Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 258, Issue 46, March/April 2026, Pages 67–96
https://doi.org/10.7250/csimq.2026-46.04

Advancing Enterprise Architecture Debt: A Work System Theory Perspective on Modernization and Resilience

Ada Slupczynski[1], Beate Krauze[2], and Simon Hacks[3⋆]

[1]Research Group Software Construction, RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany
[2]Institute of Information Technology, Riga Technical University, Kipsalas str. 6a, 1048 Riga, Latvia
[3]Department of Computer and Systems Sciences, Stockholm University, Borgarfjordsgatan 12, 164 55, Stockholm, Sweden

slupczynski@swc.rwth-aachen.de, beate.krauze@rtu.lv, simon.hacks@dsv.su.se

Abstract. Enterprise Architecture Debt (EAD) refers to the accumulation of architectural misalignments and suboptimal decisions that hinder an organization's agility, performance, and long-term sustainability. While previous research has primarily approached EAD as an extension of technical debt, its broader organizational implications require a stronger theoretical foundation. This article extends prior work by systematically evaluating thirteen theories from the information systems domain to assess their suitability for conceptualizing EAD. Using a structured comparison framework, each theory is analyzed along key dimensions. In addition, this study introduces modernization and resilience as two critical outcome areas affected by EAD. These dimensions are essential for understanding how architectural decisions influence an organization's capacity to adapt and evolve in a dynamic environment. The analysis confirms the strengths of Work System Theory (WST) in capturing the socio-technical nature of EAD, while also identifying complementary perspectives across alternative theories. The findings contribute to a more robust theoretical foundation for EAD, supporting the development of strategies for its identification, evaluation, and management.

Keywords: Enterprise Architecture Debt, Foundational Theory, Work System Theory, Modernization, Resilience.

1 Introduction

Enterprise Architecture Debt (EAD) has been defined as *"the deviation of the currently present state of an enterprise from a hypothetical ideal state."* [1] It arises when decisions in Enterprise Architecture (EA) are delayed, or shortcuts are taken, leading to problems. These problems can hinder an organization's ability to adapt, perform effectively, or achieve its objectives. However, in available sources, EAD is not grounded in any theory but is based on the concept of technical

⋆ Corresponding author

Reference: A. Slupczynski, B. Krauze, and S. Hacks, "Advancing Enterprise Architecture Debt: A Work System Theory Perspective on Modernization and Resilience", Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 46, pp. 67–96, 2026. Available: https://doi.org/10.7250/csimq.2026-46.04

Additional information. Author ORCID iD: A. Slupczynski – https://orcid.org/0000-0001-5403-7720, B. Krauze – https://orcid.org/0009-0005-9630-2234, S. Hacks – https://orcid.org/0000-0003-0478-9347. PII S225599222600258X. Received: 31 December 2025. Accepted: 9 March 2026. Available online: 30 April 2026.

debt [1], making it difficult to study or manage effectively. A strong theoretical foundation can help by identifying EAD, making it easier to measure and analyze. It also ensures that decisions about managing EAD are based on evidence rather than intuition.

A theoretical foundation also improves how we study EAD [2]. This enables researchers to examine its causes and effects in a structured manner and identify patterns across various organizations. This also enables the development of more effective strategies for managing EAD and for learning from others' experiences. Additionally, having a theory allows EAD to connect with other fields, such as IT management and organizational strategy, for a more comprehensive understanding.

Work System Theory (WST) [3] is an excellent fit for studying EAD, as it examines the relationships between an organization's people, processes, technologies, and information. EAD often arises when these elements are not aligned [4], [5], and WST can help explain how these misalignments occur and how they can be fixed. Another reason WST is a good choice is that it focuses on how systems change over time. EAD is not static; it grows or evolves as organizations adapt to new challenges [6]. WST's focus on change makes it well-suited to study the dynamic nature of EAD. Finally, WST emphasizes the role of stakeholders – people and groups affected by or involved in a system. EAD often involves trade-offs between stakeholders, such as balancing short-term cost savings against long-term system performance [7]. WST provides a framework for examining these trade-offs and their impact on the organization.

This article extends earlier work [8] by broadening the theoretical basis for conceptualizing and analyzing EAD beyond a single lens. Whereas the original article primarily applied WST to categorize illustrative instances of EAD, we contribute (1) a structured assessment framework and (2) a systematic comparison of thirteen IS theories along a shared set of dimensions grounded in IS literature. The resulting comparative view surfaces conceptual blind spots in how EAD has been theorized to date and clarifies where WST provides strong explanatory leverage and where alternative theories offer complementary or competing accounts.

We also extend the assessment framework with two additional dimensions, *modernization* and *resilience*, to reflect concerns that are increasingly salient in enterprise architecture practice. Importantly, these dimensions are not presented as entirely new theoretical constructs; rather, they synthesize and operationalize themes that recur across prior work on digital transformation, legacy evolution, and organizational responses to disruption. By making modernization and resilience explicit evaluation criteria, we enable a more practice-relevant and future-oriented appraisal of theoretical support for EAD, capturing both the organization's capacity for long-term renewal and its ability to withstand and recover from shocks (e.g., supply chain disruptions and cybersecurity threats).

The rest of this work is structured as follows: Section 2 provides an overview of EAD (including consideration of resilience and modernization) and a comparison of theories. Section 3 presents the mapping between EAD and WST. Section 4 applies WST to real-world examples, illustrating how EAD manifests across different system components. Section 5 identifies key research gaps in the related literature, followed by future directions. Finally, Section 6 concludes the article.

2 Foundations

2.1 Enterprise Architecture Debt

The increasing pace of digitalization and the widespread adoption of agile methods have significantly impacted how organizations manage their EA. One of the core challenges lies in the time available to define robust target architectures, as product owners tend to favor short-term business value over long-term architectural sustainability [9]. Simultaneously, there remains a scarcity of approaches that effectively support long-term architectural planning [10], [11].

Therefore, Hacks et al. [1] introduced the concept of Enterprise Architecture Debt (EAD) by extending the idea of Technical Debt, formulated initially to describe technical shortcuts that hinder

future IT development [12], [13], to the enterprise level. While Technical Debt has proven valuable in identifying software deficits, guiding decision-making, and raising awareness [14], [15], its focus remains mainly on isolated systems. This narrow scope often neglects the broader architectural concerns that span entire organizations [16], [17], [18].

EAD is "the deviation of the currently present state of an enterprise from a hypothetical ideal state" [1]. This deviation may result from short-term decisions that increase the future cost of change or from shifts in strategic direction that render previous architectural decisions suboptimal. In both cases, EAD acts as a hindrance to achieving an updated, strategically aligned EA. To support a shared understanding of the terminology in this emerging field, Slupczynski and Hacks [19] have developed a domain ontology that captures and structures key concepts in EAD. Research on EAD has evolved along two principal streams [6]: (1) technical aspects, which focus on architectural artifacts and tooling, and (2) socio-technical aspects, which consider stakeholder dynamics, organizational processes, and human decision-making.

In the technical stream, Salentin and Hacks [20] introduced the concept of EA Smells, a set of indicators for architectural inefficiencies. These were operationalized through a prototype that could identify such smells in ArchiMate models. Smajevic et al. [21] further advanced this work by developing an automated tool to support EA Smell detection.

Regarding the management of EADs, Yeong et al. [22] proposed a prioritization method based on portfolio theory and utility functions, enabling organizations to evaluate and sequence their debt remediation activities. Building on this, Liss et al. [23] introduced refactoring strategies to eliminate identified debts. Complementing these approaches, Slupczynski et al. [24] proposed a process model for evaluating the prudence or recklessness of architectural debts, thus offering a decision-support perspective.

The socio-technical dimension of EAD is addressed in studies focusing on process integration and stakeholder engagement. Alexander et al. [6] proposed a management framework for EAD that includes identifying, collecting, assessing, prioritizing, and resolving debts. Jung et al. [4] contributed a workshop format designed to identify EADs and EA Smells not easily captured by models alone. This format was later refined by Daoudi et al. [5] for improved time efficiency and impact assessment.

To empirically assess the utility of the EAD concept, Hacks and Jung [25] conducted a controlled experiment in which students modeled a fictitious organization. While the experiment did not yield a measurable improvement in EA outcomes, it represents an important first step in evaluating the practical impact of EAD as a conceptual tool.

There are also similar works to consider: Chis et al. [26] explain how WST concepts can be lifted into an RDF knowledge graph that is continuously updated with runtime data from operational systems. By representing participants, information, and technologies as semantically linked nodes, the graph becomes a living digital twin of the enterprise work systems. Such a twin enables architects to query cross-layer dependencies (e.g., "Which customer-facing capabilities still rely on an unsupported database engine?") and reason over them using off-the-shelf SPARQL and inference engines. In an EAD context, this offers two benefits: (1) fine-grained visibility of "hidden" debts that span several work systems, and (2) the ability to monitor debt interest automatically as the graph evolves with every change to the underlying landscape.

Flórez et al. [27] survey and re-implement more than 50 automated analyses for enterprise models, ranging from cost optimization to change-impact estimation. When expressed in an extended ArchiMate metamodel, the catalog highlights the additional metadata a model must contain for a given analysis to run correctly. From an EAD standpoint, the contribution is two-fold. First, several analyses (e.g., redundancy detection, technology obsolescence, workload bottlenecks) map directly onto known "EA smells" that signal debt. Second, the catalog clarifies input and output requirements, paving the way for repeatable, tool-supported debt dashboards rather than ad-hoc spreadsheet calculations.

Flórez et al. [28] present an approach that turns ArchiMate models into a living laboratory for automated enterprise analysis. Each "analysis method" is packaged as a plug-in that declares the extra attributes it needs (e.g., age, MTTR, license cost) and can be hot-deployed into their ArchiMate tooling. Because the metamodel itself is extensible, the architecture repository evolves in lock-step with the battery of analyses a team wants to run. For EAD, this matters on three levels: (1) Detection—plug-ins such as technology-obsolescence or redundancy-finder act as debt sniffers the moment the required metadata exists; (2) Quantification—the plug-ins return counts of affected elements (principal) plus impact-propagation graphs that approximate interest; and (3) Prioritization, architects can chain several plug-ins to compute composite risk scores, ensuring that remediation budgets attack the highest-yield debts first. The architecture, therefore, shifts EAD management from an ad-hoc, spreadsheet-based exercise to a repeatable, continuously extensible pipeline.

2.1.1 Resilience in Enterprise Architecture

Resilience has become a defining quality for organizations whose core business activities depend on complex, continuously evolving digital architectures [29]. In information systems research, resilience is typically understood as an organization's ability to anticipate, absorb, recover from, and adapt to disruptions while continuing to deliver essential services [30]. Within the context of EA, this capability extends beyond technical robustness and reflects the coherence of the entire socio-technical system. Resilience is shaped by the degree to which architectural artifacts remain aligned with the organization's strategic and operational needs. Misalignments caused by deferred decisions, fragmented processes, outdated technologies, or inconsistent information structures gradually accumulate as EAD, which in turn constrains the organization's ability to respond effectively to change. As a result, resilience becomes not only a matter of EA design but a direct outcome of how this debt evolves over time. Consequently, resilience must be understood systemically, acknowledging how decisions, trade-offs, and EAD influence the organization's behavior as a whole [30].

2.1.2 Modernization through EA Lens

Modernization is a corrective process used when traditional maintenance methods are unable to serve their purpose [31]. Modernization remains a significant challenge for enterprises that rely on large, complex legacy systems. Such systems remain business-critical and are expected to remain in operation. These systems typically exhibit high levels of accumulated technical and enterprise architecture debt, which increases the cost and complexity of maintaining legacy systems. Due to its corrective nature, intended to help the system meet its current needs [32], it can be viewed from the perspective of EAD as a mechanism to address the structural misalignment described in that context. Before modernization, debts had to be repaid to ease the complexity of modernization. During modernization, debts must be continuously monitored and repaid to make prudent decisions and avoid introducing new, unnecessary debts into the modernized system. Finally, after modernization, debts need to be monitored, documented, and regularly repaid to ensure that the next modernization is not unnecessarily large or complex. EAD thus plays a significant role at every step of modernization.

In the context of EA, other terms such as migration, evolution, or digital transformation are well established; thus, it is important to clarify the focus on modernization. Each of the related terms carries a distinct connotation:

- **Migration** denotes the replacement of an existing information system's components to reach a target system [33], whereas modernization focuses on the degraded state and the need for corrective measures. Migration may be perceived as a method for modernization, but modernization may utilize other methods.

- **Evolution** denotes continuous change, assuming that the system is fulfilling current requirements [34], whereas modernization assumes the existence of a gap between AS-IS and the TO-BE, which constitutes EAD.
- **Digital Transformation** denotes change aimed at value redefinition and organizational change [35], whereas modernization focuses on the remediation of accumulated debts.

Modernization is corrective in nature, aimed at bridging the misalignment between what is required and what the information system can deliver [36], making it the most appropriate term in the context of EADs. In this context, modern refers primarily to alignment with current requirements, rather than adoption of the newest technology.

2.2 A Theory for Enterprise Architecture Debt

2.2.1 Theory Comparison Criteria

This section introduces and explains the criteria used to compare theories relevant to analyzing EAD. Selecting a suitable theoretical lens requires more than checking thematic alignment. It involves assessing whether the theory provides the right concepts, structures, and mechanisms to describe and address the challenges posed by EAD. The criteria are derived from established literature in information systems (IS), EA, and technical debt. Each category highlights a dimension that is essential to understanding how EAD arises, how it is represented, and how it may be managed. By using a structured set of criteria, we ensure that comparisons between theories are both transparent and grounded in theoretical relevance. The following subsections provide a detailed description of each comparison category.

Representation of Architectural Elements. This criterion assesses whether a theory can effectively represent the key elements of EA, including business processes, technologies, data, people, and their interdependencies. EA aims to offer a holistic view of an organization by modeling these elements and their relationships [37]. The ability to represent architectural structures is essential for identifying where debt may accumulate and how it manifests across the enterprise landscape. A theory that lacks architectural constructs may overlook core aspects of EAD, particularly those that relate to misaligned or fragmented system components [38].

Support for Debt Management. This criterion refers to the extent to which a theory can support the identification, monitoring, and remediation of debt over time. Originating from the concept of technical debt [12], EAD requires not just a descriptive framework but also practical mechanisms to manage trade-offs and guide resolution. Theories that include decision processes, feedback loops, or lifecycle concepts are better suited for managing EAD because they align with the dynamic and cumulative nature of debt [14]. Without support for debt management, a theory may describe problems but offer little in terms of resolving them.

Stakeholder Involvement. This criterion assesses whether a theory incorporates constructs that capture the roles, interests, and interactions of stakeholders. EAD often results from trade-offs made by different stakeholders with conflicting priorities [7]. Theories that reflect human and organizational behavior, such as roles, motivations, and collaboration, enable a better understanding of why architectural misalignments occur. In particular, the inclusion of stakeholder perspectives supports participatory approaches to identifying and prioritizing architectural debts [4].

Handling of Technical Complexity. EAs are complex systems composed of many interrelated layers and components [37]. This criterion evaluates whether a theory supports reasoning about this complexity. Effective EAD analysis requires an understanding of how local issues propagate through interconnected systems and layers. A theory that simplifies or ignores these interactions may not provide an adequate explanation for how debt impacts broader organizational performance.

Support for Dynamic Change and Evolution. EAD is not static; it evolves, shifts, and transforms as enterprises grow and change [39]. This criterion examines whether a theory can model the time dimension of architectural change, including the accumulation or remediation of debt over time.

Theories with lifecycle models or support for feedback and adaptation are better equipped to capture this dynamic behavior. Without a temporal perspective, theories risk treating EAD as a one-off problem instead of an ongoing architectural challenge.

Ability to Model Decision-Making and Trade-Offs. EAD often results from explicit or implicit decisions to prioritize short-term gains over long-term architectural soundness [1]. This criterion assesses whether a theory facilitates the modeling of such trade-offs and decision-making processes. A theory that incorporates concepts such as actor goals, cost-benefit analysis, or scenario evaluation is more suitable for analyzing EAD, as it can explain how and why compromises are made [40]. The ability to reason about trade-offs also supports forward-looking decisions, helping organizations balance immediate needs with strategic alignment.

2.2.2 Comparison of Theoretical Frameworks for EAD

To identify the theory most suitable for representing EAD, a range of established theories from the IS field have been analyzed based on their key characteristics. In his work on Work System Theory, Alter [3] refers to several foundational IS theories, including general systems theory, socio-technical theory, actor-network theory, organizational routines, soft systems methodology, and activity theory. He also argues that UML can be considered a theory in this context. In addition, other IS-related theories have been studied that Alter [3] did not include, such as grounded theory, institutional theory, affordance theory, contingency theory, and chaos theory. Table 1 summarizes the key characteristics of each theory in this context, assessed along standard criteria drawn from the IS literature (Section 2.2.1): representation of architectural elements, support for debt management, stakeholder involvement, handling of technical complexity, support for dynamic change and evolution, and ability to model decision-making and trade-offs. These criteria reflect the specific demands of modeling and managing EAD, particularly its long-term implications and the need for inclusive, systemic analysis.

Activity Theory [41] is a socio-cultural framework that examines human actions within an activity system (subject, object, tools, rules, community, division of labor) in context. In an EAD analysis, Activity Theory can shed light on how day-to-day practices and context contribute to architectural issues, potentially exposing root causes of debt by comparing the present activity state to desired goals. However, it offers little in the way of formal *Architectural Representation*: it cannot model technical components or infrastructure in detail, limiting its ability to depict the structure of an EA or the explicit points where debt resides. This gap means that the theory provides only modest *Debt Management Support*, as it lacks mechanisms to monitor how technical decisions accumulate debt or to track the long-term impact of those decisions on the architecture. On the other hand, Activity Theory inherently emphasizes *Stakeholder Involvement* by focusing on human participants and their contextualized activities, ensuring that user and practitioner perspectives are considered when analyzing EA-related problems. This approach does not explicitly address *Technical Complexity Handling* beyond broad considerations of tools and artifacts in activity systems; it struggles to capture complex interdependencies in a multilayered architecture. Similarly, its support for *Dynamic Change* is indirect; the theory acknowledges that activities evolve and that contradictions can spur change, but it does not provide a structured method to model evolving architectures or the accrual of "interest" on debts over time. Finally, in terms of *Decision-Making and Trade-Offs*, Activity Theory is more descriptive than prescriptive: it can contextualize why certain short-term architectural compromises were made by highlighting the goals and motivations of actors, but it does not offer formal guidance for evaluating or balancing architectural trade-offs to manage debt.

Actor-Network Theory (ANT) [42] conceptualizes information systems as networks of human and non-human "actors" whose interactions collectively shape outcomes. This socio-technical perspective emphasizes the interplay between people, technologies, and institutional structures. While it draws attention to distributed agency and negotiation in system development, ANT offers

Table 1. Comparison of Theories Applicable to Enterprise Architecture Debt

Theory	**Architectural Representation**	**Debt Management Support**	**Stakeholder Involvement**	**Tech. Complex. Handling**	**Dynamic Change Support**	**Decision-Making and Trade-Offs**
Activity Theory [41]	Task-focused only	Explains behavior causes	Actor/Mediator covered	No EA constructs	Supports develop. cycles	No trade-off tracking
Actor-Network Theory [42]	Network view only	No constructs for debt	Stakeholders as actants	Socio-technical focus	Dynamic relations modeled	No trade-off logic
Affordance Theory [43]	Focus on UX only	No architectural support	Emphasizes users	No technical scope	Static model	Trade-offs absent
Chaos Theory [44]	No structure modeling	Not applicable to debt	Ignores roles	Dynamic complexity handled	Captures change	No decision framework
Contingency Theory [45]	No structural modeling	Contextual reasoning only	Indirect support	Weak interdep. logic	Static view	Context-driven choices
General Systems Theory [2]	High-level abstraction	No debt mechanisms	Stakeholders implicit	Holistic view only	Feedback loops included	No decision modeling
Grounded Theory [46]	Pattern-based, no structure	Qualitative identification	Emerges from data	Limited to context themes	Longitudinal possible	No formal reasoning
Institutional Theory [47]	Organizational view only	Explains inertia causes	Actor roles acknowledged	No system details	Change pressures noted	Context-sensitive choices
Organizational Routines [48]	No EA detail	No support for propagation	Supports recurring roles	Process-centric only	Implies incrementalism	Ignores decision logic
Socio-Technical Theory [49]	No formal architecture model	Social framing only	Strong on user-system dynamics	Lacks layering	Static perspective	Human constraints only
Soft Systems Methodology [50]	Abstract conceptual model	No tooling for debt	Inclusive of multiple views	Focus on organization	Static interpretation	Negotiation emphasis
UML (as Theory)	Formal structural diagrams	Descriptive only	No stakeholder view	Strong on static detail	No change support	No rationale modeling
Work System Theory [3]	Covers core EA elements	Structured debt categorization	Explicitly modeled roles	Socio-technical view	Lifecycle model included	Trade-offs supported

little in the way of formal *Architectural Representation*: it does not include structured modeling constructs for depicting system components, dependencies, or layers in an EA. Consequently, *Debt Management Support* is weak, as the theory lacks methods for tracking architectural decisions over time or capturing how technical compromises contribute to the accumulation of debt. However, ANT provides strong support for *Stakeholder Involvement* by treating both human and non-human entities as actants with influence, thereby emphasizing the roles of diverse agents, technical and social, in shaping system behavior and architectural outcomes. Its approach to *Technical Complexity Handling* is qualitative and descriptive; while it acknowledges the complexity of networks and relationships, it does not offer mechanisms to systematically capture or analyze architectural interdependencies. Regarding *Dynamic Change*, ANT considers systems as inherently fluid and evolving through actor re-alignments, but it does not provide tools to model these changes formally or to assess their architectural consequences. For *Decision-Making and Trade-Offs*, ANT highlights the negotiations and translations between actors that lead to system outcomes, yet it lacks normative guidance or structured frameworks to evaluate competing architectural options or manage trade-offs explicitly.

Affordance Theory [43] originates from ecological psychology and has been widely applied in IS to examine how technology enables or constrains user actions. In the context of EAD it provides a user-centered view of how architectural features are perceived and acted upon. However, it lacks formal mechanisms for *Architectural Representation*, focusing instead on the interaction between users and system functionalities rather than modeling structural or technical components. As such, its contribution to *Debt Management Support* is limited; it does not account for how technical decisions accumulate debt over time or how architectural integrity evolves. It does offer strong *Stakeholder Involvement* by emphasizing user perceptions, actions, and goals, which can be critical when evaluating the usability consequences of architectural decisions. Nonetheless, *Technical Complexity Handling* is minimal, as the theory does not engage with system interdependencies, infrastructure layers, or scalability issues. In terms of *Dynamic Change*, affordances can shift in response to changing user needs or system modifications; however, the theory does not provide formal tools for modeling or managing such change in an architectural context. Finally, its support for *Decision-Making and Trade-Offs* is weak; while it can inform which features are perceived as useful, it offers no framework for evaluating architectural alternatives or making long-term design trade-offs.

Chaos Theory [44] is rooted in mathematics and systems science and is used in IS research to explore non-linear, unpredictable behaviors in complex systems. In the context of EAD, it highlights the sensitivity of system behavior to initial conditions and the potential for emergent architectural outcomes. However, it provides no formal *Architectural Representation*, lacking constructs for modeling architectural layers, elements, or dependencies. As a result, *Debt Management Support* is also absent, since the theory does not include mechanisms for tracing technical decisions or understanding how architectural compromises lead to accumulated debt. It provides little for *Stakeholder Involvement*, focusing instead on system-level dynamics rather than human or organizational roles. For *Technical Complexity Handling*, Chaos Theory contributes conceptually by framing EA as a complex adaptive system; however, it does not offer practical tools for analyzing or managing this complexity in a structured manner. It is more applicable in its support for *Dynamic Change*, emphasizing emergent behavior and the unpredictability of system evolution over time, which aligns with how some EADs may unexpectedly manifest or escalate. However, in terms of *Decision-Making and Trade-Offs*, the theory provides no normative guidance or evaluative tools for making or analyzing architectural choices in light of potential debt consequences.

Contingency Theory [45] is a managerial and organizational theory that argues that there is no one-size-fits-all solution; rather, optimal decisions depend on contextual factors. In the context of EAD, this theory highlights the importance of aligning architectural decisions with specific environmental or organizational conditions. However, it provides no formal *Architectural*

Representation, lacking constructs for modeling system structure, components, or architectural layers. Its value for *Debt Management Support* lies in its ability to explain why particular compromises or architectural shortcuts might be appropriate in certain situations, but it does not offer concrete tools to track, measure, or manage debt over time. Regarding *Stakeholder Involvement*, the theory emphasizes alignment with the organizational context but does not explicitly address the roles or perspectives of specific actors involved in architectural decisions. For *Technical Complexity Handling*, it provides only general guidance, suggesting that more complex environments require more adaptable systems, without offering models or methods to analyze technical dependencies or interactions. Its support for *Dynamic Change* is implicit in its responsiveness to environmental shifts, but it lacks mechanisms to model or simulate changes in architecture or debt progression. In terms of *Decision-Making and Trade-Offs*, Contingency Theory supports situational reasoning and contextual evaluation, yet it does not provide structured methods for making or balancing long-term architectural trade-offs.

Grounded Theory [46] is a qualitative research methodology used to develop theory inductively from empirical data. In the context of EAD, it can be valuable for uncovering patterns, themes, and causal explanations of architectural issues as they emerge from real-world practice. However, it offers no formal *Architectural Representation*, as it does not include modeling constructs or notations suited for describing system architecture. Its contribution to *Debt Management Support* is also limited; while it may help identify sources or impacts of EAD through qualitative analysis, it does not provide tools to monitor or manage debt over time. Regarding *Stakeholder Involvement*, Grounded Theory often includes stakeholder perspectives during data collection and theory generation, giving it moderate strength in this category. It offers little in terms of *Technical Complexity Handling*, since its qualitative approach is not designed to capture detailed system structures, dependencies, or layers. Support for *Dynamic Change* is possible through longitudinal data analysis, which allows researchers to observe how architectural decisions and their consequences evolve over time; however, there is no formal mechanism to model these changes. Finally, in terms of *Decision-Making and Trade-Offs*, Grounded Theory offers no prescriptive guidance, instead focusing on understanding phenomena rather than evaluating alternative courses of action.

Institutional Theory [47] explains how organizational behavior is shaped by institutional pressures such as norms, rules, and cultural expectations. In the context of EAD, it provides insight into how organizational inertia, regulatory constraints, and legitimizing behaviors can influence architectural decisions and the persistence of architectural debt. However, it does not include a formal *Architectural Representation*; it lacks the constructs necessary to model technical layers, components, or system relationships. Its value for *Debt Management Support* lies in its ability to explain why certain debts may persist due to institutional resistance to change or path dependence, though it offers no tools to measure or manage those debts. With respect to *Stakeholder Involvement*, the theory acknowledges the roles of actors within institutional structures, including decision-makers and influencers, but it does not provide a detailed model of their interactions or contributions to the architecture. *Technical Complexity Handling* is minimal, as the theory focuses on organizational rather than technical dynamics. Institutional Theory does support *Dynamic Change* by accounting for how change processes are influenced by institutional pressures and legitimacy concerns, which can affect when and how architectural debt is addressed. In terms of *Decision-Making and Trade-Offs*, the theory helps explain the context and rationale behind decisions, particularly how external expectations or internal norms shape choices, but it does not provide formal methods for evaluating architectural options or balancing trade-offs.

Organizational Routines [48] refer to repetitive, recognizable patterns of interdependent actions carried out by multiple actors within an organization. In the context of EAD, they provide a lens to understand how architectural decisions and technical practices become stabilized over time, potentially embedding debt into the organizational fabric. However, they do not provide

a structured *Architectural Representation*, as routines are typically modeled through narratives or process diagrams rather than system components or architectural layers. Their support for *Debt Management* is indirect: routines can help identify persistent practices that lead to or reinforce debt, but they do not offer tools to measure, visualize, or manage debt accumulation. The theory engages with *Stakeholder Involvement*, as routines inherently involve human actors, and it supports an understanding of how architectural roles and responsibilities are distributed across teams or departments. Regarding *Technical Complexity Handling*, its contribution is limited to the observation of how complexity may be stabilized or obscured through repeated actions, rather than being explicitly modeled. In terms of *Dynamic Change*, routines accommodate gradual evolution, often through variation and selective retention, which can help explain incremental architectural shifts, though without providing structured change management tools. Finally, for *Decision-Making and Trade-Offs*, the theory does not offer normative guidance but may reveal how past decisions are reinforced through habitual behavior and embedded practices.

Socio-Technical Theory [49] emphasizes the interdependence between social and technical systems within organizations. In the context of EAD, it provides a valuable framework for understanding how technical architecture and human factors jointly influence the development and consequences of architectural debt. However, it offers no formal *Architectural Representation*; while it stresses the need to balance social and technical aspects, it does not include modeling constructs for architectural elements or system structure. Its support for *Debt Management* is conceptual, highlighting how misalignments between technical and organizational systems can lead to suboptimal solutions that accumulate debt over time, though it lacks formal tools to track or quantify this process. Socio-Technical Theory is strong in *Stakeholder Involvement*, as it inherently considers users, designers, and organizational roles in system development and maintenance. It contributes modestly to *Technical Complexity Handling* by promoting awareness of layered socio-technical interactions; however, it does not provide explicit analytic methods for managing technical dependencies or complexity. The theory has a static view of system configuration, which limits its ability to model or support dynamic change in EA over time. Similarly, for *Decision-Making and Trade-Offs*, the theory encourages a balanced consideration of human and technical perspectives, but does not provide structured mechanisms for evaluating or managing architectural compromises.

Soft Systems Methodology [50] is an approach developed to tackle complex, ill-structured organizational problems through participatory modeling and iterative learning. In the context of EAD, it offers a means to explore diverse stakeholder perspectives and organizational needs that may contribute to or be affected by EAD. However, it employs abstract conceptual models, such as rich pictures and root definitions, providing only a limited *Architectural Representation* that is not well-suited for capturing detailed system components or technical structures. As a result, its *Debt Management Support* is weak, since it lacks mechanisms to identify, monitor, or quantify architectural debt. It excels in *Stakeholder Involvement*, emphasizing inclusive dialogue, multiple viewpoints, and collaborative understanding of system issues. *Technical Complexity Handling* is indirect: while the methodology recognizes complexity and systemic interactions, it does not offer analytical tools for modeling dependencies or technical architectures in depth. Its orientation is primarily static, focusing on problem exploration rather than modeling dynamic architectural evolution, which limits its utility for addressing *Dynamic Change*. Regarding *Decision-Making and Trade-Offs*, Soft Systems Methodology supports negotiation among stakeholders and reflective inquiry, which can inform architectural decisions, but it does not provide structured decision frameworks for evaluating technical alternatives or managing debt-related trade-offs.

UML is commonly used as a formal modeling language for software and systems design. When viewed as a theory, it provides syntactic and semantic tools for representing structural and behavioral aspects of systems. In the context of EAD, UML provides strong support for *Architectural Representation* through class diagrams, component diagrams, and deployment

diagrams, which can model technical layers, interfaces, and dependencies. However, it provides only descriptive capabilities and lacks built-in constructs for *Debt Management Support*; it does not account for architectural degradation, decision rationale, or the temporal accumulation of debt. *Stakeholder Involvement* is minimal, as UML primarily addresses technical perspectives and does not explicitly incorporate roles, responsibilities, or organizational viewpoints. UML is relatively strong in *Technical Complexity Handling*, as it enables detailed representations of system components and their interrelations, which is useful for identifying technical hotspots that may harbor EAD. That said, UML provides limited support for *Dynamic Change*; while state diagrams and sequence diagrams can illustrate system behavior, UML does not inherently model architectural evolution over time or track the emergence of debt. Lastly, it lacks support for *Decision-Making and Trade-Offs*; UML diagrams do not include a rationale for design decisions or mechanisms for evaluating alternatives in terms of long-term consequences or technical debt.

Work System Theory [3] provides a comprehensive framework for analyzing systems within organizations, where a work system comprises people, processes, information, and technologies that interact to produce products or services. In the context of EAD, it offers structured constructs for understanding how enterprise systems function and evolve over time. It supports *Architectural Representation* by including core elements such as participants, information, technologies, and processes, which can be mapped to architectural concerns. It also provides moderate *Debt Management Support* through structured analysis of mismatches or breakdowns in work system elements, which may correspond to architectural deficiencies or compromises. The theory emphasizes *Stakeholder Involvement* by explicitly modeling roles and responsibilities within the system, facilitating inclusive consideration of how architectural decisions affect users and maintainers. While its handling of *Technical Complexity* is limited to a socio-technical perspective and lacks formal modeling of technical interdependencies, it offers a conceptual foundation for identifying where complexity resides. Work System Theory includes a lifecycle model, offering some support for *Dynamic Change* by recognizing that systems evolve and that changes affect various components and relationships over time. It also accommodates *Decision-Making and Trade-Offs*, as it encourages analysis of system outcomes and alignment between components, though it stops short of providing detailed methods for evaluating design alternatives or debt implications.

Summary. From the theories, Work Systems Theory, General Systems Theory, and Socio-Technical Theory seems to be the most promising fit. WST is characterized as a structured and flexible framework, providing concepts relevant to EAD. While it would benefit from certain adaptations, it seems to require the least among the candidates. The other theories tend to focus on singular aspects of EADs, failing to capture the complexity of EAD as a whole. This analysis reveals that while several theories contribute valuable perspectives, only WST offers a sufficiently comprehensive and adaptable framework for capturing the multifaceted nature of EADs.

Overall, no single theory is universally applicable in the context of IS, highlighting the need to compare different theories within this domain.

2.3 Work System Theory

WST [3] was proposed by Steven Alter as a means to bridge the gap between business and IT stakeholders working on IS by providing a method for describing systems without the often complex IT concepts. Defined based on Gregor's [2] five categories of theories, it is defined as *an integrated body of theory that includes a Type 1 analytical theory (the work system framework) and a Type 2 explanatory theory (the work system life cycle model), which in combination give the basis of a Type 5 design theory (work system model)*.

In 2013, Alter [3] presented an overview of the core concepts, extensions, and challenges related to WST. The core idea of the WST is to consider systems within organizations as the base unit, using Work Systems as a foundation. A Work System is a socio-technological system in which humans

and machines collaborate to produce specific products/services for their customers (cf. Figure 1). To accomplish that, they use information, technology, and other resources.

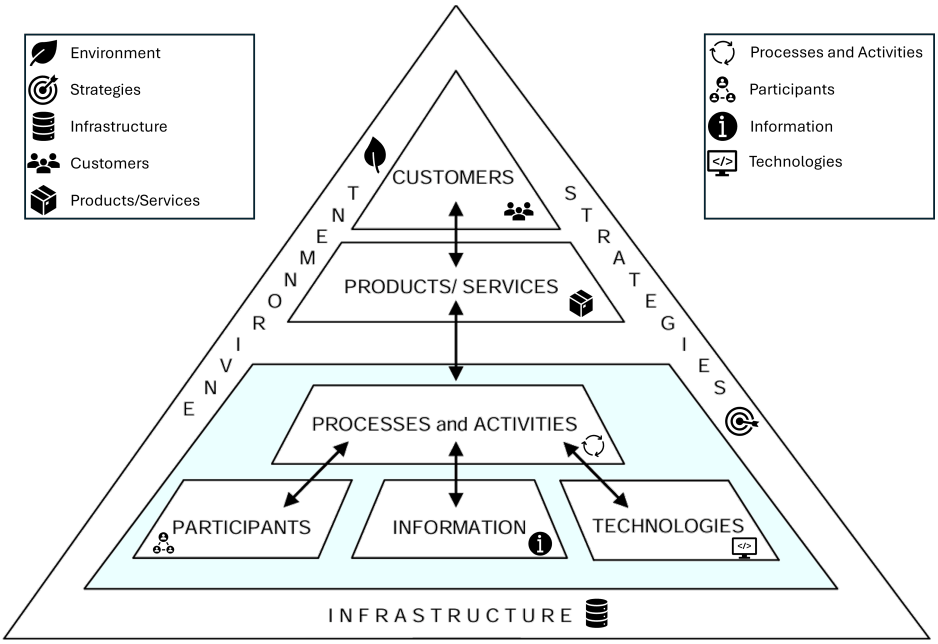

Figure 1. The work system theory framework as proposed by Alter [3], with icons

Nine elements comprise a system [3]: Four of the elements are viewed as inside the work system, defining the system's core functionality. (1) **Processes and activities** are meant to produce products/services for the customers. The processes and activities of a work system define how work is performed (the AS-IS state), not the ideal TO-BE state. (2) **Participants** perform work within the work systems. They are not limited to IT users but also focus on participants who do not use IT systems directly. (3) **Information** entities in the context of work systems are *used, created, captured, transmitted, stored, retrieved, manipulated, updated, displayed, and/or deleted by processes and activities*. There is no distinction between data and information. (4) **Technologies** include both tools used by the participants and automated agents, allowing work systems to be decomposed into fully automated sub-systems.

Two elements are viewed as partially inside the work system, defining entities that interact with the core components of the work system. (1) **Products/Services** consist of information, physical things, and/or actions created for the customers. They should provide benefits to customers who use them. (2) **Customers** receive the products/services and use them for purposes different than work activities inside the work system itself.

Three of the elements are viewed as outside the work system, despite directly influencing the work system. (1) **Environment** describes the *organizational, cultural, competitive, technical, regulatory, and demographic* context in which the work system operates. Factors of the environment might directly or indirectly affect the work system. (2) **Infrastructure** includes *relevant human, information, and technical resources* used by the work system or multiple work systems, which are managed outside of it. (3) **Strategies** *include enterprise strategy, department strategy, and work system strategy*.

WST is explicitly designed to support drill-down and decomposition: a work system can be analyzed at multiple levels of granularity, where an element (e.g., a single process, a participant group, or an information set) can be "isolated" and treated as a sub-work-system while still preserving traceability to the parent work system and its purpose. This drill-down capability is particularly relevant for system-of-systems like settings because it enables analysts to move between enterprise-level views and subsystem-level details without losing the referential links that

explain why a subsystem exists, which other elements it depends on, and how changes at one level affect performance and outcomes at another.

3 Mapping Enterprise Architecture Debt to Work System Theory

EAD refers to the accumulation of compromises, deferred decisions, or shortcuts made in managing EA. This concept originates from the broader notion of technical debt as "the deviation of the currently present state of an enterprise from a hypothetical ideal state" [1]. EAD arises when short-term fixes or decisions to address immediate challenges lead to structural weaknesses or architectural misalignments, resulting in long-term costs and reduced organizational agility.

EAD is a multidimensional concept that impacts key aspects of an organization's systems, processes, technologies, and information flows. Its effects can be observed across various levels, from operational inefficiencies to strategic misalignment [4], [5]. In the following section, we illustrate the connection between EAD concepts and WST:

Participants and customers: EAD often originates from decisions prioritizing short-term productivity or immediate usability for specific teams [4]. Over time, these decisions can create gaps in skills, collaboration, and communication. For instance, reliance on a small group of experts to maintain legacy systems may lead to knowledge silos, reducing the organization's flexibility and resilience.

Processes and activities: Compromises in the design or implementation of processes can lead to inefficiencies or rigid workflows that fail to adapt to changing needs. For instance, quick fixes to streamline one part of a process may cause bottlenecks elsewhere, embedding inefficiencies into the system [51]. These process-related debts can make it harder for organizations to scale or innovate.

Technologies and products: Using outdated, fragmented, or incompatible technologies is a major source of EAD. Quick decisions to implement short-term solutions can lead to systems that are difficult to integrate, maintain, or scale. Over time, the cost of maintaining these technologies grows, and their limitations restrict the organization's ability to adopt innovations [40].

Information: Poorly designed ISs or inconsistent data standards can contribute significantly to EAD. Issues such as duplicate or incomplete data repositories, lack of integration between systems, or misaligned data structures [4], [5] can reduce the quality and reliability of information. These issues often lead to errors, inefficiencies, and lost opportunities for leveraging data-driven decision-making.

3.1 Resilience as a Function of Work System Theory

A WST perspective highlights that resilience emerges from the interaction of participants, processes, information, and technologies, situated within a wider strategic, infrastructural, and environmental context. Disruptions rarely affect a single architectural element in isolation; interdependencies across layers mean that local weaknesses can escalate quickly, reducing the organization's capacity to maintain stable operations during a crisis.

Resilience of a work system is a function of how effectively its social and technical elements, and the relationships among them, enable the system to continue, recover, adapt, and transform under disruption while still producing acceptable products or services for customers (cf. Figure 2). The resilience of the work system increases when the elements of the system are well aligned and when the system has stronger capabilities to monitor, respond, learn and anticipate in both social and technical components [52].

Building on this perspective, the following analysis illustrates how specific forms of EAD embedded within core WST elements constrain resilience in practice.

Processes and activities: Operational resilience depends on processes that support timely detection and response. As highlighted in this article, fragmented workflows, embedded workarounds and legacy process structures accumulate as process-related EAD, limiting the

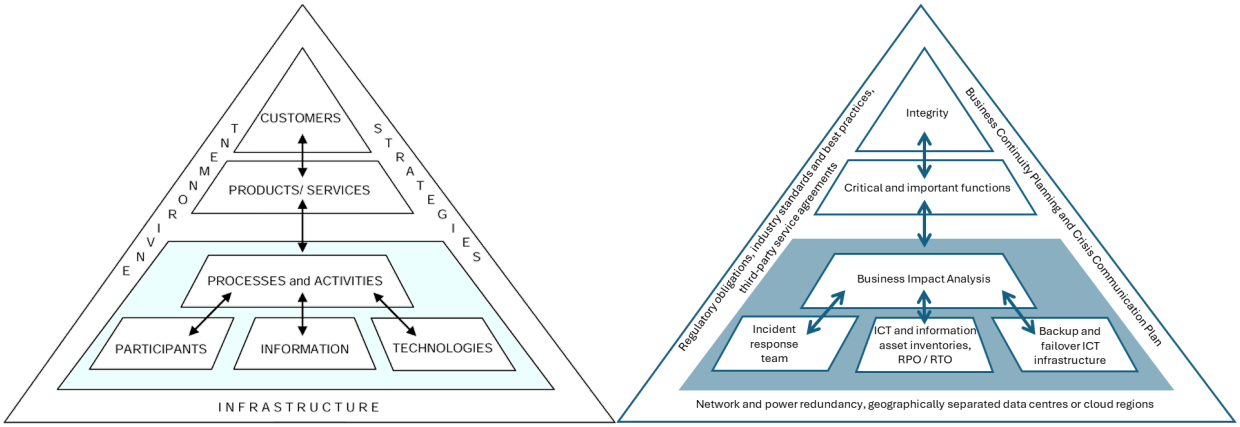

Figure 2. Resilience as a function of Work System Theory

organization's ability to scale or adapt. These inefficiencies slow down recovery, restrict flexibility during disruptions, and weaken resilience.

Participants: Resilience is directly shaped by participants' skills, responsibilities, and interactions. This particular form of EAD arises when legacy roles, knowledge silos, or unclear ownership undermine a coordinated response. For instance, organizations become dependent on a small number of experts or maintain siloed structures, both of which reduce adaptability and increase vulnerability during a crisis.

Information: High-quality information supports situational awareness and informed decision-making. Inconsistent repositories, outdated documentation, and misaligned data structures weaken organizational visibility. This reduces the ability to understand system states, delays response actions, and makes it harder to adapt processes or technologies during a crisis.

Technologies: Technological robustness is a key determinant of resilience. Technology-related EAD, including legacy systems, fragmented tool chains, and limited integration capabilities, restricts automation, increases recovery times, and elevates operational risk. When core technologies lack interoperability or rely on unsupported components, disruptions propagate more easily, reducing overall resilience.

Infrastructure: Infrastructure provides shared services and resources that underpin stability and continuity. As shown in the examples below, EAD accumulates when asset inventories are outdated, shared platforms are inconsistently managed, or common components are fragmented across multiple platforms. Such deficiencies limit the organization's ability to implement resilient practices and hinder compliance with regulatory expectations, including those specific to IT risk management[†].

Environment: External conditions, including regulatory obligations, market dynamics, and technological trends, shape the expectations of resilience placed on the organization. Environmental constraints become linked to EAD when work systems fail to keep pace with changing requirements. For instance, new sectoral resilience regulations may require near-real-time incident reporting, but the organization's legacy processes and information structures are unable to support it.

3.2 Mapping Modernization to Work System Theory

Existing modernization frameworks tend to focus on the technical aspects of modernization, providing valuable insights into the technical or architectural state of the legacy system. However, modernization is not only a technical endeavor, but also requires consideration of business, organizational, and technical perspectives [31], [53]. WST can complement existing frameworks by

[†] Regulation (EU) 2022/2554 of the European Parliament and of the Council of 14 December 2022 on digital operational resilience for the financial sector.

focusing explicitly on people and practices. The information on actual work practices, knowledge practices, stakeholders, and their dependencies gathered with the help of a WST may help identify and reduce risks, improve alignment of AS-IS with TO-BE, and ensure that delivered technical changes are well-received and valuable to various stakeholders.

WST supports modernization before it happens by identifying EADs that hinder modernization and assessing risks related to it. It can also be applied during modernization to ensure the proper alignment of goals and the currently employed strategy, guiding decisions and structuring knowledge. Finally, it can be employed after modernization to assess whether the organizational goals have been met by assessing the current state of the system. In this way, the WST can be employed continuously throughout the entire cycle of modernization, supporting the planning, execution, and assessment of modernization, as well as the more technically oriented frameworks.

As such, WST can be helpful in modernization by identifying degrading elements, establishing relationships between them, and monitoring the state of the system and its components. In the following, we illustrate the connection between modernization concepts and WST:

Environment and Strategies: Modernization must consider external factors such as regulatory obligations and market dynamics, which shape both the urgency and the constraints of modernization endeavors. Unaddressed evolving regulatory requirements may impose compliance risks during the transition period. A clearly defined modernization strategy, aligning technical factors with the external and business ones, helps shape the decisions to prioritize longevity of the business over temporary convenience. However, misaligned strategies risk the introduction of new debt, for instance when prioritizing immediate needs over architectural coherence.

Infrastructure: Modernization benefits from shared infrastructure services by enabling the reuse of common components and consolidating fragmented systems. Such shared services may be disrupted if modernization of connected systems is performed at different times, affecting the dependent systems.

Participants and Customers: One of the challenges of modernization is the risk of knowledge loss when legacy system experts become unavailable. During modernization, knowledge is updated and distributed among relevant stakeholders. This enforces shared ownership of systems, reducing people's risk and encouraging collaboration between various teams. However, modernization projects might face resistance due to the changed roles and different responsibilities.

Technologies and Products: Integration failures of legacy and modernized components, performance bottlenecks, and increased security exposure represent significant risks during modernization. On a technical level, modernization removes outdated and duplicate components, identifies the integration interfaces needed, and encourages modularization of the system. This results in reduced integration costs, enabling scalable development of new products and services.

Processes and Activities: The fragmented, yet complex processes are replaced with well-designed and easily adaptable ones, meant to ease the work on the system. The previously used shortcuts, manual approval structures, and complex, incomprehensible operational flows are being restructured to prevent the accumulation of debt. At the same time, the co-existence of both old and new workflows may cause operational inconsistencies and ambiguous ownership.

Information: Modernization requires a large amount of information about the systems and their domain to be properly executed, often enforcing a re-documentation of valuable information. However, missing or incomplete documentation is a common and significant threat, which may lead to flawed decisions. Modernization enables the restructuring of necessary information, adopting a company-wide documentation standard, and eliminating obsolete data or outdated repositories. After modernization, the data is more reliable than before, supporting informed decision-making on various enterprise levels.

As such, modernization can be seen as a continuous EAD management function. It realigns the system with its desired architecture, repaying the debt incurred from short-term compromises, and implementing actions aimed at preventing the recurrence of EAD accumulation.

4 Work System Theory to Characterize Enterprise Architecture Debt: Examples

We illustrate the application of WST to categorize EADs by five examples. The first two examples are also briefly discussed in the original publication [1]. The examples are illustrated by Figure.

4.1 First Example: Automotive Supplier Specializing in High-End Engine Manufacturing

The first example (cf. Figure 3) focuses on an automotive supplier that specializes in high-end engine manufacturing. Over time, the company has undergone several mergers, resulting in a highly complex organizational structure. This complexity is mirrored in the company's internal processes, which have become inefficient and difficult to optimize. Although management is aware of these inefficiencies, its ability to improve the processes is constrained by collective bargaining agreements that protect employee positions for a set duration. This legal obligation restricts an organization's flexibility to reassign roles, consolidate responsibilities, or automate certain functions—measures that would otherwise improve efficiency and responsiveness.

This situation constitutes a clear instance of EAD. It results from a trade-off between economic efficiency (the desire to streamline operations) and legal obligations (employment protections). Because the organization consciously operates under suboptimal conditions that diverge from an ideal or intended architecture, these trade-offs accumulate as EAD. Explicitly categorizing this situation as EAD helps the organization recognize it as a matter requiring conscious architectural decision-making, rather than merely accepting the resulting misalignments as static, unavoidable constraints.

Using the WST lens, this case reveals debt across multiple components:

- *Processes and Activities*: The core operational workflows are fragmented and suboptimal due to legacy practices inherited from merging organizations. These inefficiencies represent process-related EAD, where operational behavior deviates from the ideal streamlined state.
- *Participants*: Employees are locked into legacy roles and structures due to contractual agreements. This inhibits the evolution of participant roles to better align with current business needs or technological capabilities, contributing to participant-related debt.
- *Technologies*: Each merged entity likely introduced its own IT systems and tools. These may not be fully integrated, leading to technological fragmentation. The result is increased maintenance costs and limited scalability—hallmarks of technology-related EAD.
- *Information*: Misaligned or redundant data repositories across legacy systems degrade the quality and availability of information. This information-related debt impacts decision-making and slows response times.
- *Environment*: The most prominent external factor is the legal environment, namely, labor agreements, that shape and constrain the organization's architectural flexibility. These environmental constraints are outside the immediate control of the work system but heavily influence it.
- *Infrastructure*: Supporting systems and resources, such as shared IT platforms or HR systems, may still be segmented according to the pre-merger structure, impeding efforts toward consolidation and standardization.
- *Strategy*: The organization's strategic objectives may prioritize efficiency and agility, yet the current EA is misaligned with these goals due to legacy constraints and legal frameworks. This results in strategic misalignment debt.

This case exemplifies how multiple, interdependent forms of EAD can accumulate in a large, complex organization. By situating these issues within the WST framework, the organization can more effectively identify where debts reside, understand their causes and impacts, and develop a more structured approach to addressing them in future architectural planning.

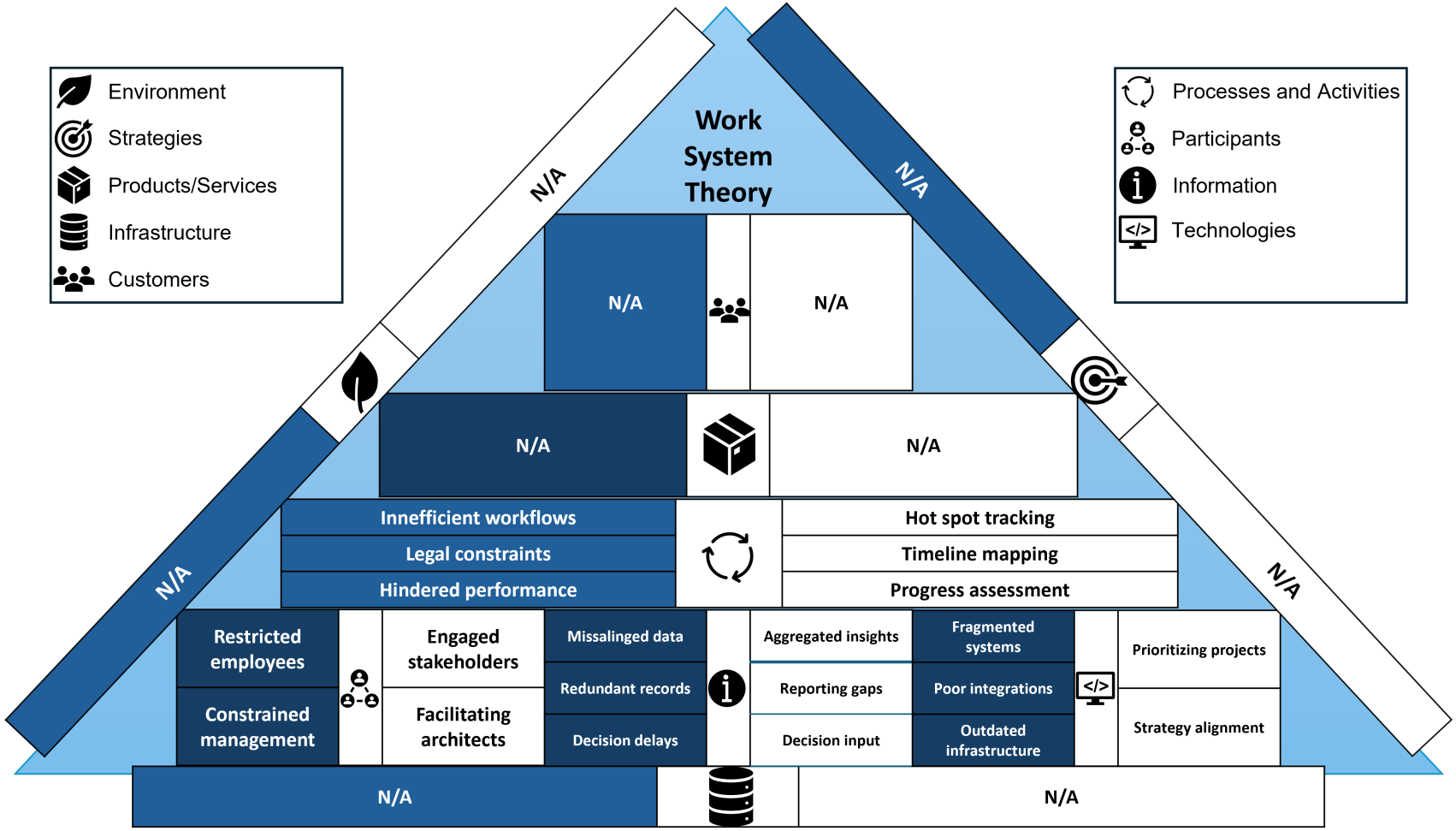

Figure 3. Exemplary illustration of two EADs in the WST framework

4.2 Second Example: A Misalignment between an Organization's Architecture and Strategic Target State

The second example (cf. Figure 3) demonstrates how EAD can manifest as a misalignment between an organization's architecture and strategic target state. While the organization does not explicitly label this gap as "EAD", the underlying issue, an ongoing discrepancy between intended and current architectural realities, fits squarely within the EAD concept. Importantly, this case does not involve technical flaws or system degradation, but rather the challenges of steering an enterprise-wide transformation toward strategic objectives, making the term 'technical debt' inapplicable.

The organization has structured its EA into a set of focal areas, known as hotspots, to operationalize its architectural strategy. Each hotspot corresponds to a key domain within the IT strategy, encompassing areas such as infrastructure and application modernization, data governance, and process redesign. For each hotspot, enterprise architects define and map specific actions onto a timeline, creating a forward-looking roadmap of intended changes.

Progress is monitored through a structured, quarterly feedback mechanism. Architects conduct interviews with stakeholders who are accountable for each hotspot. These interviews yield both qualitative insights and quantitative indicators, capturing the extent to which each domain has progressed toward its target state. The collected data is then aggregated into reports that inform senior management, offering a high-level overview of architectural alignment and surfacing areas where goals are not being met.

This process represents a deliberate attempt to manage strategic alignment debt, a form of EAD characterized by persistent divergence between the designed trajectory of the EA and its practical implementation. When examined through the lens of Work System Theory, this case reveals EAD across several components:

- *Processes and Activities*: The architecture transformation is operationalized through recurring activities, such as roadmap planning, stakeholder engagement, and quarterly assessments. While these activities are structured, the presence of gaps and delays suggests that some processes are not functioning as intended, resulting in process-related debt.
- *Participants*: Stakeholders from across the organization are central to the transformation effort. Their engagement in interviews and their roles in executing change initiatives reflect their importance and influence. Inconsistent ownership or uneven commitment across hotspots could result in participant-related EAD if human factors hinder progress.
- *Information*: Aggregating qualitative and quantitative feedback into executive reports is essential for tracking architectural alignment. Any inaccuracies, inconsistencies, or delays in collecting or interpreting this information contribute to information-related debt.
- *Technologies*: The actions taken within each hotspot likely involve updates or replacements of legacy systems, data platforms, or infrastructure. Misalignment between architectural goals and the actual pace or scope of the technological implementation may result in technology-related debt.
- *Environment*: This initiative operates in a dynamic context of business demands, strategic shifts, and possibly regulatory or market pressures. External factors such as shifting priorities or budget constraints may contribute to deviations from the planned architectural path.
- *Infrastructure*: Shared resources, such as enterprise-wide data repositories, integration platforms, or workflow tools, may be impacted by or impact the architectural changes in each hotspot. Delays in upgrading infrastructure can ripple through the transformation process.
- *Strategy*: This case is fundamentally about alignment with strategic objectives. When planned architectural initiatives fall out of synchronization with execution, strategy-related EAD emerges. The entire initiative aims to identify, make visible, and reduce this type of debt.

Through the structured use of interviews, timelines, and progress reporting, the organization attempts to bridge the gap between "AS-IS" and "TO-BE" states, thereby mitigating alignment

debt. However, this process also highlights how EAD can evolve as misalignments persist, priorities shift, or execution lags behind its intended purpose. Framing this challenge as EAD within the WST framework empowers the organization to manage it not as a vague strategic drift but as a specific, observable, and actionable deviation from architectural intent.

4.3 Third Example: Resilience Constraints Through EAD

The third example illustrates how resilience in a commercial bank can be constrained by accumulated EAD embedded within its internal ICT control systems (cf. Figure 4). Over many years, the bank expanded its digital services, regulatory obligations, and product offerings, but its internal ICT control environment evolved only incrementally. As a result, several core control functions, such as access management, configuration monitoring, asset inventories, and change authorization, operate through fragmented systems and partially manual procedures. This situation does not produce immediate failures, but it systematically reduces resilience by slowing detection, limiting visibility, and making adaptation more cumbersome.

Examined from the WST perspective, the resilience constraints can be mapped to several interdependent forms of EAD:

- *Strategy*: In this context, resilience constraints do not arise from isolated control failures but from the cumulative effect of EAD across the work system. As ICT services, regulatory obligations, and organizational structures become more complex, deferred design decisions become embedded within everyday practices. From a WST perspective, resilience is therefore weakened not by the absence of controls, but by the growing misalignment between how work is prescribed and how it is actually performed. To emphasize, resilience requirements imply that institutions must be able to consistently sustain reporting capabilities during disruptions, while maintaining accuracy, completeness, and timeliness.
- *Processes and activities*: Processes central to Regulation (EU) 2022/2554 of the European Parliament and the Council of 14 December 2022 on digital operational resilience for the financial sector (hereinafter – DORA) compliance, such as ICT incident detection, classification, escalation, and reporting, remain fragmented across organizational units and rely on manual reconciliation steps. Historical workarounds, intended as temporary solutions, have become embedded in operational routines, resulting in inconsistencies in process execution. As a result, the institution experiences delays in meeting DORA's reporting timelines.
- *Participants*: The responsibility for ICT controls is distributed across technology, security, operations, and compliance units. Historical organizational arrangements have led to unclear role boundaries and ownership in several control domains. This EAD relies heavily on a small group of experienced practitioners who understand the legacy landscape and know how to operate effectively within its limitations. Resilience is weakened because critical response and recovery capabilities are tied to individuals rather than being systematically centralized.
- *Information*: DORA requires accurate, up-to-date, and complete information on ICT assets, dependencies, configurations, and incident details. However, the institution maintains multiple, partially synchronized repositories for asset inventories, configuration data, and service dependencies. Inconsistencies between them require extensive manual data consolidation during ICT risk assessments and incident investigations. This EAD compromises situational awareness, delays the identification of impacted services during disruptions, and hinders compliance with DORA's major incident reporting requirements, which rely on timely and accurate information flows.
- *Technologies*: The bank depends on a mixture of modern platforms and long-standing control tools that lack integration capabilities. Some systems do not ensure adequate information availability, for instance, when data from third-party service providers is not consistently integrated or when internal repositories lack the structured information required for maintaining an accurate register of information under DORA.

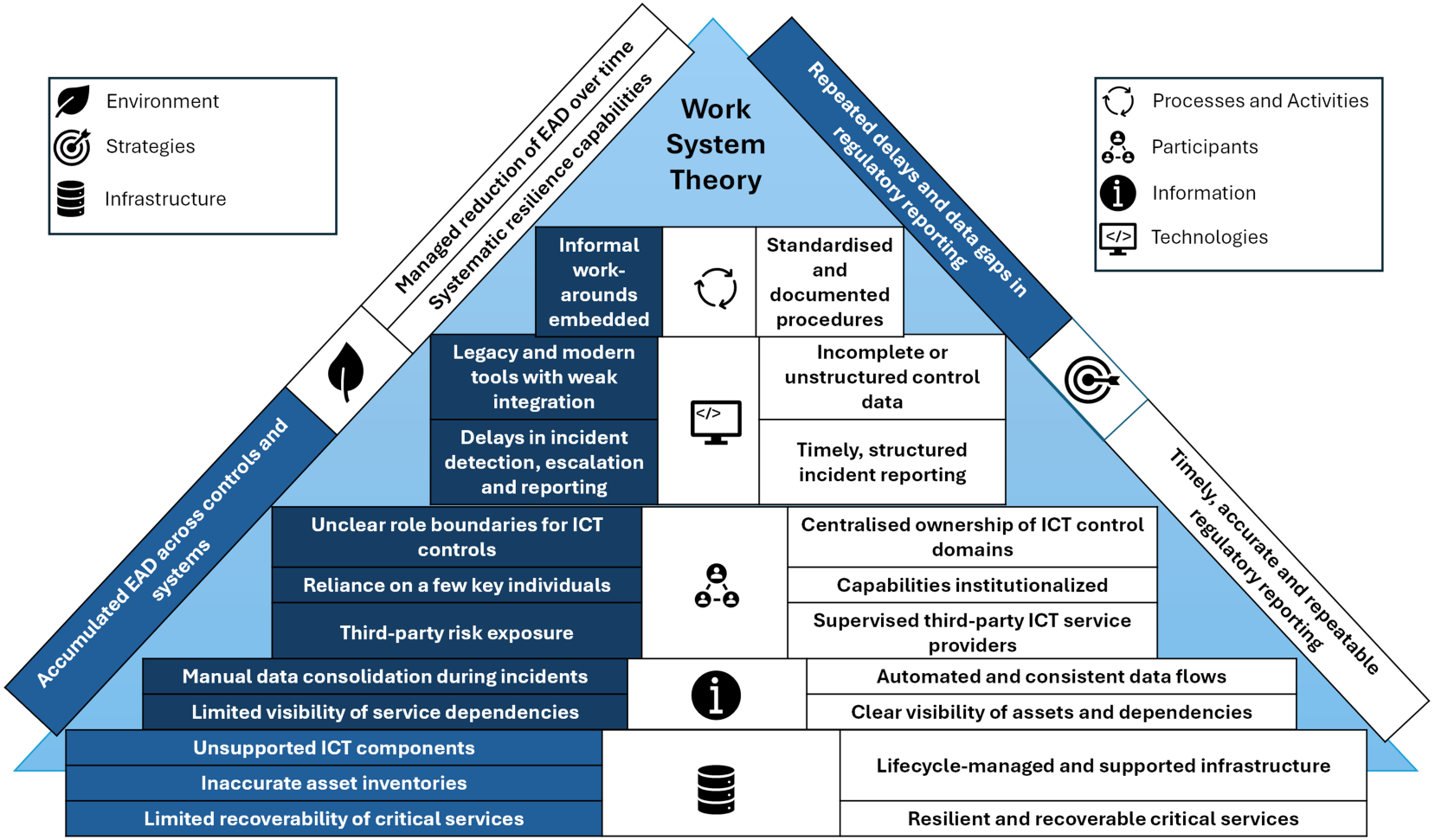

Figure 4. Exemplary Illustration of Resilience Constraints through EAD

- *Infrastructure*: Infrastructure elements rely on ICT asset inventories that still contain outdated or unsupported components, which do not meet DORA's requirements for maintaining up-to-date, lifecycle-managed, and supportable ICT assets. This EAD reduces the bank's ability to comply with DORA's expectations for resilient ICT infrastructure, including accurate asset tracking, controlled use of obsolete technologies, and the ability to ensure continuity and recoverability across critical services.
- *Environment*: The regulatory environment under DORA imposes expectations for operational resilience, ICT governance, risk management processes, and major incident reporting. Because the institution's internal control environment has accumulated debt across processes, technologies, information structures, and infrastructure, it struggles to align with these evolving obligations. Each new regulatory expectation exposes structural weaknesses, such as limited automation, poor data quality, or unclear ownership, thereby amplifying the effect of accumulated EAD on resilience outcomes.

In general, the example demonstrates that EAD accumulated within ICT control functions creates structural misalignments that directly constrain the institution's ability to meet DORA's operational resilience expectations. Rather than isolated deficiencies, the debts span multiple WST elements, collectively reducing the organization's capacity to detect, respond to, and recover from disruptions. This illustrates how resilience deteriorates when the coherence of the work system is compromised over time due to deferred architectural decisions.

4.4 Fourth Example: Highly Complex Legacy System with Outdated Processes

The first modernization case (cf. Figure 5) involves a long-running banking system that relies on tightly coupled COBOL components. Over the years, management focused on introducing new features rather than maintaining the system. This allowed hidden technical, knowledge, and enterprise architecture debt to accumulate over time. As the complexity of the system grew, it resulted in increased difficulty in prioritizing various systems and their growing needs. Among the various problems with the system, notable are the manual processes, obsolete documentation, and the continually decreasing number of available experts. The constantly growing misalignment between the developed systems, their technical state, and the organizational culture necessitates modernization. Modernization is needed not because of the age of technology, but due to the accumulated debt hindering the adoption of the enterprise's needs.

The presented case highlights the EAD in a system that prioritized introducing new features, while neglecting the growing debts. The lack of maintenance and modernization over the years, eroded documentation, and a lack of experts familiar with the system introduces a system that performs current work but is unable to fulfill growing demands. Modernization functions as an EAD repayment method, restoring the architecture structure, documenting relevant knowledge, and realigning the systems with its domain, so that the system can support the growing enterprise requirements.

Using the WST lens, this case reveals debt across multiple components:

- *Environment*: The presented system is subjected to regulatory obligations, which take priority when deciding modernization strategy. Several of the systems are additionally constrained by long-term contracts with external providers. Similar to EAD, these are constraints that are outside the immediate control of the work system, despite influencing the decisions made for it.
- *Strategies*: The organizational strategy of prioritizing new features, rather than focusing on the system's longevity, similar to EAD, may work well in the short term but leads to misalignment in the long run. The system eventually becomes unable to keep up with the requirements. This demonstrates the congruence between EAD and the need for modernization.
- *Infrastructure*: The shared infrastructure services in the enterprise are segmented, which prevents the enterprise from reusing common knowledge, resulting in larger, unnecessary

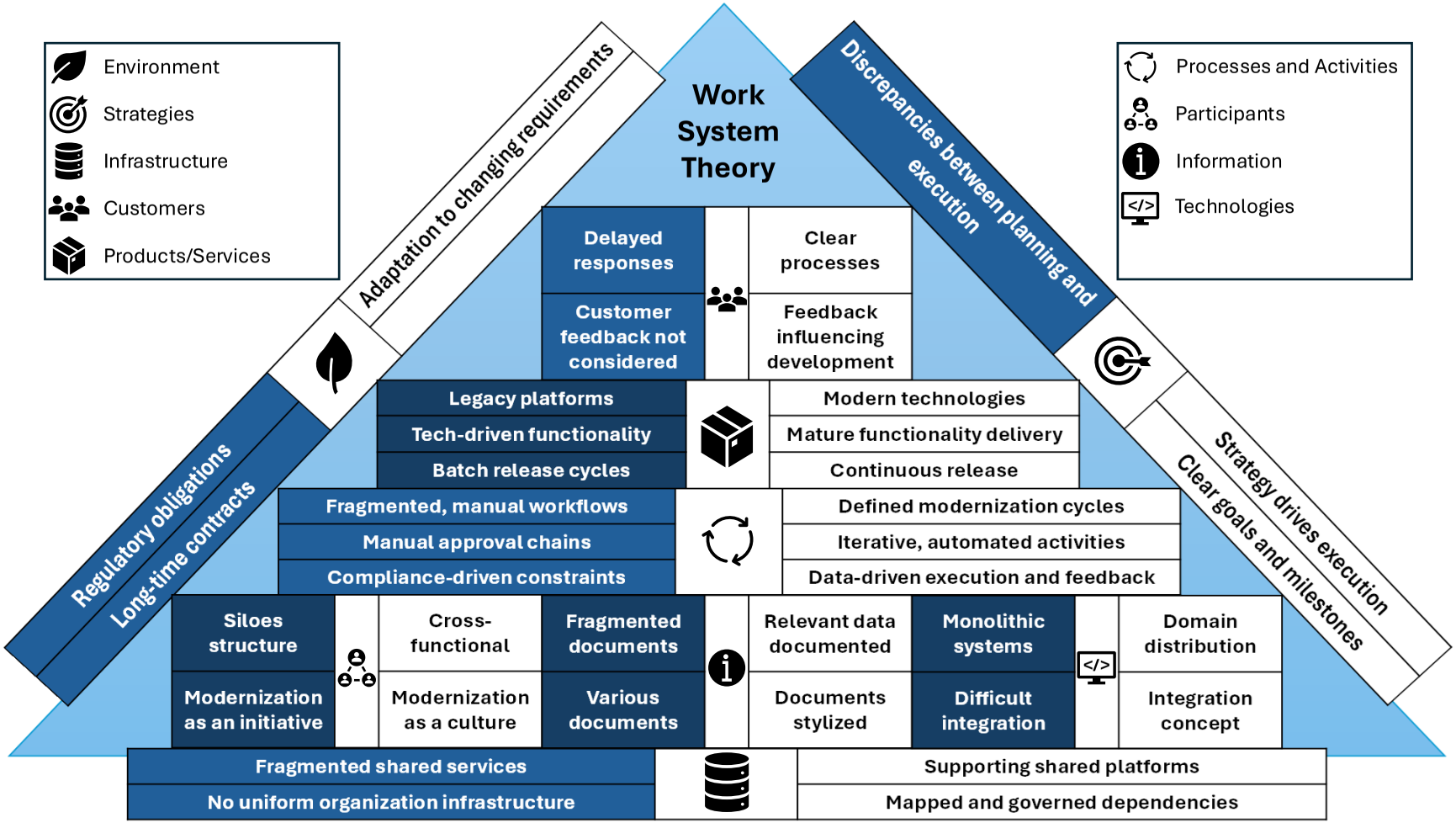

Figure 5. Exemplary illustration of two modernization cases in the WST Framework

maintenance efforts to keep the system operational. This also displays a lack of organization-wide infrastructure, which reflects an absence of an enterprise-level integration strategy, resulting in duplicated functionality, inconsistent workflows, and escalating coordination overhead.

- *Customers*: Due to organizational overhead, the responses to customers are delayed, causing customer dissatisfaction. The disregard of customer feedback often leads to delayed recognition of underlying errors in the system execution, resulting in inconsistent quality across the system.
- *Products/Services*: The provided products/services operate on legacy systems, limiting the scalability and configuration possibilities of the proposed solutions. This makes the functionality dependent on the technological capabilities of the legacy software. The batch-release cycles impose process constraints and additional waiting times, often delaying other, dependent systems and their operation.
- *Processes and Activities*: Similar to the infrastructure services, the enterprise processes are fragmented among the various systems. Every step of the process is compliance-heavy and requires manual approval, as the automatization thereof was blocked by outdated process ownership and regulatory constraints, which led to a process debt.
- *Participants*: The company workers are divided based on legacy hierarchies, mirroring the legacy components. The previous domain division separates business and IT into distinct silos based on their respective responsibilities. Modernization is then treated as an IT initiative, rather than a consideration of holistic enterprise change. This shows a people debt in the enterprise.
- *Information*: The fragmented information on both the technical and domain aspects of the system underlines the limited understanding of the systems. Additionally, the data is often stored in various document types and styles, making it harder to utilize the available information for decision-making.
- *Technologies*: The systems, which grew over time, represent a tightly coupled, monolithic structure that is difficult to analyze and understand. The integration often relies on several customized interfaces, making it more challenging and costly to integrate with newer systems. This further reduces the scalability of the systems.

This case presents how modernization in large, complex organizations requires the consideration of various EADs. Utilizing the WST framework, one can showcase the existing EADs, allowing for better recognition, understanding, and repayment in the future. Notably, EAD accumulation in the presented case is not confined to a single WST element. For instance, the siloed participant along with fragmented processes hinder documentation maintenance, as a result, limiting the ability to assess the technical state. This example demonstrates that modernization must address debts holistically, rather than separately. In this example, replacing the COBOL components without addressing the participant silos and process fragmentation is likely to reproduce these debts in the modernized systems. Overall, the WST provides a structured framework for guiding the modernization and repayment of the EAD within the system.

4.5 Fifth Example: Modern System Utilizing Automated Processes

The second modernization case (cf. Figure 5) revolves around a large enterprise that is already utilizing a cloud-native, microservices-based streaming platform. Over the years, the company has been working with a monolithic system, which has recently been modernized. This allowed hidden EAD to be repaid. The complexity of the system decreased due to the decomposition of the monolith into independently deployable services. Automated deployments, fault-tolerant service design, and continuous delivery pipelines eliminated manual operational steps, enabling rapid and reliable global releases. The focus on independent services enabled the distribution of required knowledge and responsibility among the teams, reducing reliance on the constantly decreasing number of monolith experts. Modernization allowed the enterprise to reach all of its current needs.

The focus of the enterprise is to maintain a modernization culture, ensuring the longevity of the organization.

This case presents an enterprise that recently repaid its accumulated EAD through modernization. However, modernization efforts are not finished in this case. The enterprise needs to preserve architectural modularity, maintain up-to-date documentation of the system, and retain knowledge and communication across the distributed teams. To achieve this, modernization efforts must be regularly applied, including activities such as quality assessment, retirement of obsolete functionality and components, and regular reassessment of modernization needs. This also implies continuous assessment of EAD to identify repayment needs as soon as possible, allowing for small, incremental modernizations rather than large, highly costly ones.

Using the WST lens, this case reveals debt across multiple components:

- *Environment*: The presented system has a clear understanding of its obligations and external factors. The focus needs to be put on the ability to adapt to the constantly changing requirements. This means that changes need to be actively monitored and identified as soon as possible, and that the processes triggered in response must act quickly and clearly.
- *Strategies*: In this case, the goals and milestones of the enterprise are clear and known to all relevant stakeholders. To achieve them, a clear strategy is designed for careful execution. Each part of the execution must be followed to ensure it aligns with the set goals. If the goals change, the enterprise must adapt its strategy and notify all relevant stakeholders of the change.
- *Infrastructure*: In the modernized infrastructure, the enterprise utilizes the reuse of common components, allowing for the sharing of resources and knowledge among existing systems. The identified common functionality, dependencies between systems, and integration layers are thoroughly documented, allowing for fast access if needed.
- *Customers*: The modernization introduces a set of clear processes designed to provide a structured approach for handling customer needs. The feedback gathered from or provided by customers is taken into account during development and assessed with regard to its influence on the system's usability from the user's point of view. This enables us to tailor the system to meet customer needs, resulting in higher customer satisfaction rates.
- *Products/Services*: The products and services are developed using modern technologies, allowing the utilization of the modern standards required. The functionality can be added flexibly thanks to the modernized system, allowing for a more accurate assessment of delivery times. The release cycle is technology independent and relies on the availability of new features.
- *Processes and Activities*: The enterprise processes exhibit a defined structure that guides modernization cycles, designed to respond to modernization needs while minimizing hindrances to modernization. This includes the continuous management of EADs. The activities are iterative and at least partially automated to reduce the load on the development team. Decisions are made based on available data and gathered feedback.
- *Participants*: The teams working on the modernized system are cross-functional, ensuring quick access to domain experts needed to assess the developed functionality. The responsibility for the systems and their architecture is shared. The modernization itself is prioritized and considered an integral part of the enterprise culture. This enables teams to effectively utilize their abilities and minimize waiting times or missing information during development.
- *Information*: During modernization, the system was re-documented, and the documentation is kept up to date to avoid difficulties during future modernization endeavors. The documents share a common style, enabling other teams or experts to easily access relevant data from other parts of the system. This is particularly relevant when considering parts with dependencies.
- *Technologies*: Instead of the previously existing monolith, the system is now operating on components distributed based on domains. This enables a more intuitive understanding of the system and brings it closer to its conceptual model. This also allows for a clearer separation of responsibility and the integration concept.

This case exemplifies how modernization efforts do not stop after EAD repayment. Despite recent modernization, these efforts need to be maintained and integrated into standard processes to ensure that the system remains in an acceptable state, as successful modernization allows for both reactive and preventive EAD management. Here, the cross-functional team structure reduces the occurrence of participant-related debts, whereas automated deployment pipelines limit the process-related ones. It is important for the enterprise to consider modernization as a continuous cultural change rather than a one-time solution. The long-term value of modernization lies not in the technical aspects, but in establishing organizational mechanisms, made explicit through WST, to prevent re-accumulation of EAD. Only such a state allows the system to develop further.

5 Research Gaps and Future Directions

Existing research on identifying EAD covers several elements that align with WST's categories, offering a holistic perspective on how organizations can understand and manage their architectural shortcomings. This research can be categorized into two main areas: works identifying potential symptoms of EAD, referred to as EA Smells [20], [51], [54], and works seeking efficient methods for identifying EAD that inadvertently produce potential EADs as a byproduct [4], [5].

The *people* element is addressed through the roles and perspectives of EA stakeholders. Research [4] highlights the importance of involving stakeholders in identifying and prioritizing EAD through workshops and interviews. This collaboration is essential for bridging the gap between technical and business perspectives, ensuring that EAD is addressed with a shared understanding.

Processes are another focus [51]. This research examines inefficiencies and anti-patterns, such as redundant workflows and poorly integrated systems, which hinder operational performance and innovation. Methods and tools have been developed to identify these process-related issues. These efforts aim to enhance the overall quality of enterprise processes and reduce the long-term impact of inefficiencies.

The *technological* aspect of EAD has received significant attention, particularly in terms of legacy systems and poorly integrated IT infrastructure. Studies [20], [54] have adapted such concepts as software architecture smells to the EA domain, identifying technical flaws that contribute to debt. Tools for detecting these "EA smells" in models have been proposed, providing a systematic way to assess and quantify the quality of an enterprise's technological architecture.

Information quality and flow within the EA are recognized as important contributors to debt. Issues such as incomplete documentation, outdated data, and misaligned information repositories often exacerbate architectural challenges. Research [5] highlights the need for aggregated reporting mechanisms to track progress in aligning the actual EA state with the desired target state, emphasizing the role of reliable information in decision-making.

External constraints, such as legal, regulatory, and organizational factors, form another dimension of EAD. Studies [4] have shown how such factors as bargaining agreements or compliance requirements can limit an organization's flexibility to make necessary architectural changes. These constraints underscore the need for careful planning to navigate the trade-offs between operational needs and external obligations.

At the heart of EAD research is the alignment of purpose, which reflects the deviation between an organization's current EA state and its ideal or target state. This misalignment is seen as the core definition of EAD [1]. Frameworks and metrics are being developed to measure and address this misalignment, with catalogs of EADs and smells providing tools to help organizations manage these gaps and ensure that their architecture remains aligned with strategic objectives.

While significant progress has been made in understanding EAD, several gaps remain in its exploration through the lens of WST. For the people dimension, research often highlights stakeholder involvement in workshops and interviews. Yet, little attention is given to how differing

roles, organizational cultures, and stakeholder incentives influence the prioritization and resolution of EAD. Similarly, while anti-patterns and inefficiencies have been identified in the processes category, there is a lack of focus on the dynamic evolution of processes and their relationship with architectural changes over time.

Other dimensions of WST are similarly underexplored. In the information domain, gaps exist in understanding how robust data governance and integration practices can help manage information-related EAD. Regarding external constraints, while regulatory and legal obligations are recognized as contributors to EAD, there is limited research on frameworks for balancing these constraints with architectural flexibility.

A further limitation of our analysis is that we largely treated the WST elements as separate "boxes" and did not systematically examine their connections. In practice, many debts are fundamentally relational and accumulate between the different aspects of WST. Future research should therefore complement element-wise categorization with explicit modeling of interdependencies to understand how debt spreads, how repayment in one area can shift debt elsewhere, and which additional EADs emerge specifically in the connectors and coordination mechanisms between the "boxes".

An additional, largely unexplored direction is to transfer WST's drill-down principle to EAD by treating debt as potentially compositional across levels of decomposition: debt may originate locally (e.g., within a subsystem's technology or process design) but "pile up" when subsystems are composed into larger end-to-end capabilities, especially through shared interfaces, data flows, and coordination mechanisms. This suggests the need for multi-level EAD traceability, where debts are recorded not only at a single EA layer or artifact but also linked across parent–child decompositions and across integration seams, allowing researchers and practitioners to study how debt propagates upward, how repayment at one level shifts or reveals debt at another, and how prioritization should account for cross-level ripple effects rather than isolated symptoms.

Finally, in terms of purpose alignment, the lack of standardized metrics to define and evaluate the "ideal state" of an EA poses a challenge, as does understanding how shifts in organizational strategy impact this alignment. Future research should address these gaps by developing adaptive frameworks, leveraging emerging technologies, and exploring stakeholder engagement strategies. Doing so will enable organizations to manage EAD more effectively while aligning with dynamic business needs and external pressures.

5.1 Resilience

Future work should also investigate how different forms of EAD across work system elements affect an organization's capacity to anticipate, absorb, recover from, and adapt to disruptions. This includes examining how process fragmentation, information inconsistencies, technology legacy status, and role dependencies weaken resilience, particularly under regulatory requirements that are focused on digital resilience. Such work would position resilience not as a non-functional quality attribute, but as a dynamic outcome of architectural governance and debt management practices over time.

5.2 Modernization

In the context of modernization, future work should focus on the operationalization of modernization with explicit consideration of EAD. This includes defining a concrete, EAD–aware modernization process spanning all enterprise architecture phases, as well as developing measurable indicators to support debt-informed decision-making. Currently, modernization decisions are largely driven by expert judgment and assessments of the current system's state but lack systematic support for evaluating trade-offs between missing modernization actions, short-term modernization gains, and the long-term longevity and sustainability of software

systems. Particular attention should be paid to people-related and documentation-related EAD, as these remain especially challenging in modernization contexts despite their strong influence on modernization outcomes. Finally, longitudinal, industry-oriented studies and dedicated tool support are needed to strengthen the practical applicability of the proposed approach.

6 Conclusion

Within this work, we extended the original investigation of EAD through the lens of WST [8] by integrating a broader theoretical analysis and introducing two new areas of concern: modernization and resilience. While EAD has traditionally been understood as an extension of technical debt, its broader implications across processes, technologies, information, and stakeholder engagement require a structured theoretical foundation. Our original study applied WST to categorize and analyze manifestations of EAD; this extended version expands the scope by systematically comparing thirteen theories from IS literature against a set of analytical categories relevant to EAD.

By categorizing EAD using both WST and alternative theoretical lenses, we provide a more comprehensive basis for identifying, assessing, and addressing architectural misalignments. This comparison not only reaffirms the strengths of WST in capturing stakeholder roles and systemic change but also reveals conceptual blind spots and potential synergies across theories. The structured assessment enhances our ability to make informed decisions about managing EA and positions WST within a broader theoretical landscape.

This article further introduces modernization and resilience as critical concerns affected by EAD. Modernization, the ability to renew or evolve the enterprise architecture, and resilience, the capacity to withstand and adapt to disruptions, both depend on well-aligned people, processes, technologies, and information. As EA misalignments accumulate, they impair modernization initiatives and weaken organizational resilience. Our analysis shows that EAD constrains an organization's responsiveness to change, its capacity to comply with evolving regulations, and its ability to recover from operational disruptions.

Key takeaways from this research include the importance of systematically identifying EAD to improve architectural decision-making, the need for tools and methodologies to monitor and mitigate its effects, and the necessity of integrating stakeholder perspectives to align EA with strategic goals. Future research should focus on developing adaptive frameworks, exploring hybrid theoretical models, and leveraging emerging technologies to refine the quantification and management of EAD. By advancing our understanding of EAD through a structured theoretical foundation, organizations can adopt a more proactive, sustainable, and agile approach to enterprise architecture. Additionally, future work should investigate how tool-supported work system modeling methods can operationalize multi-level EAD traceability through explicit decomposition and drill-down across interconnected work systems.

References

[1] S. Hacks, H. Höfert, J. Salentin, Y. C. Yeong, and H. Lichter, "Towards the definition of enterprise architecture debts," in *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2019, pp. 9–16. Available: https://doi.org/10.1109/EDOCW.2019.00016

[2] S. Gregor, "The nature of theory in information systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, 2006. Available: https://doi.org/10.2307/25148742

[3] S. Alter, "Work system theory: Overview of core concepts, extensions, and challenges for the future," *Journal of the Association for Information Systems*, vol. 14, no. 2, pp. 72–121, 2013. Available: https://doi.org/10.17705/1jais.00323

[4] J. Jung, S. Hacks, T. de Gooijer, M. Kinnunen, and K. Rehring, "Revealing common enterprise architecture debts: Conceptualization and critical reflection on a workshop format industry experience report," in *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2021, pp. 271–278. Available: https://doi.org/10.1109/EDOCW52865.2021.00058

[5] S. Daoudi, M. Larsson, S. Hacks, and J. Jung, “Discovering and assessing enterprise architecture debts,” *Complex Systems Informatics and Modeling Quarterly (CSIMQ)*, no. 35, pp. 1–29, 2023. Available: https://doi.org/10.7250/csimq.2023-35.01

[6] P. Alexander, S. Hacks, J. Jung, U. Steffens, Ö. Uludağ, and H. Lichter, “A Framework for Managing Enterprise Architecture Debts – Outline and Research Directions,” in *10th International Workshop on Enterprise Modeling and Information Systems Architectures*, vol. 2628, 2020, pp. 5–10. Available: https://ceur-ws.org/Vol-2628/paper1.pdf

[7] S. Hacks, M. Brosius, and S. Aier, “A case study of stakeholder concerns on EAM,” in *2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW)*, 2017, pp. 50–56. Available: https://doi.org/10.1109/EDOCW.2017.17

[8] S. Hacks and A. Slupczynski, “Advancing enterprise architecture debt: Insights from work system theory,” in *Perspectives in Business Informatics Research, Lecture Notes in Business Information Processing*, vol. 562, 2025, pp. 107–123. Available: https://doi.org/10.1007/978-3-032-04375-7_7

[9] Ö. Uludağ, M. Kleehaus, X. Xu, and F. Matthes, “Investigating the role of architects in scaling agile frameworks,” in *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, 2017, pp. 123–132. Available: https://doi.org/10.1109/EDOC.2017.25

[10] Ö. Uludağ, M. Kleehaus, N. Reiter, and F. Matthes, “What to Expect from Enterprise Architects in Large-Scale Agile Development? A Multiple-Case Study,” in *Proceedings of the 25th Americas Conference on Information Systems (AMCIS 2019)*, 2019. Available: https://aisel.aisnet.org/amcis2019/org_transformation_is/org_transformation_is/23/

[11] F. Gampfer, A. Jürgens, M. Müller, and R. Buchkremer, “Past, current and future trends in enterprise architecture – a view beyond the horizon,” *Computers in Industry*, vol. 100, pp. 70–84, 2018. Available: https://doi.org/10.1016/j.compind.2018.03.006

[12] W. Cunningham, “The WyCash portfolio management system,” *ACM SIGPLAN OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1993. Available: https://doi.org/10.1145/157710.157715

[13] Z. Li, P. Avgeriou, and P. Liang, “A systematic mapping study on technical debt and its management,” *Journal of Systems and Software*, vol. 101, pp. 193–220, 2015. Available: https://doi.org/10.1016/j.jss.2014.12.027

[14] P. Kruchten, R. L. Nord, and I. Ozkaya, “Technical Debt: From Metaphor to Theory and Practice,” *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012. Available: https://doi.org/10.1109/MS.2012.167

[15] C. Seaman and Y. Guo, “Measuring and monitoring technical debt,” *Advances in Computers*, vol. 82, pp. 25–46, 2011. Available: https://doi.org/10.1016/B978-0-12-385512-1.00002-5

[16] J. S. Addicks and H.-J. Appelrath, “A method for application evaluations in context of enterprise architecture,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 131–136. Available: https://doi.org/10.1145/1774088.1774115

[17] B. Curtis, J. Sappidi, and A. Szynkarski, “Estimating the Principal of an Application’s Technical Debt,” *IEEE Software*, vol. 29, no. 6, pp. 34–42, 2012. Available: https://doi.org/10.1109/MS.2012.156

[18] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, “In search of a metric for managing architectural technical debt,” in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, 2012, pp. 91–100. Available: https://doi.org/10.1109/WICSA-ECSA.212.17

[19] A. Slupczynski and S. Hacks, “Towards a knowledge base of terms on enterprise architecture debt,” in *Enterprise Design, Operations, and Computing. EDOC 2023 Workshops, Lecture Notes in Business Information Processing*, 2024, vol. 498, pp. 194–210. Available: https://doi.org/10.1007/978-3-031-54712-6_12

[20] J. Salentin and S. Hacks, “Towards a catalog of enterprise architecture smells,” in *Wirtschaftsinformatik*, 2020, pp. 276–290.

[21] M. Smajevic, S. Hacks, and D. Bork, “Using Knowledge Graphs to Detect Enterprise Architecture Smells,” in *The Practice of Enterprise Modeling. PoEM 2021. Lecture Notes in Business Information Processing*, 2021, vol. 432, pp. 48–63. Available: https://doi.org/10.1007/978-3-030-91279-6_4

[22] Y. C. Yeong, S. Hacks, and H. Lichter, “Prioritization of EA debts facilitating portfolio theory,” in *Proceedings of the 7th International Workshop on Quantitative Approaches to Software Quality co-located with 26th Asia-Pacific Software Engineering Conference (APSEC 2019)*, vol. 2511, 2019, pp. 45–52. Available: https://ceur-ws.org/Vol-2511/QuASoQ-06.pdf

[23] L. Liss, H. Kämmerling, P. Alexander, and H. Lichter, “Towards a Catalog of Refactoring Solutions for Enterprise Architecture Smells,” in *Joint Proceedings of SEED 2021 & QuASoQ 2021 co-located with 28th Asia Pacific Software Engineering Conference 2021*, vol. 3062, 2021, pp. 60–69. Available: https://ceur-ws.org/Vol-3062/Paper09_QuASoQ.pdf

[24] A. Slupczynski, P. Alexander, and H. Lichter, “A process for evaluating the prudence of enterprise architecture debts,” in *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) – Volume 2: ICEIS*, 2023, pp. 623–630. Available: https://doi.org/10.5220/0011971400003467

[25] S. Hacks and J. Jung, “A first validation of the enterprise architecture debts concept,” in *Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing*, 2023, vol. 479, pp. 217–226. Available: https://doi.org/10.1007/978-3-031-34241-7_15

[26] A. Chis, A.-M. Ghiran, and S. Alter, “Informing enterprise knowledge graphs with a work system perspective,” *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 19, 2024. Available: https://doi.org/10.18417/emisa.19.7

[27] H. Florez, M. Sánchez, and J. Villalobos, “A catalog of automated analysis methods for enterprise models,” *SpringerPlus*, vol. 5, no. 1, article 406, 2016. Available: https://doi.org/10.1186/s40064-016-2032-9

[28] H. Florez, M. Sánchez, and J. Villalobos, “Extensible model-based approach for supporting automatic enterprise analysis,” in *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*, 2014, pp. 32–41. Available: https://doi.org/10.1109/EDOC.2014.15

[29] B. Krauze and J. Grabis, “A conceptual model of digital immune system to increase the resilience of technology ecosystems,” in *Research Challenges in Information Science. RCIS 2024. Lecture Notes in Business Information Processing*, vol. 513, 2024, pp. 82–96. Available: https://doi.org/10.1007/978-3-031-59465-6_6

[30] B. Krauze, “An analysis of resilience in digital business ecosystems,” in *Research Challenges in Information Science. RCIS 2025. Lecture Notes in Business Information Processing*, vol. 548, 2025, pp. 162–171. Available: https://doi.org/10.1007/978-3-031-92471-2_13

[31] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, “How do professionals perceive legacy systems and software modernization?” in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014, 2014, pp. 36–47. Available: https://doi.org/10.1145/2568225.2568318

[32] M. Chaima, A. Sofiane, and B. Assia, “Modernizing Legacy IT Systems : Methods, Challenges, and Strategic Insights,” in *2024 1st International Conference on Innovative and Intelligent Information Technologies (IC3IT)*, 2024, pp. 1–6. Available: https://doi.org/10.1109/IC3IT63743.2024.10869403

[33] M. L. Brodie and M. Stonebraker, “Darwin: On the incremental migration of legacy information systems,” *GTE Laboratories Incorporated*, vol. TR-0222-10-92-165, 1993. Available: https://api.semanticscholar.org/CorpusID:7953055

[34] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W.-G. Tan, “Types of software evolution and software maintenance,” *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 13, no. 1, pp. 3–30, 2001. Available: https://doi.org/10.1002/smr.220

[35] L. Wessel, A. Baiyere, R. Ologeanu-Taddei, J. Cha, and T. Blegind Jensen, “Unpacking the Difference Between Digital Transformation and IT-Enabled Organizational Transformation,” *Journal of the Association for Information Systems*, vol. 22, no. 1, pp. 102–129, 2021. Available: https://doi.org/10.17705/1jais.00655

[36] W. M. Ulrich and P. H. Newcomb, *Information Systems Transformation: Architecture-Driven Modernization Case Studies*, 1st ed. Morgan Kaufmann, 2010. Available: https://doi.org/10.1016/C2009-0-19987-7

[37] R. Winter and R. Fischer, “Essential layers, artifacts, and dependencies of enterprise architecture,” in *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW’06)*, 2006, pp. 30–30. Available: https://doi.org/10.1109/EDOCW.2006.33

[38] W. F. Boh and D. Yellin, “Using enterprise architecture standards in managing information technology,” *Journal of Management Information Systems*, vol. 23, no. 3, pp. 163–207, 2006. Available: https://doi.org/10.2753/MIS0742-1222230307

[39] S. Hacks and H. Lichter, “A probabilistic enterprise architecture model evolution,” in *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, 2018, pp. 51–57. Available: https://doi.org/10.1109/EDOC.2018.00017

[40] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, “The financial aspect of managing technical debt: A systematic literature review,” *Information and Software Technology*, vol. 64, pp. 52–73, 2015. Available: https://doi.org/10.1016/j.infsof.2015.04.001

[41] Á. Mursu, I. Luukkonen, M. Toivanen, and M. Korpela, “Activity Theory in Information Systems Research and Practice: Theoretical Underpinnings for an Information Systems Development Model.” *Information Research*, vol. 12, no. 3, p. 3, 2007.

[42] A. Tatnall, “Actor-network theory as a socio-technical approach to information systems research,” in *Socio-Technical and Human Cognition Elements of Information Systems*, 2003, pp. 266–283. Available: https://doi.org/10.4018/978-1-59140-104-9.ch013

[43] O. Volkoff and D. M. Strong, “Affordance theory and how to use it in IS research,” in *The Routledge Companion to Management Information Systems*. Routledge, 2017, pp. 232–245.

[44] N. McBride, “Chaos theory as a model for interpreting information systems in organizations,” *Information Systems Journal*, vol. 15, no. 3, pp. 233–254, 2005. Available: https://doi.org/10.1111/j.1365-2575.2005.00192.x

[45] J. Reinking, “Contingency theory in information systems research,” *Information Systems Theory. Integrated Series in Information Systems*, vol. 28, pp. 247–263, 2012. Available: https://doi.org/10.1007/978-1-4419-6108-2_13

[46] R. Matavire and I. Brown, “Investigating the use of ”grounded theory” in information systems research,” in *SAICSIT ’08: Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, 2008, pp. 139–147. Available: https://doi.org/10.1145/1456659.1456676

[47] W. Currie, “Contextualising the IT artefact: towards a wider research agenda for IS using institutional theory,” *Information Technology & People*, vol. 22, no. 1, pp. 63–77, 2009. Available: https://doi.org/10.1108/09593840910937508

[48] G. Vial and S. Rivard, “Conceptualizing information systems development as an organizational routine: Implications and avenues for research,” *SIGMIS Database*, vol. 53, no. 3, pp. 91–107, 2022. Available: https://doi.org/10.1145/3551783.3551790

[49] S. C. Palvia, R. S. Sharma, and D. W. Conrath, “A socio-technical framework for quality assessment of computer information systems,” *Industrial Management & Data Systems*, vol. 101, no. 5, pp. 237–251, 2001. Available: https://doi.org/10.1108/02635570110394635

[50] M. C. Winter, D. H. Brown, and P. B. Checkland, “A role for soft systems methodology in information systems development,” *European Journal of Information Systems*, vol. 4, no. 3, pp. 130–142, 1995. Available: https://doi.org/10.1057/ejis.1995.17

[51] B. Lehmann, P. Alexander, H. Lichter, and S. Hacks, “Towards the identification of process anti-patterns in enterprise architecture models,” in *8th QUASOQ*, vol. 2767, 2020, pp. 47–54. Available: https://ceur-ws.org/Vol-2767/06-QuASoQ-2020.pdf

[52] F. M. Assef, R. Andersen, A.-L. Andersen, T. D. Brunoe, and K. Nielsen, “Capability-based classification of resilience KPIs and research directions,” in *Advances in Production Management Systems. Cyber-Physical-Human Production Systems: Human-AI Collaboration and Beyond. APMS 2025. IFIP Advances in Information and Communication Technology*, vol. 767, 2025, pp. 50–70. Available: https://doi.org/10.1007/978-3-032-03542-4_4

[53] A. Slupczynski and H. Lichter, “Challenges and impact factors on modernization projects - results of an industry study,” *Softwaretechnik-Trends*, vol. 45, no. 2, pp. 12–13, 2025.

[54] B. Tieu and S. Hacks, “Determining enterprise architecture smells from software architecture smells,” in *2021 IEEE 23rd Conference on Business Informatics (CBI)*, 2021, pp. 134–142. Available: https://doi.org/10.1109/CBI52690.2021.10064