Advancing Enterprise Architecture Debt: Insights from Work System Theory

Simon Hacks[1][0000−0003−0478−9347] and Ada Slupczynski[2][0000−0001−5403−7720]

[1] Stockholm University, Stockholm, Sweden
simon.hacks@dsv.su.se

[2] RWTH Aachen University, Aachen, Germany
ada.slupczynski@swc.rwth-aachen.de

Abstract. Enterprise Architecture (EA) debt emerges when short-term decisions lead to structural inefficiencies that hinder organizational agility and strategic alignment. This paper applies Work System Theory (WST) to categorize and analyze EA debt, offering a structured approach to identifying and managing it. We highlight key challenges, research gaps, and future directions by mapping EA debt to WST components. The findings emphasize the need for adaptive frameworks, improved stakeholder engagement, and systematic debt management strategies.

Keywords: Enterprise Architecture Debt · Foundational Theory · Work System Theory.

1 Introduction

Enterprise Architecture (EA) debt has been defined as *"the deviation of the currently present state of an enterprise from a hypothetical ideal state."* [14] It arises when decisions in EA are delayed, or shortcuts are taken, leading to problems over time. These problems can reduce an organization's ability to adapt, perform well, or achieve its goals. Despite its importance, EA debt is not founded in any overarching theory but adapted from the concept of technical debt [14], making it hard to study or manage effectively. A strong theoretical foundation can help by identifying EA debt, making it easier to measure and analyze. It also ensures that decisions about managing EA debt are based on evidence and not just intuition.

A theoretical foundation also improves how we study EA debt [12]. It allows researchers to look at its causes and effects in a structured way and identify patterns across different organizations. This makes it possible to develop better strategies for managing EA debt and to learn from the experiences of others. Additionally, having a theory allows EA debt to be connected to other fields, like IT management and organizational strategy, for a more complete understanding.

Work System Theory (WST) [3] is an excellent fit for studying EA debt because it looks at an organization's relationships between people, processes, technologies, and information. EA debt often arises when these elements are not aligned [17, 9], 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. EA debt is not static; it grows or evolves as organizations adapt to new challenges [2]. WST's focus on change makes it well-suited to study the dynamic nature of EA debt. Finally, WST emphasizes the role of stakeholders—people and groups affected by or involved in a system. EA debt often involves trade-offs between stakeholders, such as balancing short-term cost savings against long-term system performance [13]. WST provides a way to examine these trade-offs and their impact on the organization.

This paper explores how WST can provide a theoretical foundation for understanding EA debt. Using WST, we can better define, analyze, and manage EA debt, helping organizations improve their EA. The rest of this work is structured as follows: Section 2 provides an overview of EA debt and WST. Section 4 applies WST to real-world examples, illustrating how EA debt manifests across different system components. Section 5 identifies key research gaps in the existing literature, followed by the related work. Finally, Section 6 concludes the paper.

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 [37]. Simultaneously, there remains a scarcity of approaches that effectively support long-term architectural planning [38, 11].

In response to these challenges, Hacks et al. [14] 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[6, 20], to the enterprise level. While Technical Debt has proven valuable in identifying software deficits, guiding decision-making, and raising awareness [18, 30], its focus remains mainly on isolated systems. This narrow scope often neglects the broader architectural concerns that span entire organizations [1, 8, 25].

EAD is "the deviation of the currently present state of an enterprise from a hypothetical ideal state" [14]. 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 [32] have developed a domain ontology that captures and structures key concepts in EAD.

Research on EAD has evolved along two principal streams [2]: (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 [29] introduced the concept of EA Smells, a set of indicators for architectural inefficiencies. These were operationalized through a prototype capable of identifying such smells in ArchiMate models. Smajevic et al. [33] further advanced this work by developing an automated tool to support EA Smell detection.

Regarding managing EADs, Yeong et al. [45] 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. [21] introduced refactoring strategies to eliminate identified debts. Complementing these approaches, Slupczynski et al. [31] 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. [2] proposed a management framework for EAD that includes identifying, collecting, assessing, prioritizing, and resolving debts. Jung et al. [17] 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. [9] for improved time efficiency and impact assessment.

To empirically assess the utility of the EAD concept, Hacks and Jung [15] conducted a controlled experiment with students tasked with modeling 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.

2.2 A Theory for Enterprise Architecture Debt

In his work on work systems theory, Alter [3] refers to other IS theories, such as general systems theory, socio-technical theory, actor network theory, organizational routines, soft systems methodology, and activity theory. Alter also argues that one can view UML as a theory. In the context of IS, other theories, not considered by Alter, have been studied, including grounded theory, institutional theory, affordance theory, contingency theory, and chaos theory. To identify the theory most suitable for the representation of EA Debts, the presented theories have been analyzed based on their characteristics, such as the consideration of architectural elements, support for debt management, and consideration of decision making, including the long-term consequences.

Least applicable theories As presented by McBride [24], *chaos theory* can be used to observe change and understand organizational behavior. It is applicable for complex systems with high unpredictability, which might make it interesting for EA consideration. However, it may struggle with the more measurable and manageable aspects of EA Debts. EA Debts can be systematic, traceable, and result from technical or enterprise decisions, but chaos theory might fail to capture such debts.

An increase of interest can be observed in using *affordance theory* for IS [41]. It focuses on the improvement of the perceived affordances of systems, making it a good choice for user involvement and UX consideration. However, it does not represent the strategy, technical dependencies, and system complexity related to EA Debts.

Another theory is the *contingency theory*, Reinking [27] presents the primary constructs of the theory applied in the context of IS. Contingency Theory may support decision-making by considering the context of the enterprise, but it does not have a formal way of representing EA artifacts, such as systems, components, and their dependencies. It is unsuitable to represent the architectural decision and its impact on the EA Debt.

Tatnall [35] advocates for the usage of *actor network theory*, especially in the context of the implementation of IS. Like Tatnall [34], Iyamu and Sekgweleo [16] focus on the applicability of actor network theory in the context of the social aspects of IS. Despite its usability for IS, it lacks the clear structures much needed in the EA Debt consideration. It typically does not consider the hierarchical structure, making it difficult to model the debt across various EA layers. Especially when considering the debt consequences and their propagation.

Vial and Rivard [40] argue that using *organizational routines* may improve the shared understanding of the stakeholders involved in IS Development. Organizational Routines focus on the representation of stakeholders and processes, which are recurring in nature. Organizational Routines do not model the decisions made, failing to consider debt propagation or long-term consequences. EA Debt not only focuses on the processes but also includes the consideration of technical, financial, and organizational aspects.

Moderately applicable theories Currie [7] advocates using *institutional theory* in the context of change management in IS. It can represent the organizational context, including the consideration of how the organization shapes the decisions, allowing for the analysis of the dependency of the accumulated debts and the reasons behind them. But it does not model the technical facets of EA Debts, such as component dependencies or architecture, well. Those, however, are relevant for understanding and managing EA Debts.

Winter et al. [43] present the application of *soft systems methodology* for understanding the organization behind the IS, thanks to its explicit, well-established ways of modeling organizational activity. It might emphasize the stakeholder perspective on EA and underline conflicting stakeholder views. However, it might prove inefficient when considering more architectural components, data flow, or technical dependencies of EA Debts. EA Debts need the representation of the impact that short-term decisions have on the long-term existence of the enterprise.

Mursu et al. use the *activity theory* to propose a model to help bridge the gap between the present and goal states. Activity theory focuses on activity and context, which might help identify some of the root causes, but may prove to be ineffective with the representation of the technical aspects, like architecture or infrastructure. This may hinder the monitoring of the technical decisions long-term influence.

Grounded theory has been widely used in the context of IS. Matavire and Brown [23] analyze its use and application, focusing on four approaches used in the IS context. Wiesche et al. [42] also studied how grounded theory is applied in the context of IS. Verdecchia et al. [39] applied grounded theory to architectural technical debts (ATD) to present the factors most relevant to stakeholders working with ATD. It can help represent the consequences and patterns of the EA Debts, but may be difficult to apply directly to EA artifacts due to its lack of representation of the architecture or risk related to technical decisions. Its strength lies in interpretation, rather than representation.

UML as a theory is well suited for the representation of the AS-IS state of the system, allowing for the documentation of the EA structures. Yet, it might be difficult to use to represent the consequences or the changes in EA Debt over time, such as growing interest, decreasing system quality, or rationale for accepting certain debts. EA Debt is dynamic in its nature, which is not represented by the static UML.

Most applicable theories Both Alter and Gregor [12] argue that *general systems theory* allows considering systems of interest on a high enough abstraction level to be applied to various systems. It considers input, throughput, output, feedback, boundary, and environment, but may fail to represent services, data flows, or architecture layers' interaction. Although quite general and applicable in many scenarios, it may have difficulties modeling EA artifacts or debts. Especially considering the management of EA Debts, it may fail to capture the technical granularity or to analyze the propagation of the debt through various architecture layers.

Luna-Reyes et al. [22] use *socio-technical theory* to represent the dualism in IS, focusing on social and technical challenges. Palvia et al. [26] propose a framework based on socio-technical theory to evaluate the quality of the newly implemented information system (IS). Rinta et al. [28] used socio-technical analysis to propose a two-level system-dynamics model to help predict the challenges of modernizing legacy systems with high technical debt and suggest suitable strategies. Socio-technical theory focuses on the interaction between stakeholders and systems, possibly helping to analyze how stakeholder involvement influences the debt. However, it may have problems considering the architecture and application layer due to the lack of a formal structure to represent software components, interfaces, and the various dependencies between them. Here, debt propagation is also difficult to model.

Work Systems Theory is well suited for EA Debts consideration as it provides a well-balanced model, focusing on the representation of processes, people, technologies, information structures, and their interaction. Its biggest flaw is that it might oversimplify the EA landscape by the use of the work system metaphor, oversimplifying the logic or glossing over the dependencies, which might lead to incomplete information.

Summary From the theories, Work Systems Theory, General Systems Theory, and Socio-Technical Theory seem to show the most promising fit. WST is characterized as a structured and flexible framework, providing concepts relevant to EA Debt. While it would benefit from certain adaptation, it seems to require the least among the candidates. The other theories tend to focus on singular aspects of EA Debts, failing to capture EA Debt in its entire complexity. 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 EA Debts.

2.3 Work System Theory

Work System Theory was proposed by Steven Alter as a means to bridge the gap between business and IT stakeholders working on IS by proposing a method to describe the systems without the often complex IT concepts. Defined based on Gregor's [12] 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 (WSM).*

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 the systems in organizations using Work Systems as a base unit. A Work System is a socio-technological system where humans and machines perform work to provide specific products/services to their customers. To accomplish that, they use information, technology, and other resources.

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), not the ideal to-be state of the work system. (2) **Participants** perform the 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 using them. (2) **Customers** receive the products/services and use them for purpose different than work activities inside the work system itself.

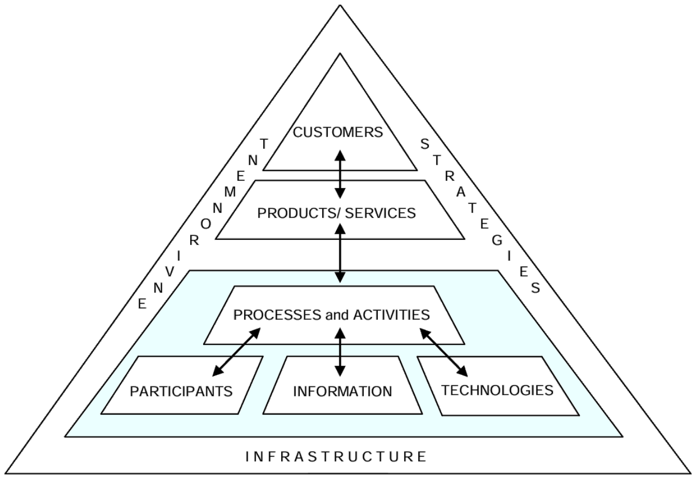

Fig. 1. The work system theory framework as proposed by Alter [3]

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.*

3 Mapping Enterprise Architecture Debt to Work System Theory

EA debt is the accumulation of compromises, deferred decisions, or shortcuts 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." [14] EA debt arises when short-term fixes or decisions to address immediate challenges lead to structural weaknesses or architectural misalignments, creating long-term costs and reducing organizational agility.

EA debt 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 [17, 9]. In the following, we illustrate the connection between EAD concepts and WST:

Participants and Customers: EA debt often originates from decisions prioritizing short-term productivity or immediate usability for specific teams [17]. Over time, these decisions can create gaps in skills, collaboration, and communication. For example, 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 [19]. 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 EA debt. 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 [4].

Information: Poorly designed IS or inconsistent data standards can contribute significantly to EA debt. Issues such as duplicate or incomplete data repositories, lack of integration between systems, or misaligned data structures [17, 9] can reduce the quality and reliability of information. These issues often lead to errors, inefficiencies, and lost opportunities for leveraging data-driven decision-making.

Evolution: EA debt is not static; it evolves as the organization and its architecture respond to internal and external pressures. New business demands, emerging technologies, and shifting market conditions often require organizations to adjust their architecture rapidly [10]. While these adjustments may address immediate needs, they frequently create new forms of debt that must be managed in the future [2]. Understanding how EA debt accumulates, changes, and impacts the organization over time is essential for developing effective management strategies.

Alignment: EA primarily aims to align IT systems, processes, and capabilities with the organization's strategic objectives [5]. However, EA debt often undermines this alignment, reducing the architecture's ability to support the business effectively. Misaligned priorities among stakeholders [13], such as balancing cost savings, speed to market, and long-term architectural integrity, can create trade-offs that result in architectural compromises. Addressing EA debt requires careful consideration of these trade-offs to ensure the architecture continues to deliver value to the organization.

Interdependencies: EA is inherently interconnected [44], with changes in one area often affecting others. For example, implementing new technology may disrupt established workflows, while shifts in business strategy may render specific architectural components obsolete. The systemic nature of EA debt means that small decisions or compromises can have far-reaching consequences. A comprehensive understanding of these interdependencies is crucial for identifying the root causes of EA debt and mitigating its effects.

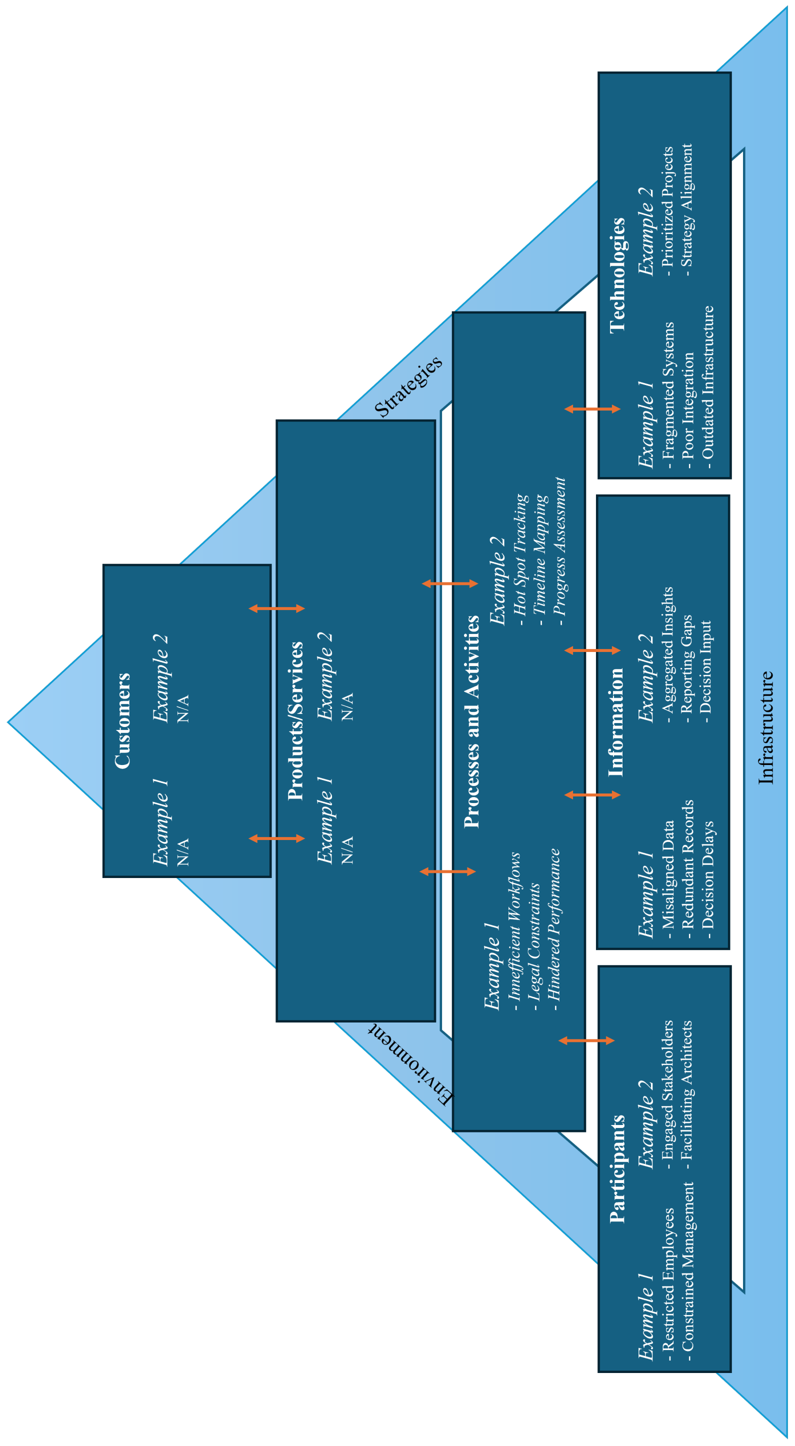

Fig. 2. Exemplary Illustration of Two EA Debts in the WST Framework

4 Work System Theory to Categorize Enterprise Architecture Debt

We illustrate the application of WST to categorize EA debts by two examples (cf. figure 2) from the original publication [14]: The first example centers on an automotive supplier specializing 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 the 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 allows the organization to track and manage the resulting misalignments more deliberately, rather than treating them 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 EA Debt 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.

The second example 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 "EA Debt," 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 enterprise-wide transformation toward strategic objectives, making the term technical debt inapplicable.

The organization has structured its EA into a set of focal areas called hot spots to operationalize its architectural strategy. Each hot spot corresponds to a key domain within the IT strategy, ranging from infrastructure and application modernization to data governance or process redesign. For each hot spot, 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 hot spot. These interviews yield qualitative insights and quantitative indicators, capturing how far 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 EA Debt 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 indicates that some processes are not performing as intended, creating 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 hot spots could result in participant-related EAD if human factors undermine progress.

- *Information*: Aggregating qualitative and quantitative feedback into executive reports is crucial 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 hot spot likely involve updates or replacements of legacy systems, data platforms, or infrastructure. Misalignment between architectural goals and technological implementation's actual pace or scope 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 hot spot. 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 sync 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 intention. Framing this challenge as EA Debt 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.

5 Future Directions

Existing research on identifying EA debt covers several elements that align with the categories of WST, offering a holistic perspective on how organizations can understand and manage their architectural shortcomings. This research can be differentiated into works identifying potential symptoms of EA debt, so-called EA Smells [29, 19, 36], and works trying to find efficient ways for the identification of EA debt that produce potential EA debts as a byproduct [17, 9].

The *people* element is addressed through the roles and perspectives of EA stakeholders. Research [17] highlights the importance of involving stakeholders in identifying and prioritizing EA debt through workshops and interviews. This collaboration is essential for bridging the gap between technical and business perspectives, ensuring that EA debt is addressed with a shared understanding.

Processes are another focus [19]. 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 enterprise processes' overall quality and reduce the long-term impact of inefficiencies.

The *technologies* aspect of EA debt has received significant attention, particularly in terms of legacy systems and poorly integrated IT infrastructure. Studies [29, 36] have adapted concepts like 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 EA are recognized as important contributors to debt. Issues such as incomplete documentation, outdated data, and misaligned information repositories often exacerbate architectural challenges. Research [9] 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 EA debt. Studies [17] have shown how factors like 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 EA debt 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 EA debt [14]. Frameworks and metrics are being developed to measure and address this misalignment, with catalogs of EA debts 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 EA debt, 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 EA debt. 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 EA debt. For external constraints, while regulatory and legal obligations are recognized as contributors to EA debt, there is insufficient research on frameworks for balancing these constraints with architectural flexibility. Finally, in the alignment of purpose, 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 influence 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 EA debt more effectively while aligning with dynamic business needs and external pressures.

6 Conclusion

Within this work, we explored the concept of EA debt through the lens of WST. While EA debt has traditionally been understood as an extension of technical debt, its broader implications across processes, technologies, information, and stakeholder engagement necessitate a more structured theoretical foundation. By categorizing EA debt within the WST framework, this study provides a structured approach to identifying, assessing, and addressing these inefficiencies, ensuring that organizations can make informed decisions about their EA.

Our findings highlight that EA debt manifests across multiple dimensions, including process inefficiencies, technological fragmentation, misaligned information flows, and stakeholder constraints. The case examples demonstrated how trade-offs between economic efficiency and legal obligations and misalignment between an organization's actual and target architecture contribute to the accumulation of debt. We also identified gaps in existing research, particularly regarding the role of stakeholder incentives, real-time data governance, and adaptive frameworks for managing EA debt in dynamic environments.

Key takeaways from this research include the importance of systematically identifying EA debt 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, leveraging emerging technologies, and refining methodologies for effectively quantifying and managing EA debt. Organizations can adopt a more structured and proactive approach to ensuring sustainable and agile enterprise architecture by advancing our understanding of EA debt through WST.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Addicks, J.S., Appelrath, H.J.: A Method for Application Evaluations in Context of Enterprise Architecture. In: Proceedings of the 2010 ACM Symposium on Applied Computing. pp. 131–136. SAC '10, ACM, New York, NY, USA (2010)
2. Alexander, P., Hacks, S., Jung, J., Steffens, U., Uludag, Ö., Lichter, H.: A framework for managing enterprise architecture debts - outline and research directions. In: 10th EMISA. vol. 2628, pp. 5–10. CEUR-WS.org (2020)
3. Alter, S.: Work system theory: Overview of core concepts, extensions, and challenges for the future. JAIS **14**, 72–121 (02 2013)

4. Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P.: The financial aspect of managing technical debt: A systematic literature review. Information and Software Technology **64**, 52–73 (2015)
5. Boh, W.F., Yellin, D.: Using enterprise architecture standards in managing information technology. Journal of Management Information Systems **23**(3), 163–207 (2006)
6. Cunningham, W.: The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger **4**(2), 29–30 (1993). https://doi.org/10.1145/157710.157715
7. Currie, W.: Contextualising the it artefact: towards a wider research agenda for is using institutional theory. Information Technology & People **22**(1), 63–77 (2009)
8. Curtis, B., Sappidi, J., Szynkarski, A.: Estimating the Principal of an Application's Technical Debt. IEEE Software **29**(6), 34–42 (2012). https://doi.org/10.1109/MS.2012.156
9. Daoudi, S., Larsson, M., Hacks, S., Jung, J.: Discovering and assessing enterprise architecture debts. CSIMQ (35), 1–29 (Jul 2023)
10. Day, G.S., Schoemaker, P.J.: Adapting to fast-changing markets and technologies. California Management Review **58**(4), 59–77 (2016)
11. Gampfer, F., Jürgens, A., Müller, M., Buchkremer, R.: Past, current and future trends in enterprise architecture—A view beyond the horizon. Computers in Industry **100**, 70–84 (9 2018)
12. Gregor, S.: The nature of theory in information systems. MIS Quarterly **30**(3), 611–642 (2006)
13. Hacks, S., Brosius, M., Aier, S.: A case study of stakeholder concerns on eam. In: 21st EDOCW. pp. 50–56 (2017)
14. Hacks, S., Höfert, H., Salentin, J., Yeong, Y.C., Lichter, H.: Towards the definition of enterprise architecture debts. In: 23rd EDOCW. pp. 9–16 (2019)
15. Hacks, S., Jung, J.: A first validation of the enterprise architecture debts concept. In: Enterprise, Business-Process and Information Systems Modeling. Springer (2023)
16. Iyamu, T., Sekgweleo, T.: Information systems and actor-network theory analysis. IJANTT **5**(3), 1–11 (2013)
17. Jung, J., Hacks, S., de Gooijer, T., Kinnunen, M., Rehring, K.: Revealing common enterprise architecture debts: Conceptualization and critical reflection on a workshop format industry experience report. In: 25th EDOCW. pp. 271–278 (2021)
18. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical Debt: From Metaphor to Theory and Practice. IEEE Software **29**(6), 18–21 (2012). https://doi.org/10.1109/MS.2012.167
19. Lehmann, B., Alexander, P., Lichter, H., Hacks, S.: Towards the identification of process anti-patterns in enterprise architecture models. In: 8th QUASOQ. CEUR Workshop Proceedings, vol. 2767, pp. 47–54. CEUR-WS.org (2020)
20. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. Journal of Systems and Software **101**, 193–220 (2015). https://doi.org/10.1016/j.jss.2014.12.027
21. Liss, L., Kämmerling, H., Alexander, P., Lichter, H.: Towards a Catalog of Refactoring Solutions for Enterprise Architecture Smells. In: Gan, B., Ouh, E.L., Wadhwa, B., Chawla, S., Lichter, H., Aydin, S., Sunetnanta, T., Anwar, T. (eds.) Joint Proceedings of SEED 2021 & QuASoQ 2021 co-located with 28th Asia Pacific Software Engineering Conference 2021, Taipei [Virtual], December 6, 2021. CEUR Workshop Proceedings, vol. 3062, pp. 60–69. CEUR-WS.org (2021)

22. Luna-Reyes, L.F., Zhang, J., Ramón Gil-García, J., Cresswell, A.M.: Information systems development as emergent socio-technical change: a practice approach. European Journal of Information Systems **14**(1), 93–105 (2005)
23. Matavire, R., Brown, I.: Investigating the use of "grounded theory" in information systems research. In: SAICSIT '08. p. 139–147. ACM, New York, NY, USA (2008)
24. McBride, N.: Chaos theory as a model for interpreting information systems in organizations. Information Systems Journal **15**(3), 233–254 (2005)
25. Nord, R.L., Ozkaya, I., Kruchten, P., Gonzalez-Rojas, M.: 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. pp. 91–100 (2012). https://doi.org/10.1109/WICSA-ECSA.212.17
26. Palvia, S.C., Sharma, R.S., Conrath, D.W.: A socio-technical framework for quality assessment of computer information systems. IMDS **101**(5), 237–251 (2001)
27. Reinking, J.: Contingency theory in information systems research. Information Systems Theory pp. 247–263 (2012)
28. Rinta-Kahila, T., Penttinen, E., Lyytinen, K.: Getting trapped in technical debt: Sociotechnical analysis of a legacy system's replacement. MISQ **47**(1) (2023)
29. Salentin, J., Hacks, S.: Towards a catalog of enterprise architecture smells. In: 15. Internationalen Tagung Wirtschaftsinformatik. pp. 276–290. GITO Verlag (2020)
30. Seaman, C., Guo, Y.: Measuring and monitoring technical debt. Advances in Computers **82**, 25–46 (2011). https://doi.org/10.1016/B978-0-12-385512-1.00002-5
31. Slupczynski., A., Alexander., P., Lichter., H.: A process for evaluating the prudence of enterprise architecture debts. In: 25th ICEIS. pp. 623–630. SciTePress (2023)
32. Slupczynski, A., Hacks, S.: Towards a knowledge base of terms on enterprise architecture debt. In: International Conference on Enterprise Design, Operations, and Computing (EDOC). pp. 194–210. Springer (2023)
33. Smajevic, M., Hacks, S., Bork, D.: Using Knowledge Graphs to Detect Enterprise Architecture Smells. In: Serral E., Stirna J., Ralyté J., Grabis J. (eds.) The Practice of Enterprise Modeling. PoEM 2021. Lecture Notes in Business Information Processing, vol. 432, pp. 48–63. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91279-6_4
34. Tatnall, A.: Actor-network theory as a socio-technical approach to information systems research. In: Socio-technical and human cognition elements of information systems, pp. 266–283. Igi Global (2003)
35. Tatnall, A.: Actor-network theory in information systems research. In: Encyclopedia of Information Science and Technology, pp. 42–46. IGI Global (2005)
36. Tieu, B., Hacks, S.: Determining enterprise architecture smells from software architecture smells. In: 23rd CBI. vol. 02, pp. 134–142 (2021)
37. Uludağ, Ö., Kleehaus, M., Xu, X., Matthes, F.: Investigating the role of architects in scaling agile frameworks. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC). pp. 123–132. IEEE (2017)
38. Uludag, Ö., Reiter, N., Matthes, F.: What to Expect from Enterprise Architects in Large-Scale Agile Development? A Multiple-Case Study. In: 25th AMCIS (2019)
39. Verdecchia, R., Kruchten, P., Lago, P.: Architectural technical debt: A grounded theory. In: 14th ECSA. pp. 202–219. Springer (2020)
40. Vial, G., Rivard, S.: Conceptualizing information systems development as an organizational routine: Implications and avenues for research. SIGMIS Database **53**(3), 91–107 (2022)
41. Volkoff, O., Strong, D.M.: Affordance theory and how to use it in is research. In: The Routledge companion to MIS, pp. 232–245. Routledge (2017)

42. Wiesche, M., Jurisch, M.C., Yetton, P.W., Krcmar, H.: Grounded theory methodology in information systems research. MIS quarterly **41**(3), 685–A9 (2017)
43. Winter, M., Brown, D., Checkland, P.: A role for soft systems methodology in information systems development. EJIS **4**(3), 130–142 (1995)
44. Winter, R., Fischer, R.: Essential layers, artifacts, and dependencies of enterprise architecture. In: 2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06). pp. 30–30 (2006). https://doi.org/10.1109/EDOCW.2006.33
45. Yeong, Y.C., Hacks, S., Lichter, H.: Prioritization of EA debts facilitating portfolio theory. In: Lichter, H., Fögen, K., Sunetnanta, T., Anwar, T. (eds.) 7th QUASOQ. CEUR Workshop Proceedings, vol. 2511, pp. 45–52. CEUR-WS.org (2019)