Living Decision Architecture

Abstract

This paper proposes *Living Decision Architecture* (LDA) as an alternative to static, document-centric approaches to architectural governance. LDA treats architecture not as a set of declared intentions, but as an evolving property of a system observable through behaviour, change, and constraint.

Rather than attempting to record architectural decisions as isolated artefacts, LDA focuses on making decisions *inferable* from the interaction between code, runtime behaviour, and delivery activity. The objective is not to replace all forms of design documentation, but to anchor architectural understanding in evidence that evolves with the system itself.

1. The Limits of Decision Records

Architecture Decision Records (ADRs) were introduced to capture architectural intent and rationale in a lightweight, reviewable form. In principle, they offer traceability and institutional memory. In practice, their effectiveness is limited by how they are produced and maintained.

Common failure modes include:

- **Post-hoc rationalisation**: decisions are documented after implementation, presenting a simplified or idealised narrative that omits trade-offs, uncertainty, and contingency.

- **Detachment from reality**: ADRs frequently describe models that diverge from the behaviour of the running system.
- **Illusory auditability**: the presence of a document is mistaken for traceability, despite no linkage to code changes or operational state.
- **Process substitution**: the artefact becomes the goal, satisfying governance expectations without improving architectural understanding.

The result is a body of documentation that describes a system as it was imagined in meetings, rather than as it exists in production.

2. Architecture as an Observable Property

If architecture is understood as the structure of a system and the constraints that shape its evolution, then it follows that architecture is most accurately observed where those constraints are exercised:

- in code changes,
- in runtime interactions,
- and in the patterns of work that modify the system.

From this perspective, decisions are not primarily textual declarations. They are *expressed* through:

- the interfaces systems expose,
- the dependencies they acquire or shed,
- the failure modes they tolerate,
- and the costs they impose on future change.

Living Decision Architecture begins from this premise: architectural understanding should be derived from evidence generated by the system itself.

3. The Living Decision Architecture Model

LDA is not a tool or a framework. It is a way of wiring existing practices so that architectural decisions become traceable as a consequence of normal delivery activity.

At a high level, LDA links four domains:

1. **Runtime behaviour**
2. **Code change**
3. **Delivery intent**
4. **Architectural structure**

The aim is to allow questions about architecture to be answered by correlation rather than recollection.

4. Core Mechanisms

4.1 Behavioural Identification

All inter-component interactions carry explicit identity and correlation metadata. At minimum, this includes:

- a stable identifier for the calling component,
- a propagated trace or correlation identifier.

This information is enforced through shared libraries rather than convention, ensuring consistency and completeness.

4.2 Observability as Architectural Evidence

Runtime telemetry is treated as architectural data rather than purely operational data. This includes:

- call graphs and dependency direction,
- latency and volume characteristics,
- retry and failure behaviour.

Collected centrally, this information describes how the system is actually composed and exercised over time.

4.3 Structure Extraction

From observed behaviour, architectural structure can be periodically derived:

- service or component graphs weighted by interaction intensity,
- identification of critical paths and choke points,
- emergence of coupling not reflected in design documentation.

This structure is descriptive rather than aspirational. It reflects what exists, not what was intended.

4.4 Change Correlation

Delivery artefacts are explicitly linked to system structure:

- work items identify affected components,
- code changes reference work items,
- deployments can be correlated with behavioural change.

This allows architectural impact to be traced through actual change, rather than inferred from documentation alone.

5. What LDA Enables

When these mechanisms are in place, a range of architectural questions become answerable without interpretive reconstruction:

- When was a particular dependency introduced?
- Which changes altered a system’s failure characteristics?
- How did a specific piece of work affect downstream components?
- Which parts of the system are becoming structurally critical over time?

The answers are grounded in artefacts generated as part of normal engineering work.

6. Relationship to Design Documentation

LDA does not eliminate the need for design discussion, modelling, or intentional constraint setting. Initial designs, goals, and hypotheses remain valuable.

However, LDA changes their role:

- design documents express intent,
- living architecture expresses outcome.

Where divergence occurs, it becomes visible rather than obscured.

7. Adoption Characteristics

LDA does not require new platforms or wholesale process change. Most organisations already possess the necessary components:

- version control systems,
- work tracking tools,
- structured logging and observability infrastructure.

Adoption consists primarily of:

- enforcing consistent metadata,
- correlating existing data sources,
- and treating the resulting view as authoritative for architectural understanding.

8. Architectural Implications

Living Decision Architecture reframes architectural governance:

- from approval to observation,
- from declared rationale to demonstrable effect,
- from static records to evolving structure.

It does not prevent poor decisions. It makes their consequences visible and attributable.

9. Conclusion

Living Decision Architecture treats systems as capable of explaining themselves. By grounding architectural understanding in behaviour, change, and constraint, it replaces brittle narratives with durable evidence.

The result is not cleaner architecture, but more honest architecture: one that evolves in the open and can be reasoned about as it actually exists, rather than as it was once described.