ENTERPRISE ARCHITECTURE PROFESSIONAL JOURNAL

NOVEMBER 2025

IN THIS ISSUE

EDITOR'S WELCOME

by Darryl Carr, EAPJ Editor

Welcome to the November 2025 Edition of the Enterprise Architecture Professional Journal. We serve practicing and aspiring enterprise architects, as well as those who apply the holistic perspective of enterprise architecture to other disciplines. EAPJ informs their daily work and benefits their careers with content that is focused, concise, authoritative, practical and accessible.

In this edition we have two feature articles:

- Reza Karami provides an insightful article on Enterprise Architecture Debt, highlighting its importance to organizations and EA practitioners, and suggesting ways of identifying, tracking and managing this potentially harmful characteristic that is inherent in most large organizations.
- Jackie O'Dowd has contributed an intriguing paper, focused on the much-needed ability for Architects to be able to gather, visualize and use the structured and unstructured data that exists in enormous quantities in organizations, but is rarely utilized effectively for key decision making.

We also have a note from EAPJ Founder, Dr Steve Else, talking about the purpose and drivers for starting the Enterprise Architecture Professional Journal more than a decade ago, and recognizing all of the contributors and volunteers that have turned it into a valuable resource for the profession.

It's been a busy year on the main EAPJ website, with an increase in submissions from around the world, and two versions of the Journal. We've also welcomed new columnists and reviewers, increasing the volume and value of content we are sharing with readers. It's great to see the growth over the past few years, and we look forward to it continuing.

The team at EAPJ hope you enjoy reading this edition. Please contact me at editor@eapj.org with your questions, comments, ideas and submissions. As always, I look forward to hearing from you!

Darryl Carr
Editor, Enterprise Architecture Professional Journal

FOUNDER'S NOTE

by Dr. Steve Else, EAPJ Founder

When I first began practicing Enterprise Architecture over a decade ago, I quickly discovered how difficult it was for architects to share meaningful insights in a way that was both professional and enduring. Our field is often demanding, misunderstood, and stretched thin across industries. Yet I knew it needed a place where ideas could be explored in depth—without the limits of conference slide decks, short blog posts, or word counts that clipped important arguments.

That vision became the **Enterprise Architecture Professional Journal (EAPJ)**: a space where thought leaders and practitioners could contribute articles, case studies, and reflections that stand the test of time. What makes the Journal distinctive is the balance we strive to uphold—rigorous peer review by expert Enterprise Architects, but with the flexibility to let contributors write fully and honestly, without arbitrary restrictions.

Over the years, EAPJ has welcomed voices from across verticals—finance, healthcare, government, energy, technology, and beyond. Each article strengthens the profession's collective knowledge and gives authors a credential that builds their bona fides in a challenging but rewarding career. Just as importantly, it offers future generations of architects a lasting record of how we've wrestled with transformation, governance, innovation, and the tools of our trade.

As we package this new issue, I remain deeply grateful for our contributors, reviewers, and readers. I'm proud to state that this Journal stands as proof that the discipline of Enterprise Architecture is not only alive and evolving but also anchored in a vibrant, thoughtful, and generous community.

FEATURE ARTICLE

What is Enterprise Architecture Debt and how to manage it?

By Reza Karami
karami@golsoft.com

Summary

Enterprise Architecture Debt (EAD) is a broadening of the concept of technical debt. This paper elaborates on the concept and its relationship to core elements of Enterprise Architecture (EA). It explores the origins and dynamics of EAD, demonstrating how the concept can be applied to define and manage EA change cycles, as well as to enhance EA management and governance practices. The paper concludes by introducing several research ideas for future exploration.

1. From Technical Debt to EA Debt

In software engineering, the term “Technical Debt” is well established. It was first introduced by Ward Cunningham in 1992, in his experience report on the WyCash Portfolio Management System. Cunningham used the metaphor of financial debt to describe the long-term consequences of taking shortcuts in software design and development—highlighting the trade-off between rapid delivery and maintainability. He wrote:

> *Although immature code may work fine and be completely acceptable to the customer, excess quantities will make a program unmasterable, leading to extreme specialization of programmers and finally an inflexible product. Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost of this transaction tolerable. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise*. [Cunningham-92]

Technical Debt is defined now as:

> *“In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability.”* [Dagstuhl-16]

It is evident that the term "Technical Debt" originates as a metaphor borrowed from the concept of financial debt. Far from being a superficial or arbitrary analogy, this metaphor captures the essence of the trade-offs inherent in software development. Technical debt mirrors financial debt in several

key aspects, including the accumulation of interest over time, the burden of repayment, and the risk of long-term consequences if left unmanaged:

- Debt, in itself, is not inherently negative. When used judiciously and strategically, it can serve as a powerful mechanism for growth. For both individuals and organizations, debt enables access to resources and opportunities that might otherwise be unattainable—facilitating accelerated development, innovation, and expansion. The key lies in leveraging debt wisely to ensure that its benefits outweigh its risks.
- Debt is often an unavoidable aspect of organizational growth. No firm can scale and sustain its operations without incurring a certain level of debt. Over an extended period, it is nearly impossible to find an organization that remains entirely free of debt. Strategic use of debt enables access to capital, resources, and market opportunities that can significantly enhance an organization's capacity for innovation and expansion.
- Beyond its reflection in annual financial statements, debt often influences non-functional and less visible dimensions of an organization. These subtle impacts—such as architectural integrity, long-term flexibility, and systemic resilience—are not immediately observable but carry significant weight over time. In this sense, debt resides in the submerged portion of the organizational iceberg: concealed beneath the surface, yet profoundly influential in shaping stability and capacity for adaptation.
- The longer the repayment of debt is postponed, the greater its cost to the debtor—typically in the form of accumulating interest charges. This compounding effect increases the financial burden over time, underscoring the importance of timely debt management to mitigate long-term consequences.
- Every individual or organization possesses a certain debt capacity, determined by the value of their assets and overall financial condition. Exceeding this threshold by incurring excessive debt can lead to financial insolvency or bankruptcy. Just as responsible financial management requires staying within debt limits, sustainable architecture practices must consider the capacity of systems, teams, and governance structures to absorb and manage architectural debt without compromising long-term viability.
- The ratio of an organization's debt to the value of its assets serves as a key indicator of its financial health. A balanced debt-to-asset ratio reflects responsible financial management and resilience, while an excessive ratio may signal increased risk and potential instability. This metric provides stakeholders with insight into the organization's leverage, solvency, and capacity to absorb financial shocks.
- The timing of incurring and repaying debt is critical to an organization's success. Strategic decisions about when to leverage debt for growth and when to reduce it to maintain stability can significantly influence long-term sustainability and operational resilience.

The Dagstuhl Seminar Report also presents a conceptual data model of technical debt, distinguishing between Technical Debt (TD) and TD Items. In this model, each TD Item represents a specific instantiation of a broader Technical Debt and is associated with identifiable causes and consequences. This structured approach enables a more granular analysis of technical debt, facilitating targeted mitigation strategies and improved governance across software systems.

Similar to many other concepts within the software engineering domain, technical debt can be generalized to broader contexts—first to IT architecture, and subsequently to enterprise architecture. This extension reflects situations in which the rapid delivery and ongoing maintenance of essential enterprise functionalities may come at the expense of integration, robustness, and other architectural qualities. Such trade-offs give rise to the concept of Enterprise Architecture Debt (EAD),

which encapsulates the cumulative impact of decisions that prioritize short-term utility over long-term architectural integrity.

2. What is EA Debt?

Enterprise Architecture Debt (EAD) extends the concept of technical debt beyond individual software systems to encompass the entire enterprise architecture. The term was elaborated by S. Hacks et al. in their 2019 paper Towards the Definition of Enterprise Architecture Debts and further developed through subsequent research published on the www.ea-debts.org website. EAD refers to the deviation of an organization from its ideal architectural state at any given point in time—regardless of whether that ideal state is explicitly defined or remains implicit. This deviation may result from strategic compromises, legacy constraints, or evolving business and IT landscapes, and it reflects the cumulative impact of architectural decisions on organizational agility and coherence.

This definition implicitly assumes that every organization possesses an "ideal" architectural state, even in the absence of explicit awareness or documentation. However, it may be more precise to conceptualize this not as a single, fixed ideal, but rather as a class of possible ideal states. In EA terminology, these are often referred to as "to-be" or "target" states—representing envisioned configurations that support strategic goals, operational efficiency, and adaptability to evolving business contexts.

The previously outlined characteristics of the debt metaphor can be consistently extended to the concept of EAD, by substituting financial assets with architectural building blocks. In this framework, "paying off" the debt corresponds to implementing architectural changes that reduce the gap between the current and target states. "Bankruptcy," in turn, can be interpreted as an organizational failure to perform its core mission—resulting from an unsustainable accumulation of architectural deficiencies that undermine structural coherence, operational capability, and strategic alignment.

The concept of EAD has garnered increasing attention in recent years. In their 2020 work, A. Koschmider et al. proposed a comprehensive EAD management framework that outlines a cyclic process model for addressing EA debts. This framework encompasses several key activities: identification and collection, assessment and prioritization, repayment and prevention, monitoring, and the documentation and communication of EA debts. Each phase contributes to a structured approach for recognizing, evaluating, and mitigating architectural deficiencies across the enterprise landscape [Koschmider-2020].

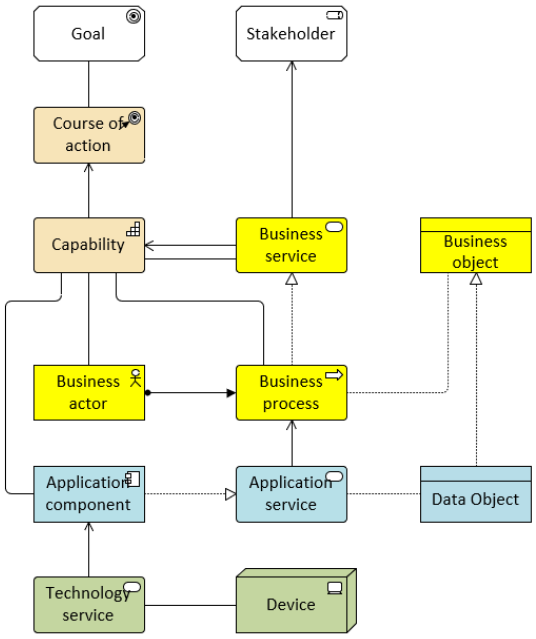
In their 2023 study, S. Daoudi et al. compiled a comprehensive taxonomy of Enterprise Architecture Debts (EAD), organized by distinct domains within Enterprise Architecture. This taxonomy identifies 12 categories of EAD, each characterized by specific architectural deficiencies and systemic implications. By classifying EADs in this structured manner, the study provides a valuable framework for analyzing architectural shortcomings and guiding targeted remediation efforts across various organizational contexts [Daoudi-2023].

3. Where does EAD come from?

Despite the fact that virtually no organization is founded with the intention of operating in chaos or inefficiency, complexity and architectural compromises are frequent realities in today's dynamic

business environments. EAD emerges not from deliberate neglect, but from the cumulative impact of decisions made under constraints—such as time pressure, resource limitations, evolving requirements, or shifting strategic priorities. So, where does EAD come from?

To contextualize the origins of EAD, it is helpful to view an organization as a system composed of interrelated architectural building blocks. These elements—such as organizational units, business processes, services, applications, and infrastructure—collectively form the foundation of enterprise architecture. The structure and interdependencies among these elements are typically described using a conceptual data model known as an architecture meta-model. The figure below presents a simplified representation of such a meta-model, modeled using the ArchiMate®[1] notation. It is important to note that this illustration is not intended to be complete, exhaustive, and universally applicable, but rather to serve as a conceptual reference for understanding architectural relationships within organizations:



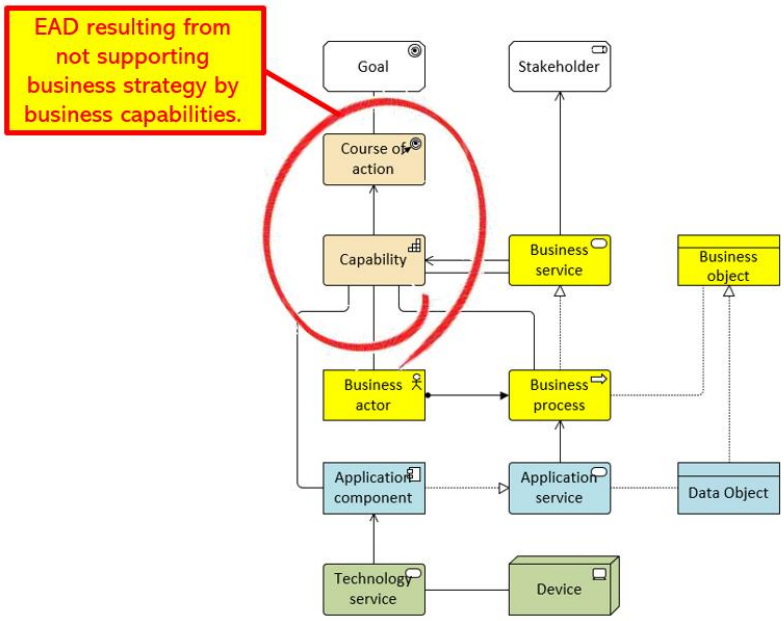
Any failure to maintain the elements and their relationships in the ideal state, as defined by the meta-model, may lead to the creation of an EAD item.

Here are some common types of EAD items:

3.1. Poor strategy support by capabilities

Business capabilities are fundamental building blocks of an organization that enable the execution of its strategy. They encompass all relevant organizational resources—people, processes, and technology—necessary to realize one or more strategic courses of action.

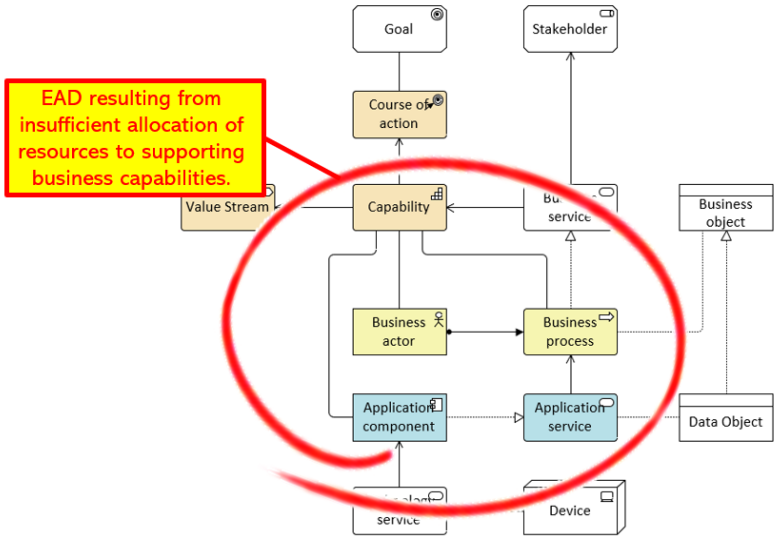
[1] ArchiMate is a registered trademark of The Open Group.



An EAD is identified when the organization lacks a sufficient set of enabling capabilities to support all strategic goals and courses of action. This type of debt can be readily detected by examining the Course of Action–Business Capability cross-matrix or by reviewing the standard Strategy View in ArchiMate. For example, consider a Course of Action-Business matrix for a manufacturing company, where the Product Diversification course of action has no associated business capability (maybe, a missing Product innovation Management business capability!)

3.2. Capability immaturity

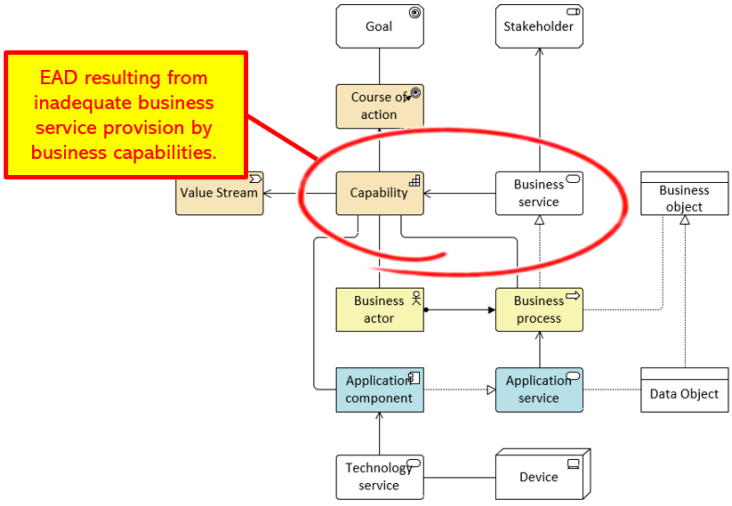
Business capabilities vary in their levels of maturity at any given time. The more mature a capability is, the better it can deliver its services and support the business. Factors such as a lack of assignment to organizational units or actors, undefined or poorly documented processes, and inadequate IT support can render capabilities immature—making them ineffective in supporting related strategies. These conditions represent another common form of EAD.



In the previous example (the manufacturer seeking a Product Diversification strategy) they may acknowledge that one of more components of associated business capability are too immature to realize the strategy adequately. For example, appropriate organizational responsibilities are not defined for innovation management, new product development processes are not designed well, or require technology services to support product development.

3.3. Business services inadequacy

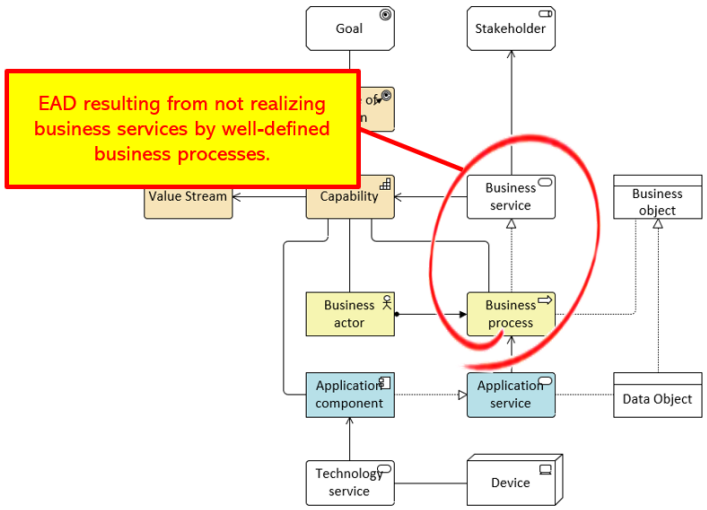
Business capabilities are expected to deliver adequate services to business actors—including external stakeholders—or to other capabilities, thereby enabling the enterprise to perform its functions effectively. A shortfall in providing these required services constitutes an EAD. Such debts are often caused by immature capabilities (particularly newly established ones) or by an incomplete definition of a capability's scope.



Returning to the previous example again, the company may have recently established a New Product Development capability, but one business service (say, Product Launch) is not yet defined.

3.4. Inadequate processes

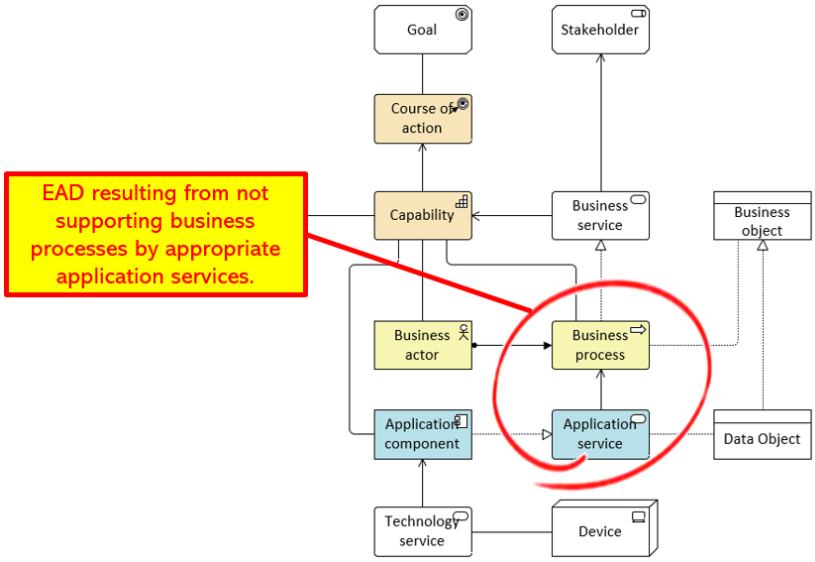
Every business service should be supported by one or more corresponding processes. In organizations with a weak process culture, these realizing processes may be inadequate or entirely absent for some—or even all—business services, resulting in a common form of EAD. Another frequent cause of this debt is the accelerated pace of change driven by digital transformation, which often compels businesses to launch new services before fully designing the supporting processes.



In the manufacturing company example, consider the situation where the Product Launch service is defined, but not realized by a well-designed repeatable business process.

3.5. Lack of IT support

Business processes should generally be supported by appropriate application services. A process lacking adequate application support, or lacking it altogether, signals an EAD. This type of debt is typically the result of uncovered areas of business operations, by appropriate IT applications, or a shortage of adding new features and functionalities to existing legacy applications to support changes in business processes

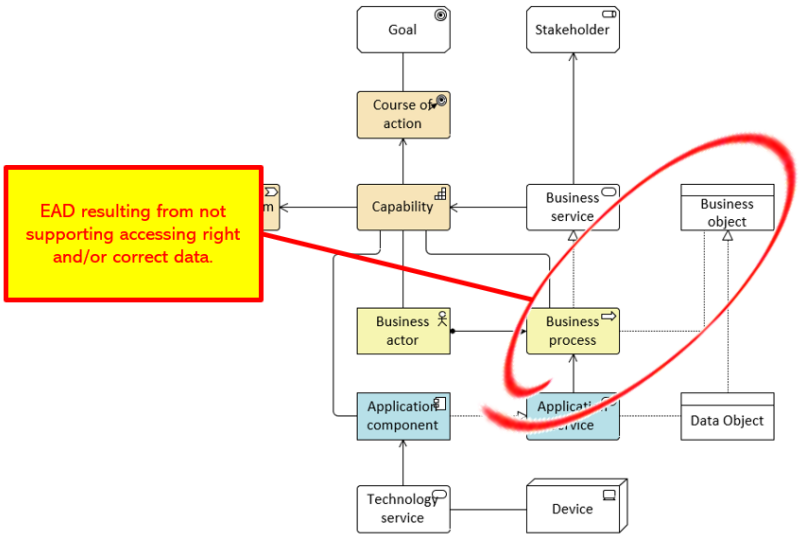


For example, consider that previously mentioned manufacturer, who has employed a Product Launch business process, but this process has no (or little) IT support and is performed manually.

3.6. Improper data access

Business processes must have access to the appropriate data in order to function properly and efficiently. In the enterprise architecture meta-model, this data is represented by business

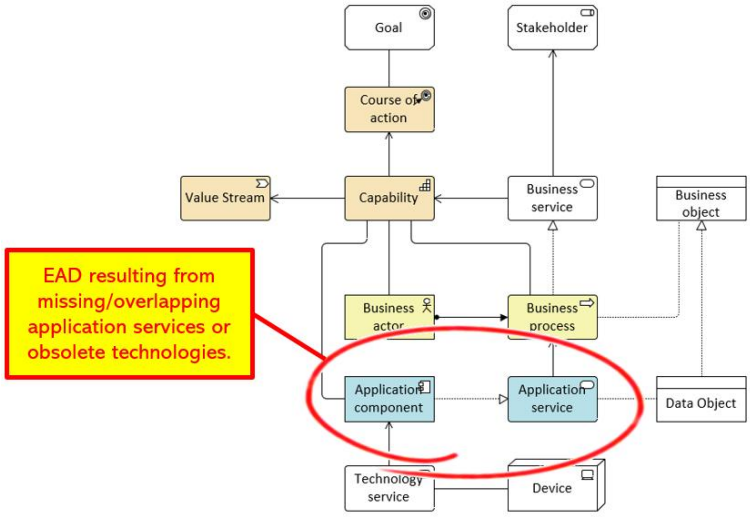
objects and data objects. When a process lacks systematic and architectural access to the required data, it constitutes an EAD.



In the manufacturing company example, the operational Product Launch process may have no access to current customers data, in order to reach out and offer the new product. This may result in an ineffective marketing for new products.

3.7. Messy application landscape

A wide range of EADs stem from issues within the enterprise application landscape. Common symptoms include applications that fail to deliver the required application services, the presence of overlapping or redundant services across multiple applications, and the use of outdated, hard-to-maintain technologies. Each of these conditions signals architectural misalignment and contributes to overall inefficiency.



The list of EAD types can be further extended by analyzing additional relationships among architecture elements. Cross-reference models—typically represented as matrices or layered diagrams—extracted

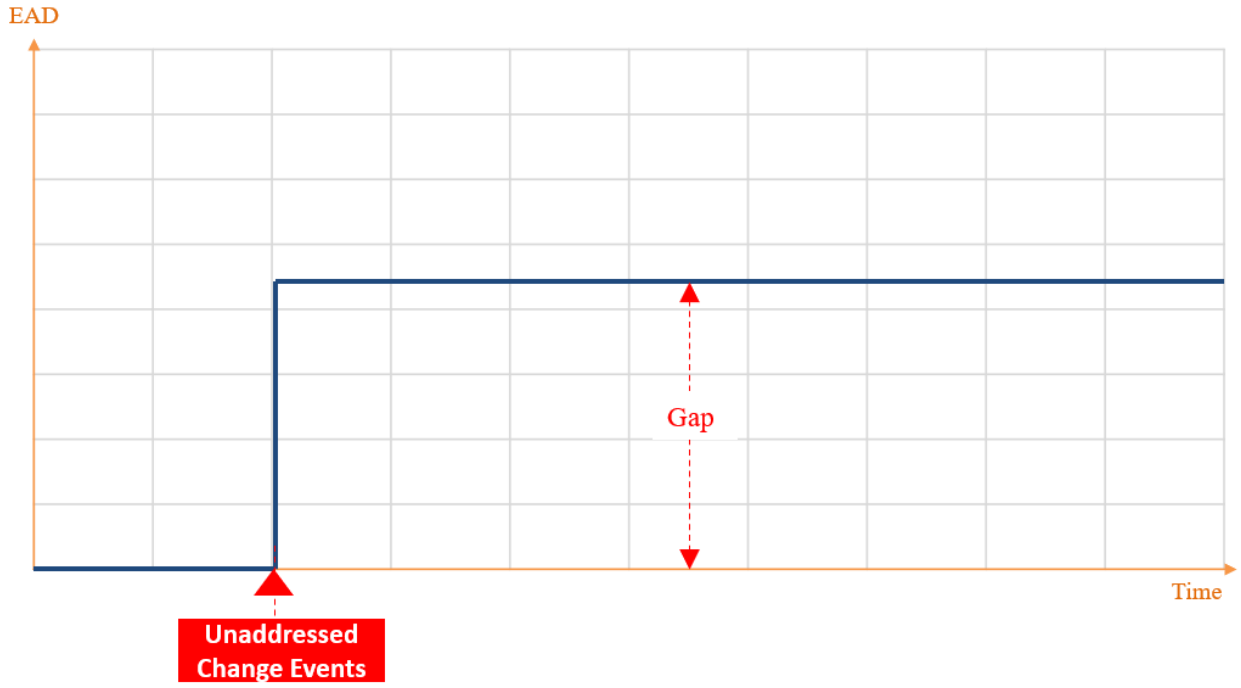
from the architecture repository provide valuable diagnostic tools for identifying EAD items in any given architectural state. The general idea here is that *EADs could be identified by the current architecture analysis*.

4. Dynamics of EA Debt

EAD is not a static phenomenon. EAD items are continually created and resolved over time. The study of these evolving patterns is referred to as the Dynamics of EAD. This perspective reveals how EADs emerge, evolve, and can be systematically managed when examined across an extended period.

Although no established method currently exists for quantifying EAD, it can be useful—especially in studying its dynamics—to illustrate changes in the hypothetical total EAD of an organization over time using a two-dimensional time graph.

The simplest conception of EAD dynamics could then be exhibited in the following diagram:

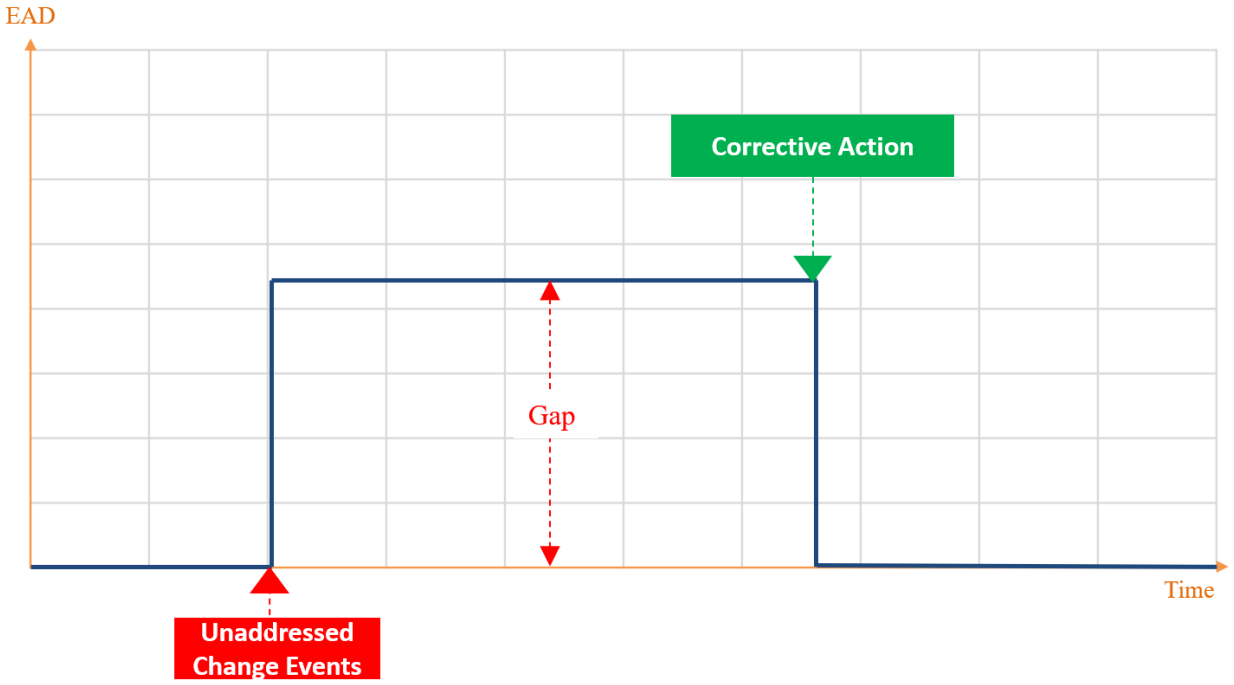


However, this conception of EAD rests on two flawed presuppositions:

a) The initial value of EAD is zero—for instance, at the moment an organization is established.

b) the EAD resulting from an unmanaged change event remains constant over time.

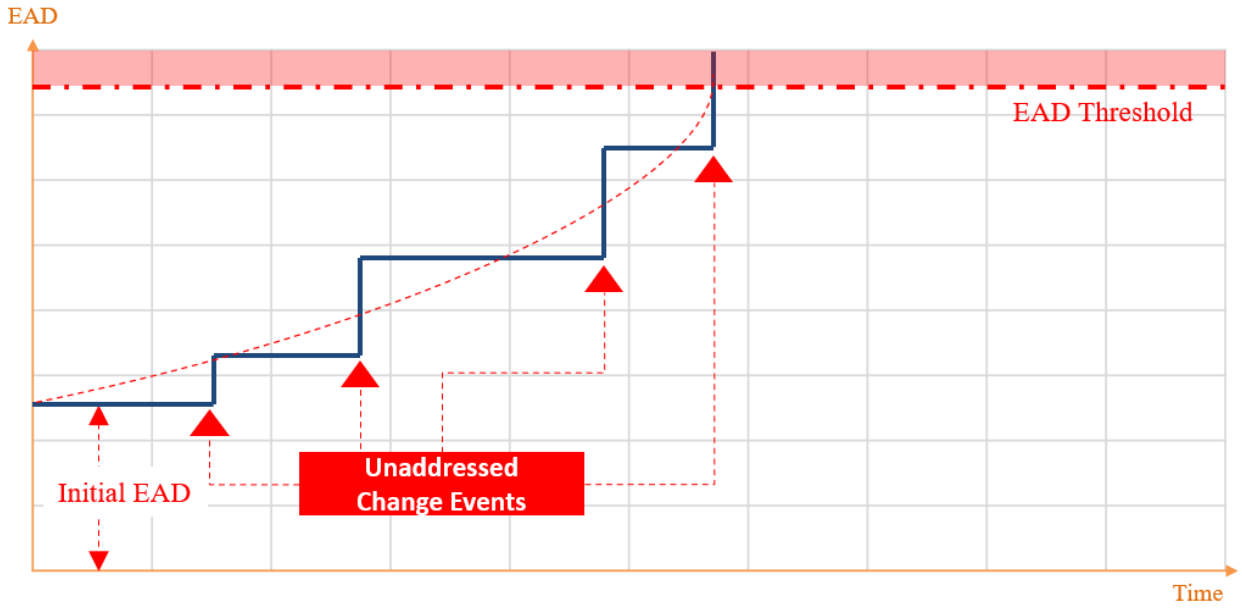
The third common misconception in understanding the dynamics of EAD arises from the belief that corrective actions can reduce EAD to zero, as often depicted in simplified models and illustrations.



To correct these misconceptions, it is important to acknowledge the following principles:

a) No organization begins with zero EAD; even at the moment of its establishment, some level of debt is inherently present.
b) No single corrective action can fully eliminate EAD; debt reduction is typically incremental and partial.
c) In the absence of deliberate intervention, the value of EAD tends to grow steadily over time

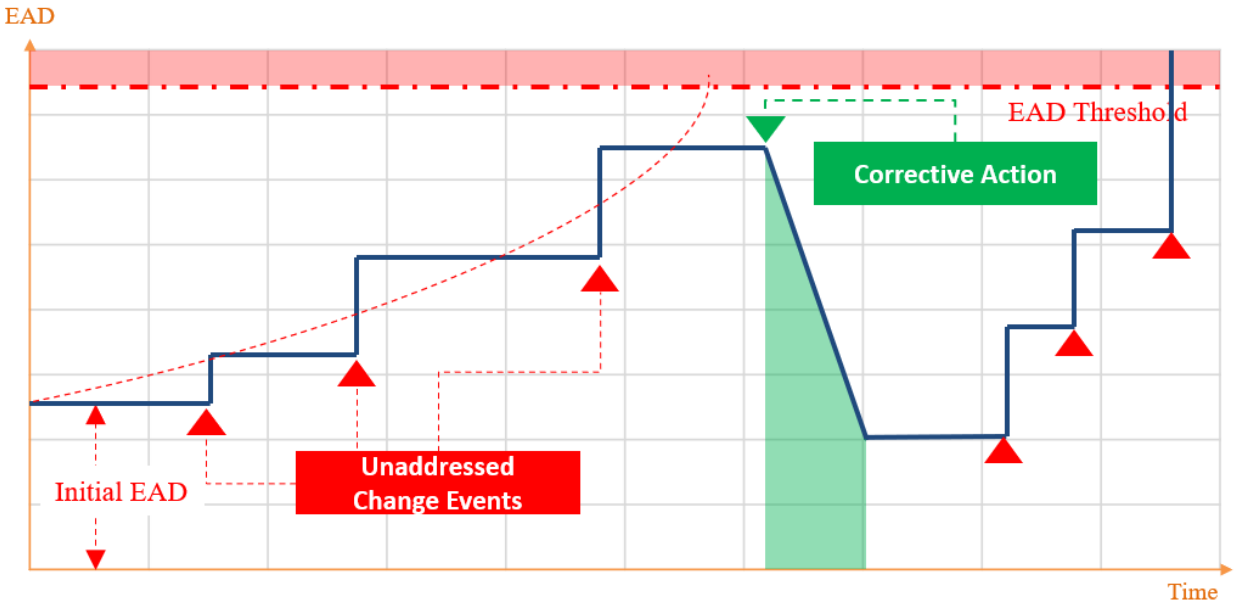
Applying these facts makes a more realistic image of EAD dynamics:



In the absence of corrective intervention, an organization's EAD tends to grow steadily due to a series of unaddressed change events—for example, the formulation of new strategies without corresponding enabling capabilities, or the definition of new business processes without adequate application service support.

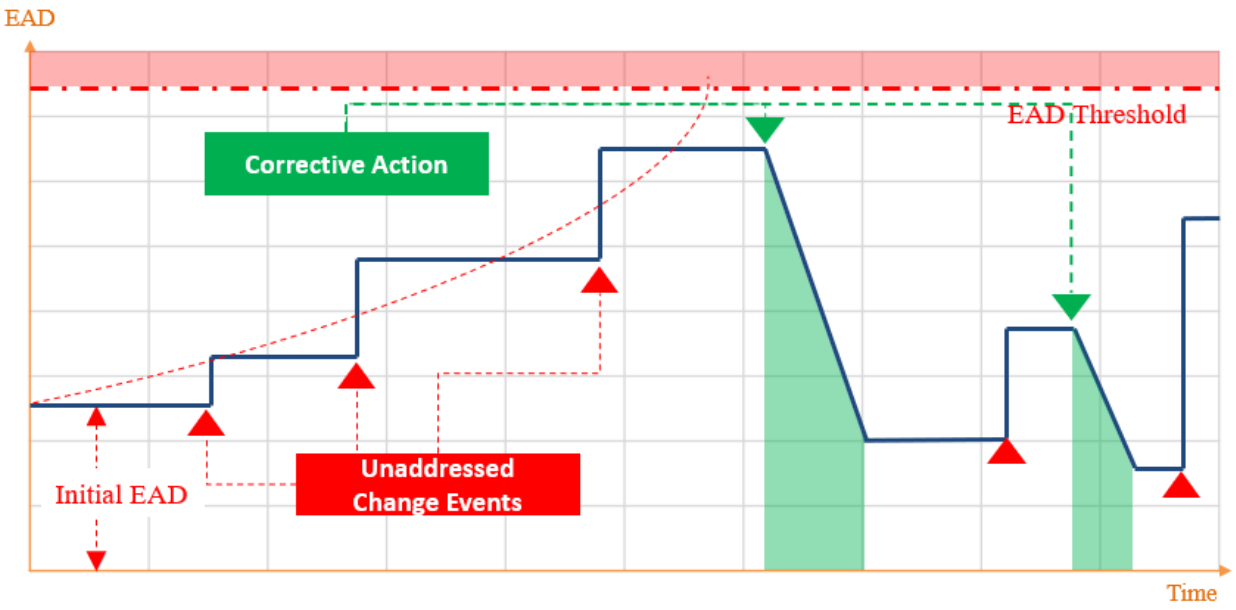
Over time, this compounding accumulation of EAD can push the organization beyond its EAD Threshold—the point at which architectural debt exceeds the organization's capacity to manage it effectively. It is the point at which, the organization enters to a state of being incapable of delivering its expected functionality[2]. At this stage, the enterprise enters a critical state akin to a bankruptcy level of debt.

To prevent EAD from exceeding the organizational threshold, it is essential to initiate an architecture change cycle. This cycle should define and implement corrective actions aimed at reducing EAD to a manageable level—though not necessarily to zero.

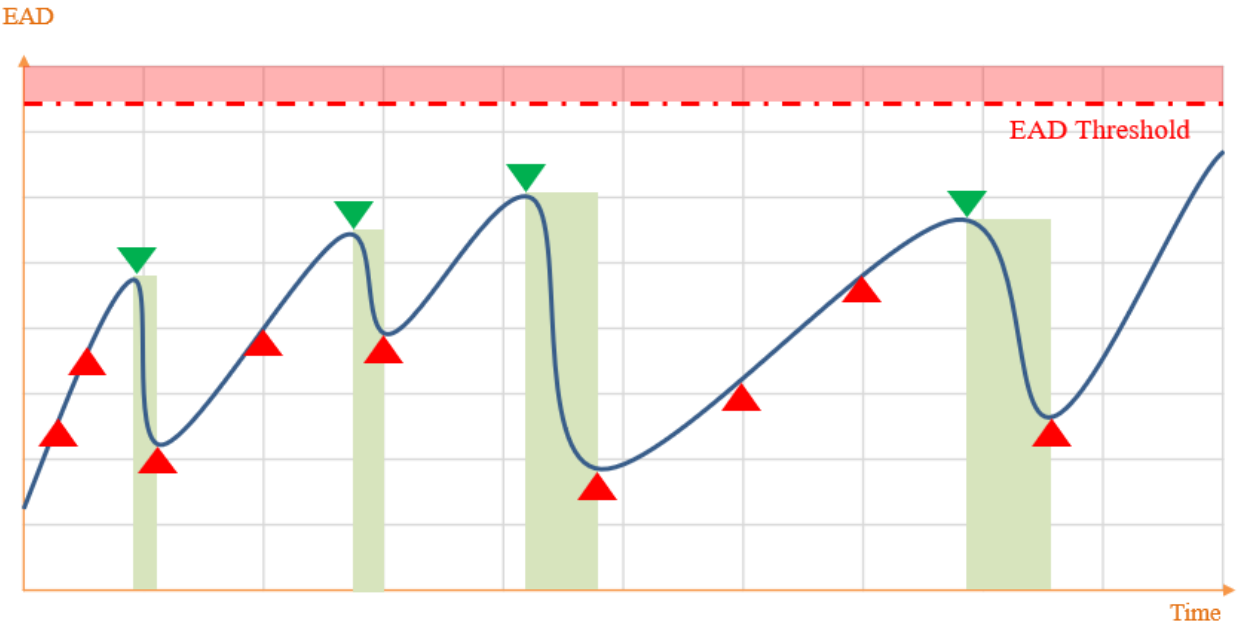


Because change events are continuous and inevitable, it is essential to establish ongoing monitoring of both organizational and environmental shifts. These observations must inform regularly defined and executed architecture correction cycles, ensuring that EAD remains within manageable boundaries.

[2] Note that "EAD Threshold" is a rather conceptual limit, not a definitive measurable amount of EAD. It can be characterized by organization's impotency to perform its business as usual and delivering value to its stakeholders, due to accumulated unaddressed EADs.



This culminates in a continuous and dynamic cycle of EAD-increasing change events and EAD-decreasing corrective actions over time. The overarching objective of the EAD management process is to maintain EAD at a sustainable level—always below the organization's defined EAD threshold.

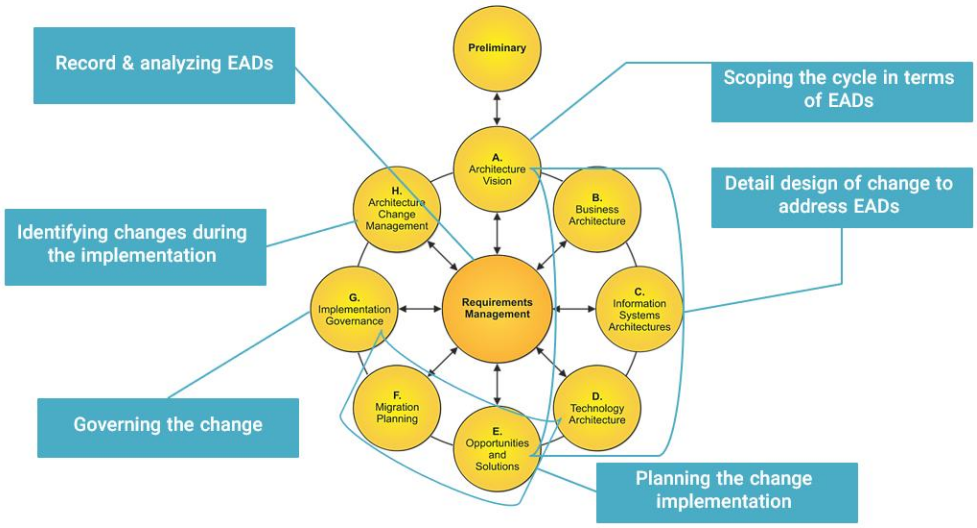


5. How EA Debts relate to EA cycles?

One of the most illuminating aspects of the EAD concept is its potential to reframe how we define, plan, and execute EA change cycles.

As explored in the previous discussion on EAD dynamics (see Part 4), every EA cycle can be interpreted as a strategic treatment plan—a collection of coordinated corrective actions aimed at reducing EAD to an acceptable level.

In this section, I will apply the EAD lens to the TOGAF®[3] Architecture Development Method (ADM) cycle, illustrating how each phase of the ADM can be aligned with EAD management principles to guide more effective and sustainable architectural transformation.



5.1. Phase A: Scoping the change

Every change begins with a vision—and in EA, that vision is often shaped by EADs that require resolution. These addressable debts can be identified either proactively—through continuous architectural monitoring and strategic foresight—or reactively, in response to pain points, failures, or missed opportunities.

In proactive identification of EAD, a major planned change within any EA layer can trigger a cascade of architectural impacts—each potentially giving rise to new EAD items. These must be identified and addressed as part of the upcoming EA change cycle.

For example, consider an organization shifting its business model from Make-to-Order to Make-to-Stock as a strategic pivot. This transition requires re-evaluating the necessary business capabilities—both identifying missing capabilities and assessing those in need of maturity improvement.

Other scenarios that lend themselves to proactive EAD identification include new product launches, organizational restructuring, and enterprise-wide ERP implementations. In each case, architectural dependency analysis can reveal the likely EADs resulting from the planned change, enabling the design of an EA cycle tailored to mitigate them effectively.

Reactive identification of EAD can stem from a variety of sources, including internal and external environmental monitoring, ongoing EA requirements management, and periodic EA compliance

[3] TOGAF is a registered trademark of The Open Group.

reviews (see Section 4.6). In addition, EA health scans or architecture maturity assessments may surface unaddressed debts requiring treatment.
Organizations with mature Enterprise Architecture Management (EAM) or Enterprise Architecture Governance (EAG) capabilities typically implement systematic mechanisms to detect, analyze, and prioritize EADs as part of their continuous architecture oversight.

Whether identified proactively or reactively, the results of EAD analysis can serve as critical inputs to the scoping phase of the EA change cycle—specifically, Phase A: Architecture Vision in the TOGAF ADM. These insights inform the high-level definition of which EA domains require intervention during the cycle.

In the previously discussed examples, this vision might involve: developing or enhancing business capabilities needed to enable a new business model; designing and deploying architecture building blocks to support a new product; re-engineering business processes in alignment with a revised organizational structure; or adapting multiple EA layers to accommodate the introduction of a new ERP solution.

5.2. Phases B-D: Designing the change

The Architecture Definition phase—comprising Phases B (Business Architecture), C (Information Systems Architectures), and D (Technology Architecture) of the TOGAF ADM—focuses on designing the target architecture in alignment with the vision established in the preceding phase.

This stage involves the development of a comprehensive set of architectural models that not only reflect the desired future state but also demonstrate alignment with the articulated strategic vision.

Critically, the EADs identified earlier provide essential input to this process. These documented debts help shape the design by highlighting architectural gaps, constraints, and remediation priorities, ensuring that the resulting models are both aspirational and grounded in architectural reality.

Moreover, the process of designing the target architecture can itself reveal new instances of EAD. In some cases, specific architecture building blocks (ABBs) may be intentionally omitted from the target architecture due to practical limitations such as resource constraints, risk tolerance, or organizational readiness.

For example, a software application intended to support an emerging business capability might be deferred from the target application landscape because the associated capability is still at a low maturity level. While this exclusion may be justified in the short term, it introduces a known EAD item that should be documented and tracked for future resolution.

Architecture building blocks (ABBs) that are deliberately excluded from the target architecture due to practical constraints should be explicitly identified and documented as EADs. This ensures that these deferred elements are not overlooked, but rather acknowledged as part of the architectural backlog and prioritized in subsequent EA change cycles for future remediation.

5.3. Phases E-F: Planning the change

Once the target architecture is defined, the Architecture Building Blocks (ABBs) should be translated into Solution Building Blocks (SBBs), informed by market realities and constrained by non-technical factors such as budget limitations, resource availability, and implementation risks.

A migration strategy must then be established—taking into account both internal conditions (e.g., organizational readiness, interdependencies) and external factors (e.g., regulatory constraints, vendor timelines). Where appropriate, transition architectures should be devised to serve as intermediate steps toward the target state.

Ultimately, the migration plan is formalized as a structured set of programs, projects, and activities that orchestrate the execution of change in alignment with the overall enterprise strategy and EAD management objectives.

Throughout Phases E (Opportunities and Solutions) and F (Migration Planning) of the TOGAF ADM, additional EADs may be identified and documented. A common source of such debts involves replacing ideal Solution Building Blocks (SBBs) with more feasible yet suboptimal alternatives, primarily due to non-technical constraints such as limited budget, resource availability, or implementation risk.

For example, an advanced cloud-based application platform might be excluded from the solution architecture due to insufficient infrastructure readiness or heightened risk associated with data migration. Although this substitution is pragmatically justified, it introduces a known EAD that must be tracked for future remediation.

5.4. Phase G: Change implementation governance

To successfully implement the desired changes, the programs and projects defined in the migration plan must be executed in a controlled and coordinated manner. To ensure alignment with the target architecture's principles and specifications, it is essential to apply appropriate process controls throughout the execution phase.

Architecture compliance reviews serve as a key mechanism for enforcing these controls, helping to verify that implementation activities remain consistent with the architectural intent and do not introduce new EADs.

In certain cases, it may not be feasible to revise design specifications, even when architectural nonconformities are identified during implementation. When such deviations are deemed acceptable—due to time, budget, or strategic constraints—they should be formally recorded as EADs and stored in the EA repository.

Documenting these accepted nonconformities ensures transparency, enables traceability, and allows them to be considered for remediation in future EA change cycles.

5.5. Phase H: Change management

Since the implementation period for architectural changes can be prolonged, it is essential to establish mechanisms to monitor, identify, analyze, and respond to potential internal and external changes that may affect the target architecture and migration plan. These activities are conducted during Phase H of the ADM and effectively constitute a process for identifying and analyzing EADs.

Detected change events can be classified into three categories:

1) Irrelevant to the target architecture.
2) Relevant and requiring immediate attention.
3) Relevant but suitable to be addressed at a later stage.

Change events falling into category (3) give rise to instances of EAD.

5.6. Requirement Management

Requirements management is a continuous process involving the identification, evaluation, prioritization, and documentation of architecture change requirements. EADs constitute an integral part of the overall set of architecture requirements and should be maintained within the architecture repository, managed consistently alongside other types of requirements.

The outputs of Requirements Management, including EADs, serve as inputs to all phases of the ADM, from Phase A to Phase H.

Architecture change cycles—such as those defined by the TOGAF ADM—can be more effectively understood when framed in terms of EAD and the associated process of EAD management. Adopting this perspective on EA cycles also supports the shift from traditional, project-based EA development toward a modern, continuous enterprise architecture management (EAM) capability.

6. EA Debt Management and EA Governance

We can view Enterprise Architecture Governance (EAG) as a system of process controls that oversee architectural change initiatives. These controls are supported by organizational structures—such as Architecture Review Boards (ARBs)—along with enterprise-wide policies and metrics designed to ensure alignment with EA principles and target states.

Architectural change initiatives may include:

- Designing or redesigning the organizational structure
- Developing or improving business processes
- Building application or technology services
- Designing or modifying data models
- Deploying technology infrastructure

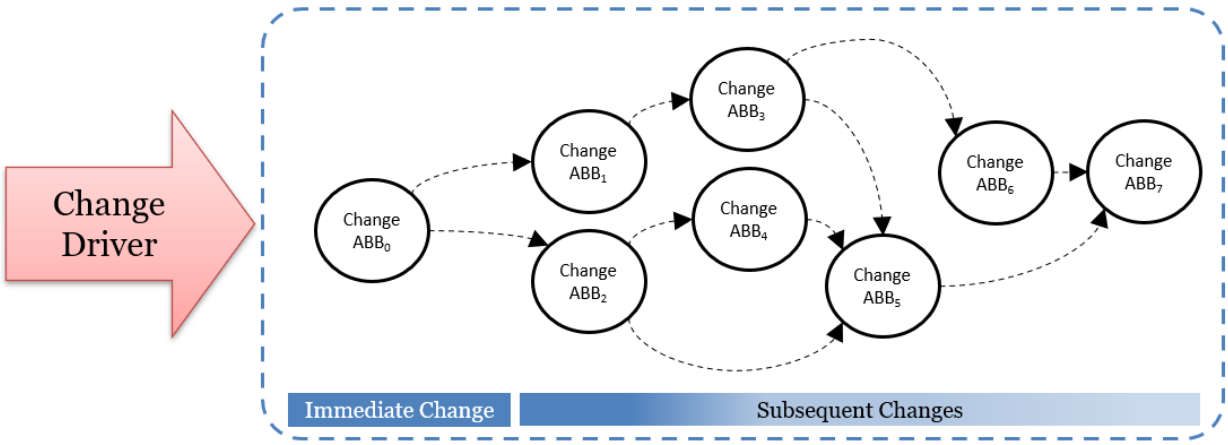
A closer examination of these processes reveals that EAG controls are closely aligned with enterprise architecture debt management practices.

6.1. Anatomy of a change process

Every architectural change is triggered by a stimulus or change driver, originating either internally or externally. Examples include a company's new cost-cutting strategy, a government regulation, or the emergence of a transformative technology trend. In reality, change drivers are often complex mixtures of both internal and external forces.

In response to these drivers, the architecture must be adjusted by modifying one or more architecture building blocks (ABBs). The initial response typically involves a limited set of ABB changes—referred to as immediate or first-tier changes. Examples of such responses include eliminating a redundant, high-cost organizational unit, implementing new process controls to meet regulatory requirements, or rolling out a new technology service.

Following these initial changes, further adjustments are usually required to maintain architectural soundness and consistency. For instance, eliminating an organizational unit may necessitate reassigning its functions to other units and updating related process roles accordingly.

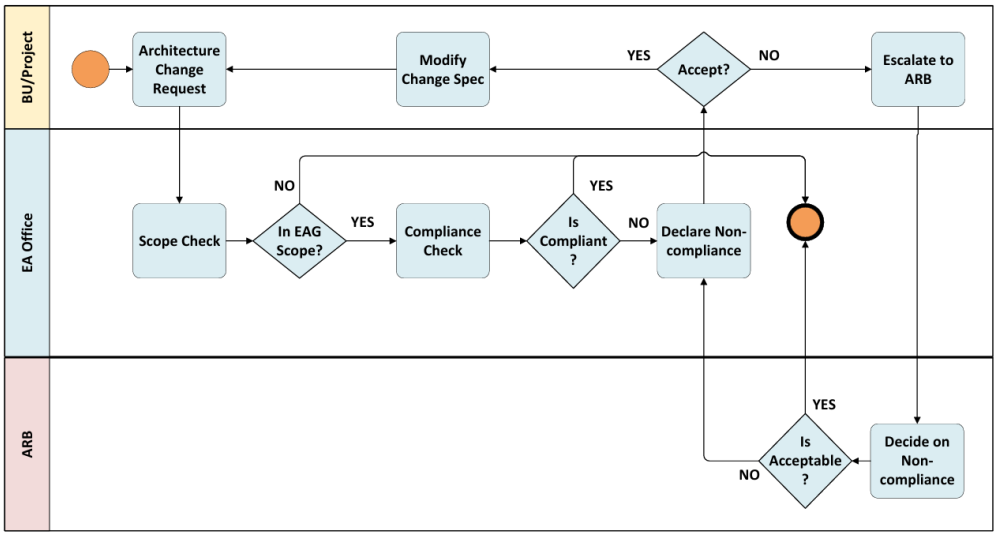


This chain of building block changes must be completed to ensure architectural consistency and operational efficiency. In practice, however, many organizations struggle to follow through on these changes. Common obstacles include limited resources, time constraints, or deficiencies in EAG practices.

Although most EA debt stems from unaddressed changes, some arise from incomplete chains of required architectural updates. A systematic approach to change process controls—central to effective EA governance—is essential for preventing these incomplete change chains that contribute to EA debt.

6.2. A generic process for architecture compliance reviews

Despite variations in how organizations structure their architecture compliance reviews, a generic process can be envisioned as illustrated in the following figure.



The process typically begins with an incoming change request from any business unit or project within the organization. This request may include a design specification for a new or revised business process, a technology service, or a redesign of the organizational structure.

The request is usually evaluated by the organizational unit or roles responsible for EA services, to ensure compliance with EA principles and alignment with target architectural designs. Depending on the organizational model, this role may be fulfilled by a centralized EA management (EAM) office, a network of decentralized EA teams, or delegated EAM representatives embedded within program or project teams. In cases where compliance is in question, a top-level governing body—typically an Architecture Review Board (ARB)—is responsible for final decision-making

Typical steps in the process are as follows:

1. Scope Validation of Change Request
   The incoming change request is evaluated to determine whether it falls within the scope of EAG. Minor changes—or those related to domains or segments not governed by the EA framework—are classified as out-of-scope. These are acknowledged and confirmed without proceeding to further review steps.
2. Compliance Assessment Against EA Standards
   If the change request falls within the scope of EAG, it is then evaluated against predefined EA principles and relevant target architecture designs. To ensure a thorough and consistent review, structured compliance checklists are typically employed during this stage.
3. Confirmation of Compliance
   If the proposed change is determined to be compliant with EA principles and target architecture, the review process concludes with a positive outcome. The change request is approved and can proceed to implementation without further escalation.
4. Issuance of Non-Compliance Report
   If the proposed change is found to be non-compliant with EA principles or target architecture, a non-compliance report is issued to the requester. This report outlines the identified gaps or misalignments and provides guidance for resolving them. The requester is then expected to revise the proposal to address the noted non-conformities before resubmission.

5. Revision and Resubmission
   If the requester accepts the non-compliance findings, the change specification is revised accordingly to address the identified issues. Once updated, the revised request is resubmitted for a new cycle of compliance review.
6. Escalation to Governing Body
   If the requester chooses to maintain the original change request despite identified non-compliances, the matter is escalated to the Architecture Review Board (ARB) or another designated governing authority. This body is responsible for reviewing the dispute and making a final decision regarding the change request's approval, rejection, or conditional acceptance.
7. Governing Body Evaluation and Exception Handling
   The ARB, or equivalent governing authority, evaluates the requester's justification for proceeding with the change—particularly in cases involving constraints such as time, resources, or market demands. The board determines whether, from a strategic and organizational perspective, it is beneficial to approve the change despite non-compliance with enterprise architecture. This decision typically involves conducting risk analysis, architectural trade-off assessments, strategic alignment evaluations, and other investigative techniques to ensure informed, balanced judgment.
8. Rejection and Return for Revision
   If the ARB rejects the requester's appeal, the change request is returned to Step 4. The requester must then address the outstanding issues outlined in the non-compliance report before resubmitting the revised proposal for review.
9. Approval and Process Conclusion
   If the ARB approves the requester's appeal, the change request is accepted despite prior non-compliance concerns. The review process concludes with a positive outcome, and the change may proceed to implementation in accordance with the final decision.

As previously noted, there are several valid reasons why a non-compliant change request may be approved under exceptional circumstances. These include:

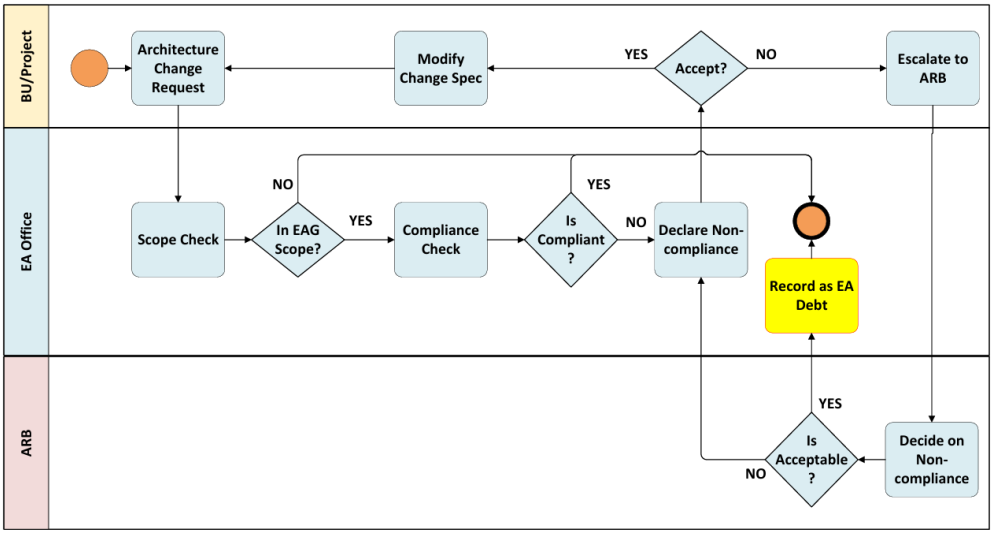
- Time Constraints
  An urgent need to implement a business-critical change may not allow sufficient time to design a fully compliant solution.
- Resource Limitations
  A lack of adequate resources—financial, technical, or human—may prevent the organization from achieving compliance within required timelines and budgets for a critical initiative.
- Market Limitations
  The absence of timely and cost-effective solutions in the market may make it impractical to implement a fully compliant design without jeopardizing business competitiveness.

6.3. Incorporating the EA Debt into architecture compliance reviews process

Any architecture compliance review process presents a valuable opportunity for identifying instances of EA debt. Specifically, when a non-compliant change request is approved—typically under justified exceptions—an EA debt item is created. Therefore, it is appropriate to

incorporate a “Record as EA Debt” activity between the ARB’s approval and the conclusion of the review process. This addition ensures formal documentation and tracking of the debt instance.

The updated review process, including this step, is illustrated in the following figure.



This activity captures instances of emerging EA debt at the moment of creation, ensuring that the resulting EA debt catalog remains current, accurate, and reflective of the organization’s architectural landscape.

6.4. Continuous EAD discovery and EA Repository

The ongoing EAD discovery mechanism described in the previous section highlights the opportunity for EAM teams to systematically store EA debt instances within the organization’s EA repository. In fact, these debt instances often represent a significant portion of the broader set of EA requirements—elements that are traditionally maintained in EA repositories for tracking, analysis, and future remediation planning.

The EA debt component of the EA repository can serve as a valuable input for planning future EA change cycles. It becomes especially relevant when the accumulated EA debt approaches the organization’s defined threshold, or when an upcoming EA cycle presents an opportunity to address and incorporate specific EA debt instances into the scope of change.

7. Research ideas

The EAD concept and its management is relatively novel in EA literature, both in academia and industry. There is still room for more explorations of the idea itself and its application to practical EA use-cases.

Here are some insightful ideas for research in this field, classified by different stages of the EAD management cycle:

7.1. Definition of EAD

- Alternative definitions of EAD
- Mapping of EAD to EA frameworks (TOGAF, …)

- Mapping of EAD to ISO/IEC 420xx series of standards

7.2. EAD identification

- Automatic EAD discovery
- EAD discovery processes design
- AI-enabled EAD identification

7.3. Classification of EAD

- EAD taxonomy design
- EAD classification by events

7.4. EAD measurement

- EAD measurement in terms of relationships between EA elements
- EAD measurement in terms of consequences
- Automated EAD measurement
- AI-enabled EAD measurement

7.5. EAD Analysis

- Effects of EAD in organizations key performance indicators (empirical study)
- Real cases of EAD in organizations (case study)
- Relationship between EAD and Enterprise Risk Management (ERM)
- Relationship between EAD and Quality Management (QM)
- Relationship between EAD and Information Security Management System (ISMS)
- Relationship between EAD and IT Services Management (ITSM)

7.6. EAD Management

- EAD management processes design
- Definition of practical EAD management use-cases.

8. Resources

[Cunningham-92] *The WyCash Portfolio Management System* (https://c2.com/doc/oopsla92.html)

[Dagstuhl-16] *Report from Dagstuhl Seminar 16162 Managing Technical Debt in Software Engineering Edited by P.Avgeriou, Philippe Kruchten, Ipek Ozkaya, and C. Seaman, 2016*

[Daoudi-2023] Sara Daoudi, Malin Larsson, Simon Hacks, Jürgen Jung, *Discovering and Assessing Enterprise Architecture Debts*, Complex Systems Informatics and Modeling Quarterly (CSIMQ), Article 192, Issue 35, June/July 2023

[Hacks-19] Hacks, S., et al. *Towards the Definition of Enterprise Architecture Debts*, in 2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW), 2019, IEEE

About the Author

Reza Karami is a seasoned, hands-on Enterprise and Business Architect with over 25 years of experience in IT consulting and more than 30 years in the IT industry. He is the co-founder and managing partner of Golsoft, an IT and Enterprise Architecture consulting firm based in Iran. In recent years, Reza's work has focused on developing the Capability-based Enterprise Architecture (CBEA), a practical and unified approach to Enterprise and Business Architecture.

FEATURE ARTICLE

Connecting the Dots:

Why it is vital for Enterprise Architects and Business

By Jackie O'Dowd

Connecting the Dots is vital, and it starts with data

Every transaction, every business activity and interaction generates data and information that shapes and impacts strategy, operational activity, and competitive insights. Yet, for many organisations this critical asset remains disconnected, underutilised, inconsistent, and in many cases unseen. This results in missed opportunity, hidden risk, time-lag between analysis and action, and in some cases blind trust. A recent survey by Procore of the Civil Construction industry highlighted some of the most concerning aspects of data management and use in that particular industry. The survey results found that there is a general low level of confidence in project data (81% of respondents) and large volumes of data often go unused (75% of respondents).

In this paper we explore how we can better connect the dots and improve the utilisation of data to drive improved outcomes.

In today's economy data is no longer a byproduct of business, it is the lifeblood of business. Large networks of devices and machines gather and transmit data. New technologies provide real-time or near real-time analysis, leading to faster exploration, explanation and insight. So how can organisations better utilise their data assets and how can enterprise architects use business data to design, build, and deliver connected, effective enterprises?

Being able to connect the enterprise data holding and transform it into trusted, usable intelligence is getting easier. It is not merely a compliance requirement, or a function of IT, it is now a strategic capability that every business needs to enable the unlocking of the value contained within existing information and intelligence systems.

It is my experience that organisations that treat data as a core business asset have the ability to react and respond faster to market and regulatory changes and use trusted accessible data at the core of all activities. In a world where increasing automation and AI driven data supported insight is the new currency.

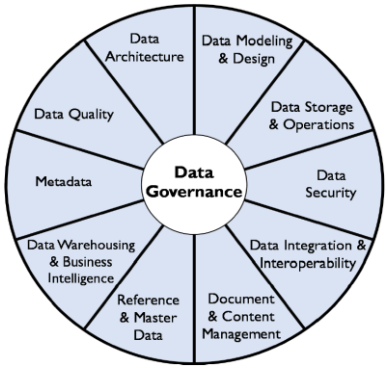
The growing connectivity and interdependence of the systems on which we depend means we need to have confidence in and be able to leverage the data we hold and use. The thesis of this paper is that new data models are needed in an increasing digital and artificial intelligence world. It highlights the value of using data analysis and visualisation to deliver new governance models and business value.

Building quality data

New governance frameworks are needed to provide guidance on what is needed to manage the increasing complexity that surrounds all things data, information, automation and digital intelligence.

Without an adaptable governance framework, analytics and AI outputs risk becoming fragmented, misunderstood, or even weaponised to serve narrow agendas. A strong governance model establishes decision rights, enforces transparency, and creates shared trust in the data, models, and processes used.

DAMA (Data Management Association) produces a data management body of knowledge which can be helpful in determining what specific data management activities an organisation needs to improve and to achieve effective governance.



Ref: The DAMA Wheel

The DAMA wheel provides a structural framework that can be used as a guide for the design and

As artificial intelligence (AI) and advanced analytics become increasingly embedded in workflows and business decision making, governance becomes even more critical. These newer technologies operate at speed and scale, producing insights and recommendations that can have far-reaching implications. Yet, without oversight, they can amplify biases, obscure accountability, and introduce new systemic vulnerability and risk.

A new data visualisation paradigm for enterprise architecture

Enterprise architecture is an essential enabler, and every business has one. It impacts how data is captured, stored and used. Today we are seeing an increasing emphasis on the need for, and use of, real-time data and enterprise architecture views. The static abstract models of the past are no longer adequate to keep up with the fast pace of change and demand for information within the average business. There is a pressing need to better understand the systems in use and leverage the data within them, how they are Implemented and used, the business processes they support, and the impact on the business if they fail or are breached. This increasing need to better understand the systems in play is leading to the use of technologies such as graph models, process mining and digital twins.

Having used graph technologies for many years, we have found when enterprise architecture data is visualised as a graph model, we are able to see the inherent relationships and linkages between elements such as systems, business processes, people, technology stacks and platforms, system dependencies and environments. Patterns can be easily seen and explored. Observations can be quickly formed, highlighted, communicated and investigated. When data is visualised in this way the complex interweave that is the organisation becomes immediately visible.

In traditional enterprise architecture diagrams relationships can appear linear and siloed, whereas a graph model shows:

Context at multiple levels simultaneously business, application, technology, data.

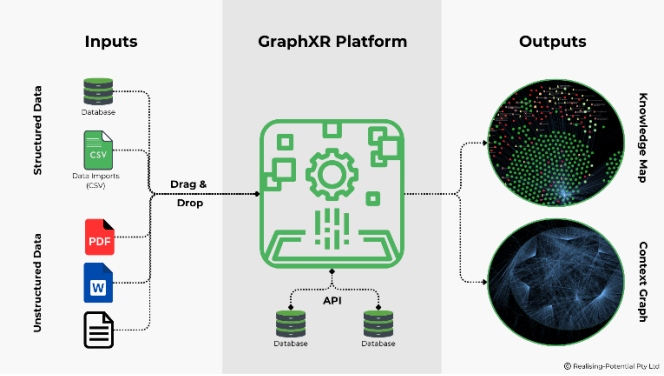
Patterns and anomalies that are often invisible in spreadsheets and static diagrams.

Redundancy revealing duplicated or similar systems.

Complexity of the systems, process, people and data in play.

Relationships and Linkages and how things actually connect.

There is a process, as there is for most things, for getting data from a static, tabular or text-based format into a graph model.

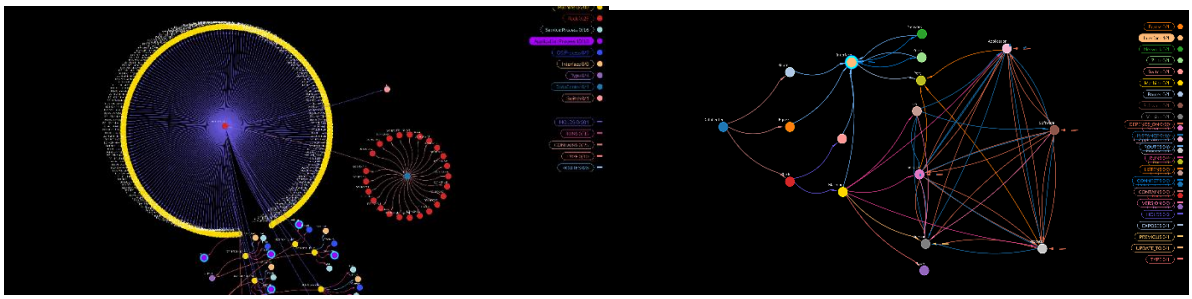


1. Confirm the problem to solve or questions to be answered.
2. Drag and drop/Import the data.
3. Determine best model view(s)
4. Search, observe, navigate and explore.

This analysis and visualisation capability, we believe, is critical as business and systems architectures grow, evolve and become more autonomous and agentic. It provides accelerated visibility, faster learning and new ways of applying governance and oversight. It brings the organisations systems, data and capability to life.

Being able to visually see the technology and systems build, and their connections and dependencies allows us to better appreciate the spread and complexity involved. When this level of visibility and understanding is available design, maintenance, and change efforts can be prioritised and any ripple effects clearly identified. This becomes inherently valuable in instances such as the one shown in the model below.

The two images below are from an interactive model generated by GraphXR and shows the construct of a particular data centre architecture.



In this instance the datacentre was being upgraded to increase performance and adapt to customer needs. This involved swapping out hardware, improving routing and implementing new monitoring

capabilities. Having the datacentre construct modelled in this way helped the team to minimise downtime and service disruption.

When systems are being upgraded, automated, fail or are breached in some way, being able to understand what is impacted is vital. Imagine walking into your data centre, not physically, but digitally. You can see every rack, every machine, every switch, and router, instantly mapped in context. One click shows you exactly which operating systems are running and where, and what applications are deployed, along with the business services they support.

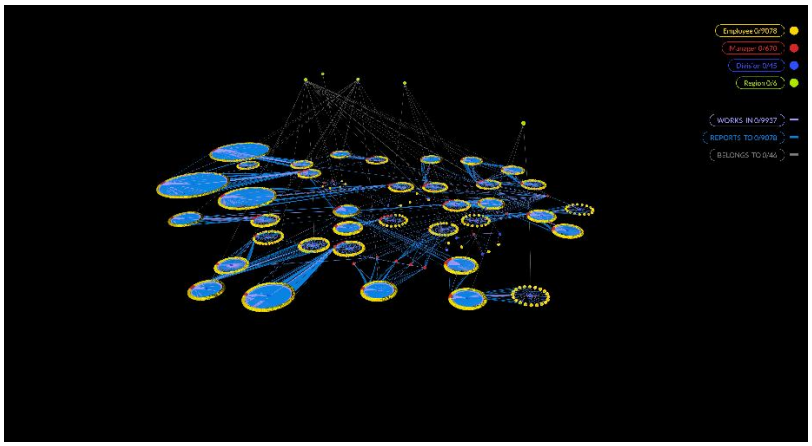
Imagine tracing a single router outage and seeing, in real time, the ripple effect through machines, apps, and critical capabilities. No guesswork. No static diagrams. Just a living, interactive model that shows you the *reality* of your architecture.

It's not just another diagram. It is a navigable, searchable map of your entire infrastructure stack, from the physical layer to the application layer. You can link the enterprise architecture back to the business outcomes it enables. It answers the questions you usually spend weeks chasing: Where are the risks? What is redundant? What happens if this changes? Who may be impacted by any change?

With GraphXR, you do not just *document* architecture, you *see* it, *explore* it, and *trust* it for decisions. This is enterprise architecture brought to life.

Connecting People and Systems

In every enterprise people and systems are almost inseparable, but the connection between them is often unclear or poorly understood. Enterprise architects often struggle to connect who the organisation actually is (people, roles, teams), with what is needed to enable it (infrastructure, applications, data). This GraphXR model shows the organisational structure, the operating locations, the roles, people fulfilling those roles and most importantly the interaction.



By linking the data centre graph to the organisational structured graph, we can start to determine which roles or people own, manage, or depend on specific systems, and capabilities.

This organisational graph unlike traditional organisation charts, doesn't just show reporting lines, it highlights the connections, regions, locations, people, and roles and the technology and capabilities they depend on. With it, you can instantly see who owns what, who will be impacted by a change, and where organisational risk hotspots sit. When combined with the data centre and systems view, it gives you a people-to-technology view, linking the human side of the enterprise to the infrastructure that enables it.

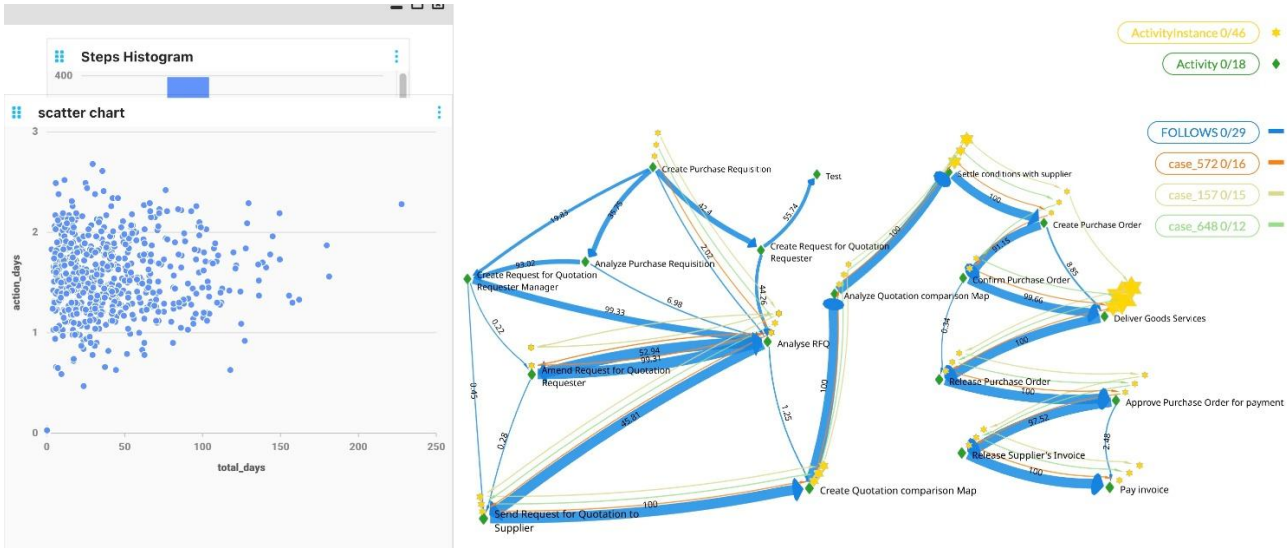
What it means in real terms is that when a data centre, rack, machine or application becomes unavailable the architect can quickly identify what regions, locations, and roles need to be informed.

This type of view can also help to plan technology and systems landscapes that support the evolving, ever changing business landscape. It can also show where specific decision-making authority or knowledge is concentrated, whether that is with a specific role or individual or a team. This is critical for understanding governance and compliance, and where accountability spans multiple domains.

Human Resources (HR) typically have organisational structure data in isolation and in many instances the ability to export the data is limited. Yet, it is a valuable information. By combining HR data with process and technology data we can start to achieve a more holistic view of the organisation.

Bringing Business Process Data to Life

Having used GraphXR for many years, we have found it to be a powerful data analytics and visualisation platform and it is incredibly useful for bringing XES (eXtensible Event Stream) Event Logs to life. These files are created by most enterprise resource planning systems (ERP) to expose how systems are actually used within the organisation. An XES file is a structured event log that uses an XML format to store events. Insights can be gained from the non-linear, dynamic and deeply intertwined data once analysed and visualised. The following model depicts data from an XES file from an Enterprise Resource Planning System (ERP).



From the file content we were able to discover the process flow from 900 cases and 22,000 records. A case is represented in the graph as a workflow path. Each Node within the graph depicts an activity or event and the Edge (the link between Nodes) shows the transition or sequence. The benefit of using GraphXR over more traditional process mining tools is it clearly shows the extent of the relational complexity and has the added benefit of bringing together structured and potentially additional unstructured data into the analysis, something typical process-mining applications don't offer.

If the structures, technology infrastructure, and business process models were linked, we have the start of an enterprise view. For architects this unlocks significant productivity gains because it limits the number of meetings and workshops that are needed to gather information, and it reduces the

fragmented manual effort that is needed to stitch these various perspectives or views together. We have found that time saving of up to 50% can be achieved on activities such as discovery, analysis and documentation.

Right now, architects are probably spending days and weeks pulling disparate data sources together, just to answer a simple question “Who or what will be impacted if we change X”? With this level of graph visualisation, it means architects spend less time chasing data and holding workshops, and more time designing the future and helping the business achieve its strategic objectives and goals.

The table below shows efficiency gains made by our team over the last two years on commercial projects.

Activity	Traditional Approach	GraphXR Approach	Productivity Gain
Dependency Mapping	Weeks of preparing and validating charts, processes, and systems from different teams	Use GraphXR to see all linked People, Processes, Systems and Infrastructure	Time reduced by up to 50%
Impact analysis	Multiple cross functional stakeholder meetings, information consolidation, risk workshops	Visualisation of who, what and how is impacted across the various domains Analysis of the quantitative and qualitative impact	From a duration of weeks and days to hours Contextual data based analysis
Change Planning	High levels of effort to identify all downstream and upstream affects, often misses hidden dependencies	GraphXR shows affected roles, processes and systems	Significant reduction in rework and improvement in communication
Business case preparation	Manually preparing reports that current and future state and the ROI	Generate views that show how the enterprise architecture supports and enables the business	Reduced prep time by up to 40-50%
Governance and Compliance Checks	Painstaking mapping and verification for audits and regulatory reporting	Visibility of ownership, data flows, and compliance sensitive nodes	Audit readiness in real-time
Knowledge transfer and onboarding	Heavy reliance on abstract models, documentation and team knowledge	Interactive exploration and accelerated learning	Ramp up time reduced by up to 60%
Scenario simulation	Multiple workshops, spreadsheets and abstract models to model “what-if”	Query the graph to instantly simulate impacts of organisation, technology or process changes	Scenario planning in minutes and hours depending on the data available.

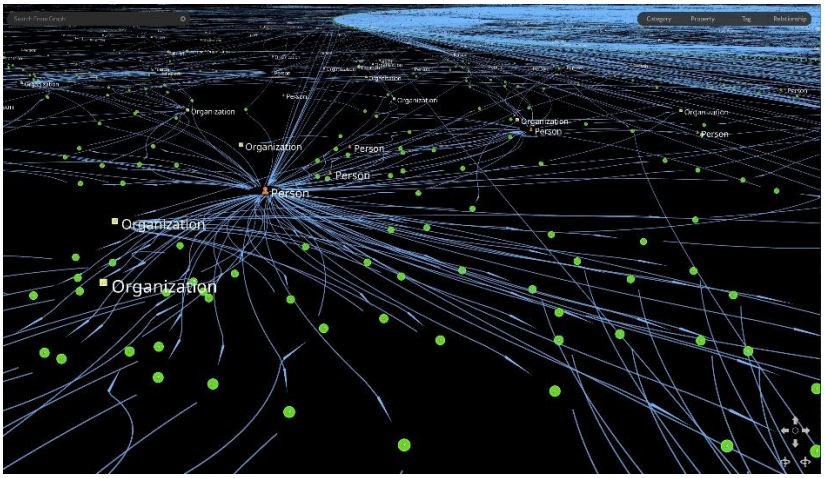
In practice, we have found using GraphXR to combine people structures, infrastructure, and business process workflows into a single graph platform gives enterprise architects a living, real-time view of the entire enterprise. It can eliminate weeks of manual discovery, reduce the risk of hidden dependencies, and allow instant answers to complex “what-if” questions. Architects can focus on designing better future-state architectures, improve alignment between strategy, technology and business needs, and accelerate automation and transformation with greater confidence. It turns enterprise architecture from a slow to react, static documentation exercise into a dynamic capability.

For enterprise architects, an integrated, interactive view of systems, process, people, and policy reveals the organisations network effect, and how interconnected elements, amplify compounding value or create unintended consequences.

Structured and Unstructured data

The vast amount of documentation that business leaders, analysts and architects need to be across can be overwhelming at times, policy and procedures, audit reports, business cases, meeting notes, the list is endless. Significant blocks of time are spent piecing together various documents and understanding the relationships between people, systems, and process, most of this is manual activity that is not scalable and can be error prone.

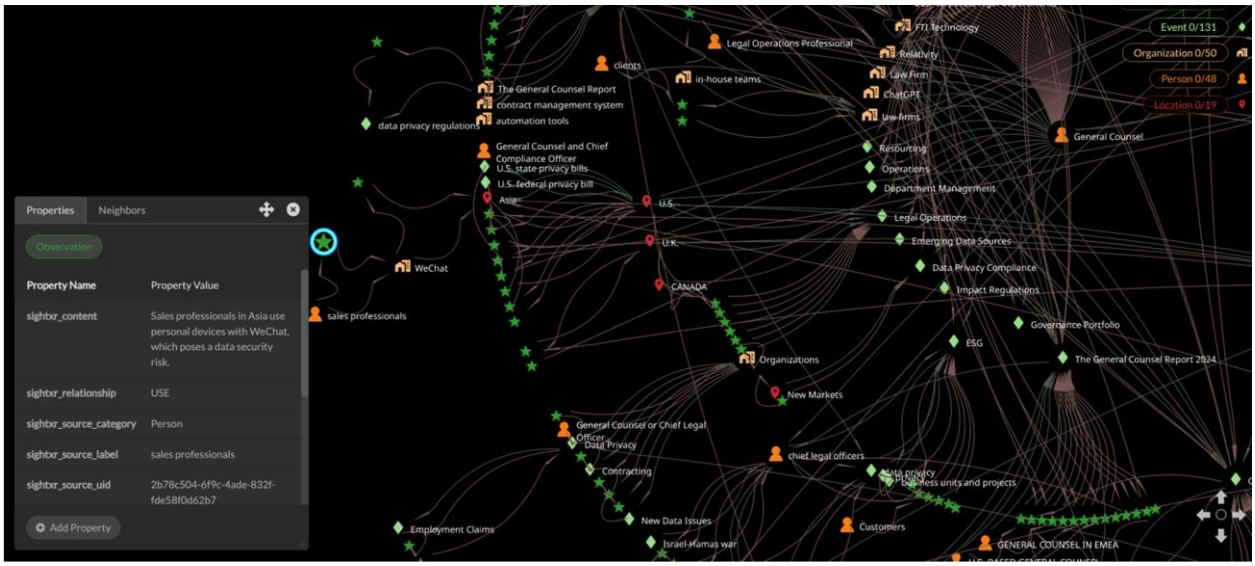
This is where GraphXR and new database technologies can be both time and cost effective. By graphing structured and unstructured data the user has the ability to explore, navigate and query document and log file content, and see it in context.



This integrated approach accelerates discovery, impact analysis, governance, and change planning by reducing discovery and reading time. It enables deeper situational awareness by connecting the dots to better understand the interplay of people, organisations, locations, and events. Instead of taking days and weeks to search and reconcile disparate documents, the user can accelerate learning and uncover valuable insights. It reduces the lag time between discovery, design, and action by being able to not only see the data in context but to use the same data to produce both simple and sophisticated scenario simulations and what-if analysis.

A New Paradigm

This new data and architecture discovery, analysis, and visualisation paradigm helps to elevate enterprise architects from curators of information to providers of insight and transformation. By providing an integrated, interactive view of systems, people, process, data, and workflows, it allows architects to move beyond simply producing abstract diagrams for current and future state to focussing on shaping the future state with greater confidence. It democratises business and architectural knowledge through the sharing of interactive models, making them accessible and understandable across the various architecture teams and stakeholders across and beyond the organisation. It helps to transition enterprise architecture to a more strategic, adaptive discipline that is capable of evolving as fast as the technology and business it enables.



Enterprise architectures are often built around specific technology stacks, capability models and business processes. As greater levels of automation and artificial intelligence are deployed, traditional static architecture artefacts are no longer sufficient. Interactive models such as the one above are more suited to showing contextual information. The model above shows the identification of a risk exposure due to the use of a non-approved application on a personal device.

As the use of agentic systems increase, they will introduce not just new capabilities, but new and different dynamics. As artificial intelligence evolves, agentic AI is set to provide autonomous capabilities in ways not seen before. These agents can perform single tasks on behalf of a specific user, or even another system. This requires a mindset shift, a repositioning of enterprise discovery, design, management and governance in order to accommodate the transition from being the gatekeeper of design to the enabler of strategic, adaptive, intelligent organisations.

Which brings to mind an old saying - *the future is not an unknown land into which we totter unsteadily one day at a time, but an extension of the present that we shape by our decisions and our actions. The future is not somewhere we are going but something we are creating*.

But before we create the future state, we need to discover and clearly understand the current state, what is done, where, when, how and by whom. Now is the time to enter the new paradigm and take advantage of GraphXR and tools like it to see and analyse data, workflows, systems, connectivity and relationships to design, build and manage more effective architectures.

About the Author

Jackie O'Dowd is the Founding Partner and CEO of Realising-Potential, based in Perth Western Australia. She advises and consults across multiple sectors, including government, legal, mining, and civil construction. She works with executive and leadership teams to improve and optimise organisational performance and enterprise architectures. She also provides expert opinion for organisations that have ICT legal disputes to settle or questions to answer.

CALL FOR SUBMISSIONS

by Darryl Carr, EAPJ Editor

The Enterprise Architecture Professional Journal welcomes contributions in its fields of interest, which are enterprise, business, application, information, integration, technology and security architecture, as well as the strategic management of business and technology transformation. EAPJ publishes peer-reviewed material that advances its fields of interest and supports the careers of its readers.

EAPJ combines the strengths of peer-reviewed technical journals and professional newsmagazines. EAPJ invites submission of academic, feature, opinion, and interview articles. The editorial staff also considers other submissions, such as images, interactive graphics, video, and animations. Successful submissions contain actionable information that enhances the capabilities of professionals working within the EAPJ fields of interest.

Each issue consists of one or more main articles, centered on a theme introduced by the Editor's Welcome. Main articles are generally no more than 5,000 words in length, but can be longer in certain circumstances, with body text preferably interspersed with numerous callouts, graphics or tables.

EAPJ encourages submissions, readership and community participation from qualified individuals representing the widest possible variety of geographical regions, cultures, backgrounds and beliefs. Authors must properly attribute all referenced content and ensure that their submissions do not infringe upon any copyrights or intellectual property laws if published in the EAPJ. EAPJ encourages potential authors to contact the editor early on to receive guidance on developing material with the greatest likelihood of publication.

EAPJ also seeks expert reviewers to work with the editor and authors on developing and selecting main articles for the journal.

Please send expressions of interest, submissions, questions, ideas or comments to editor@eapj.org. Potential authors and reviewers should introduce themselves by describing their background briefly, supplying a resume or CV, or referencing an online profile.

You can also submit ideas for publication on our website. Visit https://eapj.org for details.