path-existence and shortest-path queries (Section 5).*

Then the following hold:

1. ***Sufficiency.*** *The pair (O, R) is sufficient to determine every element of $\mathcal{V}$. In particular, for any $o \in O$, $D \in \{+, -\}$, and $d \in \mathbb{N}$, the view $V(o, D, d)$ is uniquely determined by (O, R), and for any $a, b \in O$ all path-existence and shortest-path properties are determined by (O, R).*

2. ***Irreducibility.*** *No proper substructure of (O, R) that omits at least one object from O or at least one relation from R can, in general, determine the same family $\mathcal{V}$. That is, there exist architectures for which removing any single object or relation changes at least one view or path property in $\mathcal{V}$.*

Proof sketch. (*Sufficiency*) By definition, reachability sets $T_d^+(o)$ and $T_d^-(o)$ are computed solely from the directed relation set R over the object set O. Views $V(o, D, d)$ are induced subgraphs defined by membership conditions over these reachability sets and relations in R. Path-existence is equivalent to bounded reachability (Lemma 1), and shortest-path existence follows from finiteness (Lemma 3). Hence all elements of $\mathcal{V}$ are uniquely determined by (O, R).

(*Irreducibility*) Consider an architecture with two objects and a single directed relation: $O = \{x, y\}$ and $R = \{(x, y)\}$. Removing (x, y) changes reachability (e.g. $y \notin T_1^+(x)$), and thus changes $V(x, +, 1)$ and path existence from x to y. Similarly, removing either object changes the domain of views and eliminates at least one query instance. More generally, for any object or relation, one can construct an architecture in which that element is essential for some view or path property. □

Corollary 5.7 (Diagram Redundancy)**.** *Any architectural diagram or viewpoint description that is fully determined by bounded directed reachability and induced subgraphs introduces no new ontological information beyond (O, R). Such diagrams are therefore redundant as stored architectural primitives: they can be recomputed on demand as projections of the architectural graph.*

Proof sketch. A diagram of the stated class corresponds to a view $V(o, D, d)$ or a composition of such views. By the sufficiency part of Theorem 1, $V(o, D, d)$ is uniquely determined by (O, R). Storing the diagram separately duplicates information already contained in the graph representation, while recomputation preserves correctness by construction. □

Definition 5.8 (Diagram Class)**.** *A* structural architectural diagram *is any diagram whose nodes correspond to a subset of objects O and whose edges correspond to a subset of directed relations R, possibly restricted by predicates on objects, relations, direction, or bounded path length.*

Formally, a structural architectural diagram is defined by a tuple

$$(o, D, d, P),$$

where $o \in O$ is a focal object, $D \in \{+, -\}$ is the direction of analysis, $d \in \mathbb{N}$ is a depth bound, and P is a finite set of selection predicates that filter objects or relations without introducing new ontological entities.

Remark. Annotations, labels, visual styling, and layout information commonly found in architectural diagrams are explicitly excluded from the ontological core. Such elements may aid interpretation or communication, but do not affect the underlying architectural structure and are therefore treated as representational metadata rather than architectural primitives.

6 Discussion

The proposed model shifts architectural reasoning from representational artifacts to structural operators. This shift has several important implications for architectural practice and theory.

6.1 Ontology Versus Representation

Traditional architectural approaches often conflate the ontology of a system with its representations. Diagrams, layers, and viewpoints are treated as primary architectural entities, with relationships reconstructed implicitly from visual or notational conventions.

In contrast, the present model defines architecture solely in terms of objects and directed relations. Representations are derived, not stored. This separation eliminates representational inconsistency and removes the need for manual synchronization between multiple diagrams.

While contemporary architectural practice often relies on diagrammatic views for communication and analysis [1], the present model treats such views as derived artifacts rather than ontological primitives.

The strict separation between ontological structure and representational artifacts follows the same methodological stance adopted in earlier work on document-centric systems [2].

6.2 Invariance Under Representation

Because architectural views are defined as projections of a single underlying graph, the model is invariant under changes of notation, diagram style, or visualization tooling. Different representations of the same view do not affect the architecture itself.

This invariance enables architectural analysis to remain stable across organizational boundaries, documentation formats, and lifecycle phases. Architectural knowledge becomes portable rather than embedded in tool-specific artifacts.

6.3 Canonical Analysis Operators

By grounding architectural analysis in a small set of canonical graph operators (reachability, path analysis, and degree-based metrics), the model avoids the proliferation of ad hoc viewpoints. Each analysis is expressible as a composition of well-defined operations over the same architectural graph.

This approach encourages repeatability and formal reasoning. Architectural questions become queries over a shared structure, rather than interpretations of diagrams.

6.4 Relation to the Architecture of Complex Systems

The model aligns naturally with the principles of the Architecture of Complex Systems (ACS), which emphasize structure, interaction, and emergent behavior over static decomposition. By treating architecture as a directed graph, the proposed approach provides a concrete and operational interpretation of these principles.

While ACS is often discussed at a conceptual level, the present work demonstrates that its core ideas can be realized as precise operators without introducing additional ontological layers.

6.5 Limits and Scope

The proposed model is intentionally minimal. It does not prescribe relation semantics, object taxonomies, or modeling notations. Such concerns are treated as external refinements that may be layered atop the core graph.

This minimalism is a strength rather than a limitation. By constraining the ontological core, the model remains adaptable to diverse domains and evolving requirements, while preserving analytical rigor.

The ontological priority of architecture over dynamics adopted here is consistent with earlier regime-based analyses of computing architectures [4], although the present work operates at the level of object–relation graphs rather than regime transitions.

7 Conclusion

This paper has presented a minimal and complete model of architecture for complex systems, grounded in a directed object–relation graph. We have shown that the architectural ontology requires no primitives beyond objects and directed relations, and that all commonly used architectural views can be expressed as computable projections of this structure.

By formalizing reachability, path analysis, and structural vulnerability as graph operators, the model separates architectural reasoning from representational artifacts. Diagrams are reinterpreted as contextual views rather than ontological entities, eliminating redundancy and preserving consistency by construction.

The minimality of the object–relation model was established by demonstrating both its sufficiency for deriving all canonical architectural views and its irreducibility with respect to those views. As a consequence, storing diagrams as primary architectural artifacts introduces no additional structural information.

Although the model is intentionally abstract, it is not purely theoretical. A reference implementation of the proposed operators has been realized directly at the relational database level, without application-layer recursion, demonstrating the practical viability of the approach.

By relocating architectural reasoning from diagrams to graph operators, the proposed model provides a stable foundation for analysis, comparison, and evolution of complex architectures, independent of notation, tooling, or visualization.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley, 2012.

[2] A. A. Nekludoff, "A minimalist design ontology for document-centric web systems: Html as a primary artifact," 2026. DOI: 10.5281/zenodo.18492483 [Online]. Available: https://doi.org/10.5281/zenodo.18492483

[3] A. A. Nekludoff, "Architecture of complex systems," 2026. DOI: 10.5281/zenodo.18277986 [Online]. Available: https://doi.org/10.5281/zenodo.18277986

[4] A. A. Nekludoff, "The ontology of continuation: Growth fronts, closure points, and the stabilizing role of atomic hydrogen," 2026. DOI: 10.5281/zenodo.18473120 [Online]. Available: https://doi.org/10.5281/zenodo.18473120

[5] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.

[6] H. A. Simon, "The architecture of complexity," *Proceedings of the American Philosophical Society*, vol. 106, no. 6, pp. 467–482, 1962.