MIS Quarterly

RESEARCH ARTICLE

GETTING TRAPPED IN TECHNICAL DEBT: SOCIOTECHNICAL ANALYSIS OF A LEGACY SYSTEM'S REPLACEMENT[1]

Tapani Rinta-Kahila
UQ Business School, The University of Queensland,
St. Lucia, QLD, AUSTRALIA {t.rintakahila@uq.edu.au}

Esko Penttinen
School of Business, Aalto University,
Espoo, FINLAND {esko.penttinen@aalto.fi}

Kalle Lyytinen
The Weatherhead School of Management, Case Western Reserve University,
Cleveland, OH, U.S.A. {kalle@case.edu}

Organizations replace their legacy systems for technical, economic, and operational reasons. Replacement is a risky proposition, as high levels of technical and social inertia make these systems hard to withdraw. Failure to fully replace systems results in complex system architectures involving manifold hidden dependencies that carry technical debt. To understand how a process for replacing a complex legacy system unfolds and accumulates technical debt, we conducted an explanatory case study at a local manufacturing site that had struggled to replace its mission-critical legacy systems as part of the larger global company's commercial-off-the-shelf (COTS) system implementation. We approach the replacement as a sociotechnical change and leverage the punctuated sociotechnical information system change model in combination with the design-moves framework to analyze how the site balanced creating digital options, countering social inertia, and managing (architectural) technical debt. The findings generalize to a two-level (local/global) system-dynamics model delineating how replacing a deeply entrenched mission-critical system generates positive and negative feedback loops within and between social and technical changes at local and global levels. The loops, unless addressed, accrue technical debt that hinders legacy system discontinuance and gradually locks the organization into a debt-constrained state. The model helps managers anticipate challenges that accompany replacing highly entrenched systems and formulate effective strategies to address them.

Keywords: Legacy system, COTS system, IS discontinuance, IS replacement, organizational change, technical debt, PSIC model, design moves, process theory, system dynamics

We are stuck in an eternal cycle where we don't get rid of the old systems because we don't have anything new to start building on, while the old system still exists and serves us so well.

—A manager at the case company

Introduction

Commercial-off-the-shelf (COTS) systems promise firms increased agility and scalability (Davenport, 1998; Shang & Seddon, 2002) by giving them new digital options: IT-enabled capabilities that open up novel ways of operating (Sambamurthy et al., 2003; Woodard et al., 2013). Also, COTS systems decrease firms' technical inertia—change-

[1] Likoebe Maruping was the accepting senior editor for this paper. Narayan Ramasubbu served as the associate editor.

resistance caused by technical systems' rigidity—by harmonizing systems across sites and externalizing most of the system maintenance and enhancements to system providers (Berente et al., 2016). Because of these benefits, implementing company-wide COTS solutions has remained an attractive option and managers continue to make investments following such logic—especially in global organizations with multiple sites. In most of these organizations, deploying the chosen COTS system entails replacing local legacy systems (Aanestad et al., 2017), which embody the business logic inherited from a local site (Holland et al., 1999). These systems typically become deeply embedded in the site's operations and infrastructure (Fürstenau et al., 2019). Therefore, in addition to efforts to implement the new COTS system, the replacement process involves deciding when and how the organization will discontinue the use of its legacy systems and address the technical and social inertia associated with the replacement process (Furneaux & Wade, 2011, 2017).

Alas, many organizations fail to fully remove their legacy systems, even after pouring significant effort to the task (Advanced, 2021). They often become unconsciously "caught between" the old and new systems for an extended time, having to manage growingly complex system architectures in which the old and new systems/components interact and operate simultaneously (Aanestad et al., 2017; CRN, 2012; Gómez, 2020). This state burdens the organization with a constant accumulation of technical debt—maintenance obligations that make further system modifications increasingly difficult, risky, and costly (Rolland et al., 2018; Cunningham 1992). The "caught-between" state constrains agility and scalability originally pursued via the COTS implementation (Deloitte, 2021) and exposes the organization to increased security risks and technical instability (Ramasubbu & Kemerer, 2016). How does such an outcome emerge irrespective of the initial intent to achieve the opposite?

When implementing a COTS system, managers must first overcome social inertia—the social system's (people, processes, hierarchies) resistance to change—arising from discrepancies between the new COTS functions (and system logic) and legacy systems' ingrained operations (Arvidsson et al., 2014). Also, they have to mitigate the technical risks of the new system failing to function properly (Avison et al., 2006). Therefore, implementation entails decisions on how to balance the need to customize the COTS solution to current operations against the need to adapt the social system to the new COTS functionality (Berente et al., 2010, 2019; Davenport, 1998). These challenges are more acute in large multinational, multi-site, and federated organizations implementing a common COTS system. In such situations, the COTS solution often fails to align with the many, varied local needs and practices (Arvidsson et al., 2014; Berente & Yoo, 2012; Rolland & Monteiro, 2002). Discontinuing legacy systems in such settings imposes further challenges, since those systems embody deeply entrenched cognitive assumptions and behaviors rooting the systems tightly in the local social setting (Mehrizi et al., 2019).

During implementation, managers must also address the technical inertia accompanying the legacy systems' monolithic nature. The extent of such inertia can be conceptualized via measurements of the various forms of technical debt of said systems. Future maintenance obligations created by system design and implementation decisions typically manifest themselves in poor documentation, "spaghetti code," loss of relevant personnel, or performance bottlenecks (Li et al., 2015). Thus, in lieu of adopting a "big bang" approach, managers typically replace local legacy systems in stages, with each round in the "staggered" implementation replacing a selected set of legacy systems or components to reduce the risks and address related technical inertia (Rolland & Lyytinen, 2021).

Becoming "caught between" is arguably a complex function of the technical and social inertia associated with the new COTS system implementation, consequent decisions about customization and social adaptation, and how all of these shape legacy system discontinuance. To avoid caught-between outcomes, managers must overcome both sorts of inertia and understand their interactions over time. However, the pertinent literature has thus far tackled only the challenges brought by each of the two separately. The software-engineering literature (e.g., Almonaies et al., 2010; Bisbal et al., 1999) approaches legacy systems' replacement as a technical problem of how to overcome technical inertia while ignoring the role and impact of social inertia. In the information systems (IS) domain, meanwhile, organizational COTS implementation studies (e.g., Berente et al., 2019; Rivard & Lapointe, 2012) and nascent IS discontinuance research (Mehrizi et al., 2019, 2022) focus on how to overcome social inertia but neglect the role of technical inertia and how it shapes (and is shaped by) social inertia. Though some recent studies have examined interactions among technical inertia, social inertia, and digital options during system implementation (Fürstenau et al., 2019; Rolland et al., 2018), they have thus far not addressed the challenges that legacy systems' withdrawal brings. A crucial knowledge gap is evident since understanding these dynamics in the context of legacy system replacement helps managers (especially at large, global, multi-site organizations) navigate the risky proposition of discontinuance and avoid undesirable caught-between situations.

Thus, we raise the following research questions: (1) How and to what extent do social inertia, technical inertia, and digital options interact in the replacement of legacy systems with COTS ones (especially in organizations with multiple sites), and (2) how and under what conditions does the state of being "caught between" emerge and stabilize during this process? The lack of theory and empirics related to this topic motivated us to conduct an explanatory case study at a site of a global manufacturing firm that got caught between new COTS solutions and components of its legacy system. We used the punctuated sociotechnical IS change (PSIC) model (Lyytinen & Newman, 2008) as a sensitizing device to identify implementation equilibria between social and technical elements (inertia) and employed Woodard et al.'s (2013) design-moves concept to connect these sociotechnical equilibria with shifts in the level of (architectural) technical debt and digital options. The analysis culminated in formulating a two-level system-dynamics model that sheds light on caught-between outcomes by delineating a complex set of dynamic interactions between social inertia, technical inertia, and digital options at the local and global level.

Theory Review

This section reviews three crucial facets that characterize the legacy system replacement process, the success of which depends on overcoming both technical and social inertia to bring about a new sociotechnical equilibrium in the organization. First, we approach legacy systems as technically inert technical objects. This view is common in software engineering research, which focuses on technical debt in software assets and develops system-migration methods (e.g., Adolph, 1996; Bisbal et al., 1999). Second, we examine IS research focused on organizational system implementation. This research has focused on social inertia associated with bringing a new system into use (e.g., Lyytinen & Newman, 2015) and withdrawing an old one from use (e.g., Mehrizi et al., 2019). Finally, we review research focusing on the interplay between technical debt and digital options during system evolution—the two dimensions that need to be balanced during a COTS system implementation involving the discontinuance of an old system. Table 1 outlines the relevant research streams, their foci, and omissions.

Technical Inertia in System Replacement

Legacy systems display technical inertia because they were originally developed for specific local needs and have since been maintained in a reactive, evolutionary manner. The concept of technical debt captures the inertia likely to be faced when the system undergoes a major modification or replacement. Broadly understood, technical debt represents compromises, errors, lapses, and omissions made during the design and implementation of software that unnecessarily increase its complexity (by introducing unwarranted dependencies) and therefore add to maintenance obligations (Cunningham, 1992). Technical debt typically has to be repaid, with interest, through later refactoring, rewriting, and optimization of various parts of the code base; via documentation improvements; and/or by reorganizing and simplifying the system architecture (Li et al., 2015). Notwithstanding these efforts, completely eliminating technical debt is virtually impossible, since such dependencies are hard to detect and are costly and risky to remove.

Technical debt accumulates insidiously at multiple levels of software activity (Alves et al., 2016; Li et al., 2015; Rios et al., 2018): it increases at the code level when a programmer goes against good coding practices by introducing shortcuts; it grows during testing and documentation if the activities are carried out in an undisciplined manner; and, at the infrastructure level, debt grows from using abandoned and/or obsolete technological platforms such as archaic programming languages and hardware. "People debt" arises when critical system expertise depends on just a few people and becomes acute when those people retire or switch jobs (Sandborn & Prabhakar, 2015). While legacy systems tend to manifest multiple forms of debt (Brown et al., 2010; Holvitie et al., 2018), these systems' monolithic architecture means that *architectural debt* is a particularly relevant form of it in the amassing of technical inertia during system replacement.

Architectural debt accumulates when new unsystematic latent dependencies are introduced—intentionally or not—across the organization's software systems. This might happen, for example, when the system architecture is poorly modularized or when "technology gaps" are introduced by attempts at integrating the latest systems with legacy ones (Besker et al., 2018; Kruchten et al., 2012). The staged and piecemeal manner in which large and complex legacy systems are replaced increases the likelihood of architectural debt amassing as shortcuts and ad hoc solutions pile up. Architectural debt further encourages other types of technical debt because new latent dependencies between systems are often coded haphazardly and left undocumented (Mäki et al., 2023; Rolland & Lyytinen, 2021). These dependencies consequently decrease overall software reliability (Ramasubbu & Kemerer, 2016), incur hidden maintenance costs (Gangadharan et al., 2013), and increase software rigidity (Fürstenau et al., 2019). At the same time, the organization's exposure to risks stemming from cumulative architectural debt remains an abstract and remote concern because of its latent and complicated nature (Holvitie et al., 2018; Martini et al., 2015).

Table 1. Summary of Research Informing Legacy System Replacement

Phenomenon of interest	Literature stream	Examples	Focus	Problems/omissions
Technical inertia in system replacement (i.e., the technical system's resistance to change)	Technical debt	Banker et al., 2020; Besker et al., 2018; Holvitie et al., 2018; Li et al., 2015; Martini et al., 2015; Ramasubbu & Kemerer, 2016	How to conceptualize and measure technical inertia accumulating during software development and maintenance	Does not consider organization-level replacement decisions and processes, related social inertia, or how to realize digital options (system use)
	System discontinuance antecedents	Furneaux & Wade, 2011, 2017	What strategic considerations lie behind managers' intention to replace legacy systems	Does not consider how complex sociotechnical replacement processes will unfold; applies a narrow conceptualization of technical inertia
	Legacy system replacement methods	Adolph, 1996; Almonaies et al., 2010; Bisbal et al., 1999; Ning et al., 1994	How to conduct legacy technical system replacement and related legacy data migration to a new system	Does not provide evidence of replacement's success; does not consider social inertia; applies a narrow conceptualization of technical inertia
Social inertia in system replacement (i.e., the social system's resistance to change)	System discontinuance processes	Mehrizi et al., 2019, 2022	How to dismantle the social mechanisms that keep legacy systems in place during system replacement	Does not consider technical inertia or transformative changes necessary to unlock digital options
	System implementation processes	Arvidsson et al., 2014; Avison et al., 2006; Berente et al., 2019; Lyytinen & Newman, 2015; Newman & Zhao, 2008; Poon & Wagner, 2001; Rivard & Lapointe, 2012; Sarker & Lee, 1999; Wagner, 2010	How to assimilate a new (COTS) system into an organization's incumbent social system	Does not consider technical inertia during replacement other than how errors and use challenges are handled; does not recognize the challenge of discontinuing legacy systems
Balancing technical debt and digital options in system replacement	Digital infrastructure management	Rolland et al., 2018; Rolland & Lyytinen, 2021	How technical debt and digital options interact in implementing and managing digital infrastructures	Does not consider the challenge of discontinuing legacy systems; offers limited consideration of social inertia
	Digital product strategy	Woodard et al., 2013	How technical debt and digital options interact in making strategic (technical) changes to digital products	Does not consider social inertia or sociotechnical processes of system replacement

Factor-oriented IS discontinuance research has suggested that architectural debt weakens managers' intentions to replace legacy systems (Furneaux & Wade, 2011, 2017). In software-engineering research, scholars have examined means for system replacement by developing methods, tools, and techniques to facilitate migration from a legacy system to a new environment (Adolph, 1996; Almonaies et al., 2010; Bisbal et al., 1999). Neither stream has conceptualized the causes of growing technical inertia and its evolution during system replacement. They examine system replacement mainly as a technical and economic problem (i.e., from the cost/risk angle) while failing to consider how the legacy systems' social embeddedness and consequent social inertia shape the scope and pace of staged replacement and the growth of technical debt.

Social Inertia in System Replacement

Legacy systems reflect the organization's strategic intent and operation logic at the time of their introduction. Over time, this logic becomes socially entrenched in the organization, creating strong sociotechnical equilibrium, wherein people, tasks, and structure align with the legacy system operations (Holland et al., 1999; Mehrizi et al., 2022). Mehrizi et al. (2019) identified five self-reinforcing mechanisms that maintain this equilibrium and hinder legacy systems' discontinuance, or "unlearning" of the old system, by governing the behaviors and disposition of users, designers, and managers. The mechanisms are: indwelling, where users become oblivious to the system's underlying structures; legitimization, where the system becomes widely accepted; learning, in which the users learn to utilize the system effectively; resource complementarity, where additional, complementary system investments are made; and routinizing, wherein extensive system use lowers overall coordination costs. Successful discontinuance necessitates dismantling these mechanisms that "freeze" the incumbent equilibrium in place. Such a process unfolds in four phases: (1) realization, wherein the legacy system is scrutinized and discredited to decrease social inertia by eliminating indwelling and legitimization; (2) reversion, wherein social inertia is temporarily amplified by relegitimizing and further developing the old system so as to facilitate its upcoming termination; (3) handover, wherein the old system is connected to the system being introduced, thereby amplifying inertia, and resources reallocated to the latter so as to remove inertia; and (4) marginalization, wherein the legacy system's use is unlearned and deroutinized for purposes of removing any remaining social inertia associated with the old system. While people's legacy IS habits often complicate system discontinuance, such habits may also facilitate letting go of the old system if the new sociotechnical configuration enabled by the system replacement reveals those habits as dysfunctional (Mehrizi et al., 2022).

IS implementation research, in turn, focuses on the presence and impact of social inertia that emerges from introducing a new COTS system (related to the challenge of learning to use the new system). Extensive research suggests that implementation success comes when the organization's social system—people, task, and structure—is aligned with the new COTS system (e.g., Berente et al., 2019; Lyytinen et al., 2009; Rolland & Monteiro, 2002; Strong et al., 2014), producing a new sociotechnical equilibrium (Lyytinen & Newman, 2008). Absent such change, the organization may succeed in technically implementing the COTS system and may thereby "officially" migrate from the old system to the new one but, in this state, it fails to reap the benefits of the new system because it is not used as intended, or not used at all, on account of disequilibrium (e.g., Berente & Yoo, 2012; Poon & Wagner, 2001). In extreme cases, failing to overcome social inertia results in implementation "failure" where the disequilibrium grows so strong that the organization reverts to using the old system and recreates the old equilibrium (e.g., Avison et al., 2006; Lyytinen & Hirschheim, 1987; Newman & Zhao, 2008; Sarker & Lee, 1999). Occasionally, organizations' local sites will tailor the new system to their unique operation practices in order to sustain the old equilibrium and thus minimize the social inertia associated with the new system implementation (Davenport, 1998). Such decisions imply costly local customization that adds to the technical debt and may result in the failure to actualize the global options/benefits sought (Arvidsson et al., 2014).

IS implementation research predominantly assumes that discontinuing old systems is not problematic (Rinta-Kahila, 2018), in that the new COTS system will provide digital options superior to those offered by the incumbent solution. Yet legacy systems or parts of them often remain in place because of staggered implementation or because users resort to using these "shadow systems" in response to the new system's deficiencies (Berente et al., 2019; Lyytinen & Newman, 2015). In summary, although technical inertia associated with legacy systems complicates their discontinuance (Fürstenau et al., 2019), studies of IS discontinuance and implementation have generally ignored how technical inertia interacts with social inertia and how their interactions shape digital options during a staggered system replacement.

Architectural Debt and Digital Options during Systems Replacement

Technical debt, especially architectural debt, offers a useful lens to examine the level of an organization's technical inertia during legacy system replacement. Fully replacing some legacy system(s) with a COTS solution will offload most of the technical debt to the COTS provider. Still, high (or unknown) levels of debt and the organization's propensity to accumulate more of it during the transition can complicate such efforts. Research on the interplay between technical debt and digital options recognizes that consciously planting debt can be beneficial if it gives the organization digital options—such as speed, scalability, or flexibility—that it would not have otherwise (Rolland et al., 2018). Increased agility afforded by digital options helps organizations respond swiftly to emerging opportunities and change their business model, reengineer their key processes, and/or restructure the main organizational tasks (Sambamurthy et al., 2003). Managing technical debt and digital options jointly as a system-level response consists of balancing the contradictory demands they impose for business: planting debt opens new options in the short term but the resulting technical inertia rules out new options in the long term.

Often, large organizations implementing new systems pursue their global strategic intent and related digital options but such intent is frequently hard to channel down to all operation units. After all, local sites need to reconcile the discrepancies between the proposed new COTS system and the still useful legacy system (Arvidsson et al., 2014; Rolland & Monteiro, 2002). Because allocating resources to debt elimination is seen as limiting the exploration of new options, managers typically gravitate toward sustaining the outdated system architectures, whereby they inadvertently accumulate architectural debt beyond the point of obsolescence (Furneaux & Wade, 2017). Managers receive greater rewards for reaching short-term goals by unlocking new digital options within the limits of set deadlines and budgets than for allocating resources to difficult-to-understand long-term technical endeavors that remove debt.

Overall, research into legacy system replacement needs to cater to the dynamic relationship between architectural debt and digital options during system replacement. Woodard et al. (2013) conceptualized these dynamics via the notion of *design moves:* strategic actions that alter the technical structure or functions of one or more IT systems. The authors applied a two-dimensional "design capital" map to illustrate the effects of each move on the options/debt ratio, where every move is "represented as a vector …, indicating the extent to which the move increases or decreases the option value of a firm's designs and increases or decreases the firm's technical debt" (Woodard et al., 2013, p. 541). The map's four high/low quadrants represent alternative regions that an organization can occupy during a system replacement, depending on the debt and options inherited from past design moves. The quadrants are referred to as (1) "option constrained" (low debt, few options), (2) "low quality" (high debt, few options), (3) "debt constrained" (high debt, many options), and (4) "high quality" (low debt, many options). Ideally, organizations execute moves that decrease debt and increase their options in the long term (e.g., selecting a given COTS solution globally). This is expected to result in systems that yield higher quality (less debt) and greater value (more options). In reality, COTS implementation and related system replacement can result in unconstrained debt accumulation, given the discrepancies between local and global needs and the various managerial incentives that constantly pile on debt to create digital options. This process can push the firm into an unintended debt-constrained state (many options but high debt).

Generally, a staged COTS implementation can be framed as a series of design moves wherein each move results in varying options/debt ratios for the implementing organization. However, system-replacement studies have not utilized this idea of dynamic debt/option ratios across implementation stages: the notion of design moves has so far been applied mainly to studying the evolution of digital product designs (Woodard et al., 2013). Because legacy systems come with both technical and social inertia, their replacement involves the mutual adaptation of social and technical elements, not just technical factors. This fact invites us to examine how design moves are applied simultaneously across multiple technical systems during a legacy system replacement process and how the moves affect the social system (and related inertia) and vice versa, along with how consecutive stages shape options and the mounting of architectural debt. To this end, we integrated the design-moves framework with a sociotechnical process lens to analyze system-replacement dynamics and outcomes. This elucidates the process as a form of sociotechnical change that results in several equilibria connected with replacing legacy systems and addressing associated inertia. The analysis aids in understanding how sociotechnical change in the wake of multi-site COTS implementation develops into the surprisingly common outcome of a debt-constrained state wherein parts of local legacy systems remain operational alongside the new system for many years, notwithstanding the organization's initial intention and sustained effort to terminate the legacy system (e.g., CRN, 2012; Hemon-Laurens, 2016; Deloitte, 2013).

Research Method

We deemed a process-based theory appropriate for explaining legacy system replacement as sociotechnical change wherein improved balance is sought between digital options and architectural debt. Process analysis considers identifying critical events and their interactions within their context to explain how the identified outcomes emerge and how the process or focal unit(s) of analysis shift from one state (equilibrium) to another (Markus & Robey, 1988). The approach invites us to inspect social and technical dimensions of change and their interactions during legacy systems' replacement.

For narrating such change and its outcomes, we applied the PSIC model (Lyytinen & Newman, 2008) as a sensitizing device. The model draws on sociotechnical theory (Leavitt, 1964) to account for and explain how organizational change following system implementation dynamically integrates four recursively interdependent components or fails to do so: tasks (work processes), actors (people), structure (organizational arrangements), and technology (software and hardware, such as an enterprise resource planning, ERP, system). It assumes a hierarchical leveling of change, wherein analysis requires framing at several levels: the "work system" (the organization's incumbent work processes supported by the systems in use), the "building system" (resources and activities assembled locally for replacement), and (if

applicable) the higher-level system (global implementation). The model aids in identifying and capturing social inertia/change at multiple analysis levels (local/global) to account for changes in system use and, thereby, whether and how options are realized during the COTS implementation and related replacement process as levels of associated technical debt are ascertained and addressed.

Via the options/debt framework, one can view sociotechnical changes that alter technology as design moves with a specific resulting options/debt ratio. Such process analysis leads to fine-grained analysis of the interplay between architectural debt and digital options associated with gaps between sociotechnical components (e.g., poor technology-task fit), highlighting imbalances and related inertia that the COTS implementers need to address by changing either the social system components or the technology.

The Case Study

We used purposeful sampling (Patton, 1990) to conduct an explanatory case study—an explanation of the emergence of a state or event (Yin, 2018). To this end, we identified a research site that, notwithstanding its best intentions and effort, had struggled to discontinue its legacy systems. We were granted generous access to the site: interviewing all critical stakeholders involved in the system-replacement efforts allowed us to collect a rich dataset covering the changes in the site's systems and their architecture, changes in system use, and implementation outcomes. The site, referred to here as "EngineShop", is a Finnish factory owned by the pseudonymous EngineGroup, a multinational manufacturing company with more than 130,000 employees, worldwide. The group's business focuses on robotics, heavy electrical equipment, and automation technologies. EngineShop itself employs roughly 1,000 people and is one of EngineGroup's main sites for the production of high-quality electric motors and generators. In the early 2000s, EngineGroup launched a strategic initiative to shift from geographically based operations to a global matrix organization. This called for universal adoption of a uniform system architecture and related systems to support the group's financial, manufacturing, and design operations. It involved replacing country- and factory-specific legacy systems to simplify system architecture and enable globally coordinated design and production. For EngineShop, the change meant discontinuing a decade-old legacy system called Driving Glove (DG) and implementing common COTS solutions. After two staggered COTS implementation phases, the site had not succeeded in marginalizing or withdrawing DG. In fact, significant portions of it remain in operation today. This has resulted in a complex system architecture, mounting hidden maintenance costs, and the inability to fully exploit the digital options promised by the common COTS systems.

Collection of Data

The data collection consisted of an orientation phase followed by four rounds of interviews. We started with a group interview to chart current system use at EngineShop and identify possible research topics. Then, the study's focus evolved through several rounds of personal interviews. The early interviews probed the reasons for the site's caught-between state (Round 1, with 12 interviews), but later we narrowed our focus to dissecting the details of the replacement process in light of sociotechnical changes and related system equilibria (Round 2, four interviews). The next interviews examined these outcomes' implications for debt and options (Round 3, four interviews). Data collection concluded with validation of the emerging findings (Round 4, five interviews). Table 2 summarizes the goals for each round and its research implications.

The collection and analysis of the data iteratively followed a realist approach (Van Maanen, 1988). By collecting data in several rounds, with varying foci, we were able to gather multiple accounts of the events from diverse participants. The data collection employed semi-structured interview protocols, one developed for each round. Detailed probing of the transcripts and supplementary materials provided by informants (e.g., charts of process changes and materials pertaining to the implementation projects and outcomes) helped with thick description of the change process and trajectory. All interviews were recorded and transcribed.

Overall, the data collection yielded 400+ pages (172,595 words) of interview transcripts, eight pages of handwritten field notes, and 192 archival documents. Access to conflicting opinions and perspectives on the change and its outcomes, in combination with the sample's broad representation of the organization's roles, tasks, and functions, decreased the threat of biases and errors, enabled triangulation, and provided means for source criticism. In addition, we leveraged several secondary data sources, among them two bachelor's theses and two master's theses reporting on the specific aspects of EngineShop's legacy system replacement effort, including an account of implementing a COTS ERP system (Study 1, 2008) and its design-support tools (Study 2, 2014; Study 3, 2010; Study 4, 2015) between 2008 and 2015.

Table 2. The Data-Collection Rounds and Their Foci		
Round	**Purpose**	**Implications**
Orientation	Chart the current situation and approaches to studying legacy system discontinuance	Specifying the research objective as explaining the state of being trapped between old and new systems
1. Initial understanding of the phenomenon	Investigate why discontinuance had not succeeded	Sociotechnical (S-T) misalignments and inertia as explaining the caught-in-between situation; the PSIC model (Lyytinen & Newman, 2008)
2. Creation of the PSIC narrative	Reconstruct the narrative of events at many analysis levels, from antecedent conditions, through implementations to the "caught-between" situation	Identification of S-T changes and their implications for technical debt (Cunningham, 1992; Kruchten et al., 2012), to capture technical inertia
3. Focused examination of technical debt	Link S-T changes to accumulation of various types of technical debt within the design-moves framework	Digital options as the source of debt accumulation; identification of architectural debt as the debt type to focus on; integration of digital options and architectural debt via the design-moves framework (Woodard et al., 2013)
4. Mapping and validation of design moves	Identify design moves and their impact on digital options and architectural debt, connect the design moves to S-T changes, and validate the outcomes	Elaboration of the system-dynamics model with feedback loops, to explain the caught-between state's emergence and persistence as a function of both technical and social inertia

Data Analysis

We began our analysis with the steps outlined by Lyytinen and Newman (2008) and coded the data for critical events at several levels of analysis, using the software ATLAS.ti. The analysis involved coding for gaps and balances between the sociotechnical components over the study's full time span and identifying sources of social inertia and changes in the overall equilibria of the sociotechnical systems involved (the work system, building system, and global factors) and their vertical interactions. While some changes were identified as punctuated, in that they immediately changed the essence of the sociotechnical work system's "deep structure" (e.g., deploying the COTS system to use), others were incremental, in that only a successive series of them would fundamentally reshape the work system (e.g., modifying the COTS system).

After identifying key changes, we conducted design-moves analysis by assessing the changes' implications for architectural debt and digital options (Woodard et al., 2013). We operationalized past work on design moves (Woodard et al., 2013) and architectural debt (Besker et al., 2018; Li et al., 2015) to determine what constituted increases/decreases in the options and debt resulting from each move. The substantial archival material (system documentation) enabled validation of the informants' recollections and interpretations of the events and outcomes. We examined architectural dependencies by analyzing pre- and post-implementation architecture maps and related models. Likewise, project milestone documents provided evidence of past technical decisions and their rationale. For instance, a pre-study document on the site's "as-is" situation from 2006 confirmed the informants' claims about such strengths of DG as support for well-integrated local information flow and clean, transparent data structures (low architectural debt) that were diligently documented (low documentation debt). It also provided evidence of weaknesses such as poor support for global, multi-site manufacturing (low option value), reliance on just a few people for system development and maintenance (high people debt), and outdated technology (high infrastructure debt). Ultimately, we constructed visual PSIC maps of the two focal implementation processes (see Appendix A). In line with Woodard et al. (2013), we assessed the site's position on the design-capital map before and after each move, with the quadrants representing the option-constrained, low-quality, debt-constrained, and high-quality states (see Figure 2). In total, the analyses produced 991 codes, with 12 higher-level categories and 129 indicators. To validate our findings, we presented the analysis results and our interpretation of the outcomes to a group of site managers. They had no objections to these.

Empirical Findings

Our reporting of the findings has three parts. The first describes the antecedent conditions that triggered EngineGroup's shift toward globally coordinated operations supported by common COTS solutions. Then, we examine how the change initially pushed EngineShop into an option-constrained state, given its system architecture and the state of its legacy systems. Finally, we narrate the legacy system replacement process that followed, using the vocabulary of sociotechnical change and design moves.

Antecedent Conditions

Before the legacy system replacement initiative was launched, EngineShop had utilized DG for a decade. Its use extended to nearly every function of the organization, covering product design, configuration management, production and logistics, accounting and financials, sales, and purchasing. The national IT manager at EngineShop described DG's integration level as "world-class" in the system's heyday (1997-2005): "It was so tailored, fit for purpose, and made precisely for these operations. You couldn't find corresponding functionality in any commercial system or combinations of them." Data flowed automatically from one part of DG to another, then onward to other systems (for financials, logistics, etc.), contributing to high data integrity and quality. The system was diligently customized to support the factory's unique pattern of operations. Its local nature was reflected in its originally Finnish-only interface. Since DG's development was handled by local developers, user requests and local needs could be dealt with rapidly and responsively. This efficiency and flexibility contributed to high user satisfaction and use effectiveness. At the time, the system's use aligned well with the scope of EngineShop's operations: the ability to produce extensively customized high-end products for local customers was deemed the site's central competitive advantage.

Initiation of Change

EngineGroup's Global Initiative

To increase its global market agility, EngineGroup opened several new factories in Asia in the early 2000s. It also overhauled local and geography-based management structures, reorganizing its operations under a networked matrix structure. The chain of responsibility would no longer run vertically through a single geographical location; it would extend laterally across several dimensions locally and globally. Such a global structure required rearranging all product-design and manufacturing operations, which demanded complex and seamless coordination among multiple units across geographical boundaries. For example, an Italian factory would assemble products from components designed in Finland and manufactured in Estonia. As the change progressed, EngineGroup's leadership recognized that operating tailored, local systems at each site did not support the new logic. The local (legacy) systems lacked features enabling the sites to collaborate laterally in an effective and scalable manner; i.e., they did not provide such digital options locally. Also, they precluded cultivating the capabilities EngineGroup needed as a whole to support its strategy and achieve agility globally (i.e., global digital options). Further, the local systems and related architecture made EngineGroup's global IT architecture management complicated, with a significant level of global architectural debt. These issues prompted EngineGroup to replace its sites' legacy systems with a common COTS solution. In addition, to boost agility via common globally interoperable systems, the change outsourced most local system development and maintenance work.

EngineShop's Local Response.

Refined through a decade of use experience, DG's operation reflected a well-balanced sociotechnical system: catering to EngineShop's local needs and tasks, it was aligned with current structure (W1 in figures A1 and A2, in Appendix A). In response to the new global priorities, EngineShop entered the "realization" phase (Mehrizi et al., 2019) by starting to scrutinize DG's functionality with regard to the digital options desired globally. The review made it clear that EngineShop had to "pull the plug" on DG, resulting In the system's formal discrediting. Although architectural debt related to this and other systems had not become visible locally (thanks to DG's tight coupling of processes, local operations, and sufficient support), DG could not provide effective coordination of global product engineering and manufacturing. Hence, EngineShop found itself in an option-constrained state, wherein global strategic demands necessitated replacing DG:

> *We didn't have a choice. We had to do it because EngineGroup says so. And it did not necessarily seem sensible to us. But, of course, we understood that we had had DG for quite a few years already and, technologically speaking, it was approaching the end of its life cycle and issues with technical support might emerge.* (Project Manager)

EngineShop recognized that operating DG reliably depended on a dwindling pool of experts (people debt), and it ran with aging technology that posed future security and stability risks (infrastructure debt). The site's managers saw its replacement as an opportunity for paying off the accumulated debt, whereby future maintenance obligations should be largely shifted to COTS systems' providers.

Over the years, DG had grown organically into an all-encompassing "monolith" touching every process at the site. No single COTS solution could replace it entirely. Therefore, the local management initiated two COTS implementation projects to replace DG. The first would enable global coordination of product engineering by implementing the product-data management (PDM) system by Siemens called Teamcenter. Because of EngineShop's proven high-quality product engineering and its related system-development skills, the global organization chose this as the Teamcenter pilot site. Second, harmonized resource management was to

be achieved through the group-wide deployment of a family of SAP products. Initially, the intent was to replace DG entirely by splitting the monolith into two parts: engineering processes (including product data and the product configurator) would be moved to Teamcenter while an SAP system would handle related administrative processes (orders' fulfillment, document mediation, etc.). Implementing the two COTS systems was expected to lower various forms of technical debt *and* offer new digital options both locally and globally.

The Legacy System Strikes Back: Accumulation of Architectural Debt

Preparations for Teamcenter and SAP implementation began in 2006 and 2008, respectively. Both implementations followed a canonical pattern of staggered implementation activities, environmental events, and organizational responses. Below, we chronicle the two system-implementation projects with regard to both the social changes plus resulting inertia (PSIC analysis) and technical change in relation to digital options and architectural debt (design moves). We show how the change triggered interactions between the two dimensions of change and eventually, contrary to initial expectations, locked the site into a debt-constrained state. The implementation chronologies can be divided into multiple episodes of system upheaval "characterized by the need to reform the deep structure" of the work system (Lyytinen & Newman, 2008, p. 593). Each episode encompasses a significant technical change to the site's IT systems (see Figure 1), bracketed by punctuations that altered the work system's deep structure (via either a single move such as a system implementation or a series of incremental development efforts). The section concludes with discussion of the systems' cascading interactions and the related dependencies.

The Teamcenter Implementation: Social Change and Inertia

The three episodes outlined at the top of Figure 1 constitute the Teamcenter sociotechnical implementation narrative. This involves oscillations in work- and building-system balances/imbalances. Our visual map (see Appendix A's Figure A1) labels these W1-W12 (for the work system) and B1-B12 (for the building system).

Episode 1: Implementation project (2006-2009). EngineShop's extensive as-is analysis of legacy engineering functionality in 2006 revealed that DG's product configurator, encapsulating the site's unique business logic, was too complex to be ported to Teamcenter. Simultaneously, the global managers demanded that the factory's productivity should not decline during the replacement. That created local social inertia, as reflected in the initial building-system gap in the project (B2): the structure established to execute the change lacked sufficient scope and resources for the task of replacing DG's product-engineering functions. The revelation was followed by a "reversion" (Mehrizi et al., 2019), which narrowed the implementation task (scope) (B3): the product configurator and some other parts of the product-engineering environment would remain in use (relegitimized), and Teamcenter would be integrated with them. The site could operate as before and its productivity would not suffer. The IS manager explained,

> *[Migrating all engineering functions] would have been expensive and difficult, so why force it, when all we would have gotten would have been some architectural benefits of having the configurator and PDM in the same package? There was no business case for it.*

This decision put a "cap" on the social change needed and reduced social inertia by decreasing the extent of mandatory alterations in actors, tasks, and structure. The DG configurator was to be replaced with an equivalent COTS module later, once the other PDM functionality and the related ERP components had been implemented.

As the deployment progressed, new gaps emerged in the building system. The project team discovered that counter to initial expectations, Teamcenter could not support the complex data structures of EngineShop's product architecture and related work processes. The resources (time and money) allocated to the project were found to be insufficient for implementing viable Teamcenter functionality at the site (B4). Implementation could proceed only through adjustments to the task and heavy customization of the Teamcenter system to the site's work activities (B5). Additional gaps became visible later when the legacy data were being migrated from DG to Teamcenter (B6) and product data became corrupted due to inconsistencies in data structures between DG and Teamcenter. Compounding the issue, the factory lacked the human resources necessary to rectify the problem. To guarantee a successful "handover" (Mehrizi et al., 2019), EngineShop addressed the gaps between the task, structure, and actors by hiring temporary workers to repair the corrupt data and introducing additional customization (B7). Resource constraints led to many shortcomings in the initial solution (architectural debt) whereby data integrity suffered.

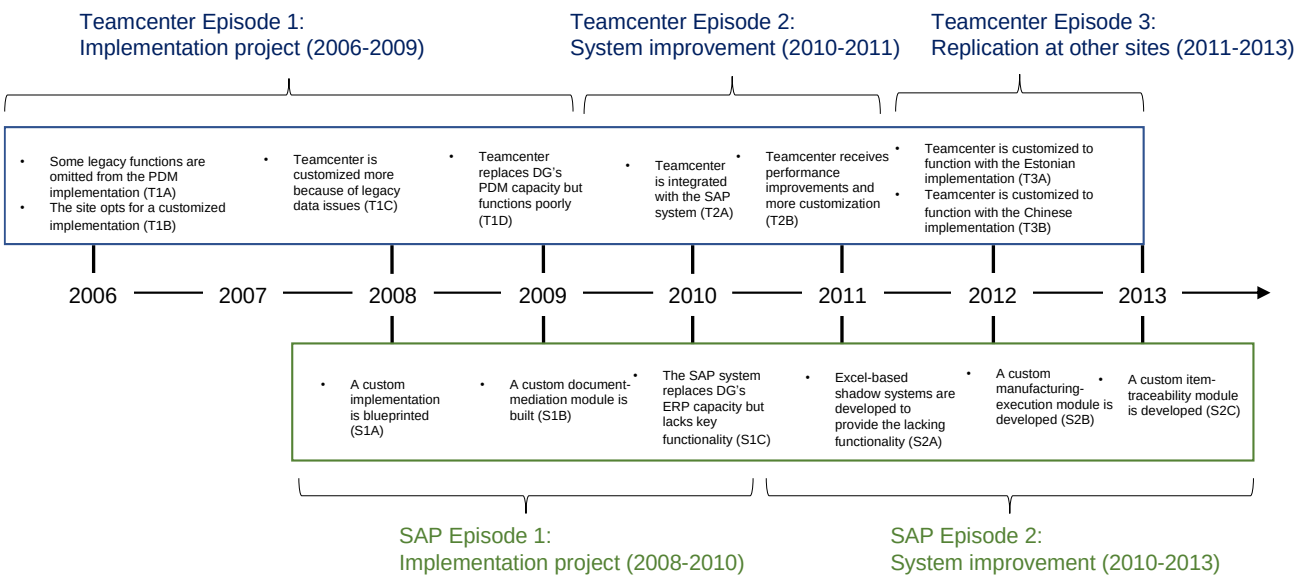

Figure 1. Timeline of Implementation Events and Episodes

Teamcenter replaced DG as the site's PDM system in 2009 (W7), a punctuation that transformed the deep structure of EngineShop's work system. Teamcenter turned out to be inferior in many respects. It was unstable, and its frustratingly slow performance increased lead times. Although the system had been tailored to the legacy system's architecture, the previous smooth integration and support for established product-design processes could not be matched. The new solution added "unnecessary" steps to the design processes, involving extra manual work (e.g., recording the same data in multiple systems, multiple times). Therefore, product engineers resisted the change and vocalized their dissatisfaction with the new system in comparison with DG.

Episode 2: System improvement (2010-2011). By this point, the SAP implementation had begun. Its strict schedule and sheer immensity dictated funneling all available resources to this endeavor rather than continuing efforts to expand and fix Teamcenter's functionality. The global management team specified that the implementation project's high priority precluded the Teamcenter changes identified as required (B8). The Teamcenter application owner explained:

> *Engineering times increased by, I think, as much as 50% … [W]e thought that [Teamcenter] would get improvements straight away, but then the SAP implementation came and took all the development resources …. Then two years passed without any developments.*

From the global perspective, the product-engineering processes supported by Teamcenter appeared to work well enough, so the system was deployed at other sites. Eventually, new resources were provided for making critical incremental changes that gradually improved Teamcenter's performance (B9), albeit with the cost of further customization (W9-W10).

Episode 3: Replication at other sites (2011-2014). Next, EngineGroup sought benefits from a uniform PDM approach by replicating the Teamcenter implementation at its other sites. In 2011-2012, EngineGroup deployed Teamcenter in Estonia and China. However, the sites there had their own legacy environments, calling for additional customization. With the locally oriented tweaks, at EngineShop and other sites alike, it remained challenging to establish a group-wide shared product-engineering environment (W10-W11):

> *We have one shared structure for component lists in Teamcenter. But there are differences [due to customization] in the ways [the structure] behaves when going forward in the process … We might have a need to modify the structure, but then another site says that [its local legacy customizations mean] you cannot modify it.* (Engineering Manager)

The gulf between locally customized systems and global governance structure dictated that EngineShop had to continue customizing its Teamcenter system in pursuit of greater compatibility with other sites.

The Teamcenter Implementation: Technical Change and Inertia

From a technical perspective, the system-implementation narrative represents a string of design moves (prompted by social inertia) with ramifications for EngineShop's digital options and architectural debt. In each move's code in Table 3, the first letter refers to the implementation project ("T" for Teamcenter), the number denotes the episode (e.g., "1" for Episode 1), and the last letter reflects the moves' order in the episode ("A" denotes the first one, etc.).

Episode 1: Implementation project (2006-2009). As noted above, EngineShop was initially in an option-constrained state. During Episode 1, the decision to replace only some DG functions and retain the configuration system created a patchy system architecture with technological gaps foreshadowing a high level of architectural debt (T1A). The technical environment grew increasingly complex as new-system functions had to be integrated with the remaining legacy environment. Similarly, architectural debt was planted during the initial customization of Teamcenter (T1B) and the tailoring that followed through numerous latent nonstandard dependencies and higher complexity (T1C). This left some migrated data in an incompatible format. Debt accrued latently because the new system still had not been fully implemented. For the same reason, the initial moves had no tangible effect on the options available.

After Teamcenter's launch, EngineShop's local options decreased significantly on account of the system's poor or absent functionality, severely constraining the site's engineering performance and ability (T1D). Since the system was not functioning properly even for basic tasks, the collaboration options sought by the global management were not realized. Implementing it also introduced notable architectural debt due to the increasingly fragmented IT architecture, with several compatibility-poor systems, technological gaps from integrating a new system into the legacy environment, and the maintenance and development burden that followed. The conditions resulting from this punctuation pushed EngineShop into a low-quality state.

Episode 2: System improvement (2010-2011). The next episode involved integrating Teamcenter with the SAP system via a customized mediation module. This move produced new options for system interoperability (T2A) but added architectural debt: the module was a home-grown system linking two new (already heavily customized) systems into DG, thus contributing to maintenance complexity and technological gaps. Further customization to improve Teamcenter's functionality ultimately increased the options by rendering the site's engineering capability comparable to that before the implementation (T2B). This, however, came at the expense of adding architectural debt due to escalating deviations from standard system architecture.

Episode 3: Replication at other sites (2011-2013). Customizing Teamcenter to achieve compatibility with the Estonian and Chinese sites (T3A-T3B) furnished EngineShop with new global collaboration options. Still, with this customization, the system strayed further from standard architecture and functionality, rendering system updates increasingly laborious and time-consuming: numerous unsystematic, nonstandard dependencies required separate testing before any major version release. Thus, increases in both digital options and architectural debt brought EngineShop from a low-quality state to a debt-constrained one. The site's ability to exploit the new collaboration options remained limited, however, because local customization at other sites restricted the extent to which Teamcenter's features could truly be utilized.

The SAP System Implementation: Social Change and Inertia

The two episodes shown at the bottom of Figure 1 constitute the SAP-related sociotechnical implementation narrative. That narrative articulates the oscillations involving balances/imbalances in the work system and building system denoted as W1-W10 (work system) and B1-B10 (building system) per the PSIC visual map presented in Appendix A's Figure A2.

Episode 1: Implementation project (2008-2010). Preparations for the SAP system implementation commenced in 2008: "The goal had been that we would not customize [the SAP system] much. 'Keep it simple'; that was our motto." A building-system imbalance was laid bare during the development of the implementation plan when the team realized that no single unified system could support the Finnish production sites' unique business logics (B2). This shattered the initial dream of a "vanilla" SAP system implementation for all Finnish production sites and led to a locally customized implementation (B3).

As the implementation advanced in 2009, it quickly became clear that the lightly customized SAP solution was still not compatible enough with the work processes at EngineShop (B4). Within the constraints set, the team now had to choose either a radical redesign of most work processes to align them with the planned (lightly customized) implementation template or perform heavy customization to support existing processes. This choice opened a widening gap between the actors and task in the building system, as the project managers could not reach agreement on how to proceed (B5).

Table 3. Design Moves in the Teamcenter Implementation

Design move	Episode and strategic intent	Sociotechnical gaps (Social inertia)	Design actions	State of design capital	Impact of design move
T1A	1: Guarantee local productivity while implementing a product-engineering system with global digital options	Building-system structure not supportive of the implementation task, because of incompatibilities within DG work-system structure	Reduce implementation scope by excluding some legacy applications from replacement	Option constrained	Increased debt
T1B		Building-system structure and actors inadequate for the implementation task, for reason of incompatibility with DG work-system structure	Customize Teamcenter for the sustained legacy environment	Option constrained	Increased debt
T1C		Building-system structure and actors not adequate for the implementation task, because of incompatibility with DG work-system structure and technology	Customize Teamcenter for the sustained legacy environment and hire people to fix the legacy system data	Option constrained	Increased debt
T1D		Customizations balancing the building system and enabling Teamcenter implementation	Replace DG's PDM functionality with Teamcenter	Low quality	Abandoned options, increased debt
T2A	2: Create conditions for global collaboration in product engineering while maintaining acceptable levels of local productivity	Incompatibility of the Teamcenter work system's customized technology with the SAP system	Integrate Teamcenter with the SAP system by using a customized module	Low quality	Created options, increased debt
T2B		Incompatibility of the Teamcenter work system's technology with the structure, actors, and task	Customize further and fix significant bugs detected as hampering engineering work	Low quality	Created options, increased debt
T3A	3: Pursue global digital options provided by the new shared system	Incompatibility of the Teamcenter work system's technology with the structure and task, because of the Estonian site's work system	Customize Teamcenter, to create compatibility with the Estonian site	Debt constrained	Created options, increased debt
T3B		Incompatibility of the Teamcenter work system's technology with the structure and task, because of the Chinese site's work system	Customize Teamcenter, for compatibility with the Chinese site	Debt constrained	Created options, increased debt

In the end, the site manager decided on the second option, greatly adjusting the vanilla solution, to accommodate the task-actor-technology relationships on the site. This circumvented the challenge of overcoming significant social inertia but resulted in a significantly customized system that involved a (document) mediation module (B6).

In 2010, the SAP system entered production use and was integrated with Teamcenter (W7). Some critical operations- and manufacturing-related legacy functions not supported by SAP had to be omitted from the implementation's scope to get the main project across the finish line in time. This choice reduced process traceability and complicated the management of critical production resources' utilization, creating additional gaps between system functions and the actor-task relationships. The SAP system offered inadequate support to critical manufacturing processes (task), it was inconsistent with present manufacturing workflows (structure), and therefore workers found adjustment to the new system difficult (actors).

Episode 2: System improvement (2010-2013). For a "quick fix," the local IT group created Excel-based shadow systems to render day-to-day operations viable at some level. Still, the switch from highly integrated legacy tools to patchy makeshift systems created inefficiencies and increased user dissatisfaction. This necessitated further efforts to bridge the evident sociotechnical gaps. Two customized modules were implemented on top of the SAP solution, to replace the shadow systems created. These made manufacturing operations' traceability feasible and reduced the gulf between the SAP system and surrounding workflows (W10).

The SAP System Implementation: Technical Change and Inertia

Table 4 summarizes the design moves triggered by social inertia during the SAP implementation and shows how the moves shaped EngineShop's options and debt. The moves are denoted similarly to those in the Teamcenter narrative above (e.g., S2A is the first move in the second episode).

Episode 1: Implementation project (2008-2010). From the initial option-constrained state, the move to opt for a local customized SAP implementation (S1A) and implement a document-mediation module built in-house (S1B) increased the debt through greater architectural complexity and new technology gaps. The debt remained latent since the system had yet to be implemented. Launching the SAP system (S1C) and decommissioning DG's functionality diminished the range of options by eliminating several core functions of the previous system, including item tracing. Integrating the customized system into the legacy environment realized the latently accumulated debt by generating new dependencies between the SAP system, Teamcenter, and the remaining legacy components, pushing EngineShop into a low-quality state.

Episode 2: System improvement (2010-2013). EngineShop built Excel-based shadow systems to replicate (discontinued) functionality not carried over to the SAP system (S2A). This created options by, e.g., enabling a degree of traceability. It also accumulated debt by introducing siloed, nonintegrated makeshift systems. During design moves S2B and S2C, EngineShop developed internal product-tracing-support modules to replace the makeshift systems. This also restored process productivity to near-DG levels, generating new local options. To sum up, the additions moved EngineShop beyond its previous capabilities, but simultaneously added architectural debt: the new, customized pieces' deviation from the SAP system's standard functionality necessitated additional maintenance, plus testing whenever a system update was planned.

Getting Stuck in a Debt-Constrained State

The accumulation of architectural debt remained largely hidden as the site continued customizing the new COTS solutions. A local IS manager stated that the risk of taking on debt "was acknowledged at a general level, but there had been no sufficient understanding of the gravity of its implications" for system functionality and version updates until most systems were in use. Design moves to bridge sociotechnical gaps implied that EngineShop had to choose between the Scylla of addressing significant social inertia from radical process reengineering and the Charybdis of mounting architectural debt from considerable continued customization. At the time, locally mounting debt was not deemed troublesome globally as long as productivity was maintained and (globally strategic) digital options were realized. Hence, both COTS systems were customized so that significant changes to the social system (structure, task, actors) would not be needed. Customization notwithstanding, misalignment between the COTS systems and the social system drastically reduced local design- and manufacturing-related options. Subsequent customization to restore site productivity led to further architectural debt. The simultaneous void of global pressure for local debt elimination and the growing "combat fatigue" (arising from difficulties in system implementation and use) at the local level provided additional inertia. Accordingly, the local management never dedicated itself to fully removing DG. In consequence, marginalization (Mehrizi et al., 2019) of the replaced legacy functions was not achieved: some senior engineers insisted on using DG's item search functions, which remained as a shadow system after this function had officially been moved to Teamcenter.

After the two implementations, EngineShop consciously strove to reduce architectural debt. In 2011-2020, several full-replacement initiatives were proposed, with such evocative titles as "Kill DG Configurator." None received the global management's approval since the site was unable to make a business case for withdrawing a seemingly well-functioning local system. Another factor in the lack of success was the emerging global development context, which hindered the creation of a local building system that could replace DG (no local slack existed).

Accumulation of architectural debt rendered management of the system increasingly difficult. Ironically, though the local management had viewed the two system implementations as an opportunity to eliminate different forms of technical debt accumulated with DG, the COTS implementations increased architectural debt, and other forms of technical debt too. While the original DG solution was documented fairly well, most of the complex integration solutions and customization to connect the new systems with legacy applications remained poorly documented, thereby increasing documentation debt. This left the site dependent on a small pool of system consultants who understood the solutions (people debt). Also, infrastructure debt remained a problem since the old system components and related hardware had to remain in place.

The aggregate impact of design moves to implement the two systems (Tables 3-4) is depicted in Figure 2. The figure presents EngineShop's overall option-debt ratios during the staggered replacement process, narrating a journey with an unintended outcome: moving from an option-constrained to a debt-constrained state.

Table 4. Design Moves in the SAP System Implementation

Design move	Episode and strategic intent	Sociotechnical gaps (Social inertia)	Design actions	State of design capital	Impact of design move
S1A	1: Guarantee local productivity while implementing an SAP system with global strategic goals	Building-system task incompatible with DG work-system structure	Blueprint a locally customized SAP system implementation	Option constrained	Increased debt
S1B		Building-system structure inadequate for the implementation task, because of incompatibility with DG work-system structure	Develop a customized document-mediation module; omit other key functionality from the implementation	Option constrained	Increased debt
S1C		Customizations balancing the building system and enabling SAP system implementation	Replace DG's ERP functionality with SAP functions	Low quality	Abandoned options, increased debt
S2A	2: Create conditions for global SAP system integration and maintain acceptable levels of local productivity	Incompatibility of the SAP work system's technology with the structure, actors, and task	Develop quick fixes with Excel to enable process traceability	Low quality	Created options, increased debt
S2B		SAP work-system technology incompatible with the structure and actors	Develop a custom manufacturing-execution module	Low quality	Created options, increased debt
S2C		SAP work-system technology incompatible with the structure and actors	Develop a custom item-traceability module	Debt constrained	Created options, increased debt

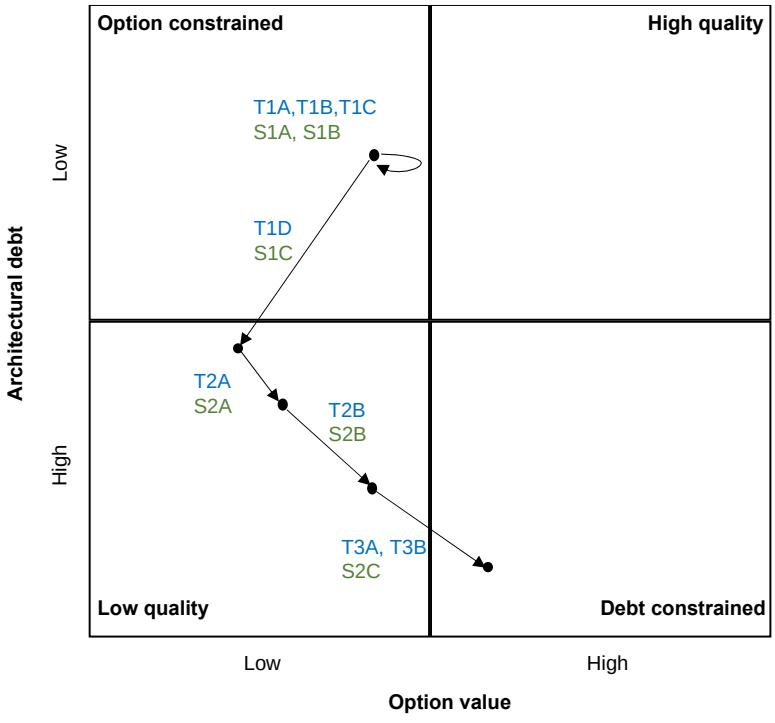

Figure 2. The Effect of Design Moves in the Two Implementations on Debt and Option Levels

Both implementations exhibited a pattern in which an unplanned and reactive string of technical changes over several years dramatically increased architectural debt. Initial customization to restore legacy-afforded digital options built a path dependency whereby the further pursuit of additional options came through additional customization. These together dragged the site into deeper unintended architectural debt.

Theoretical Synthesis

Our analysis suggests that the systemic relationships among social inertia, technical debt, digital options, and legacy system discontinuance are complex, path-dependent, and dynamic—more so when the replacement operations are staggered, involve multiple systems, and touch multiple sites with significant dependencies. The dynamic interactions between elements produce feedback loops that push system architectures toward higher technical debt while inhibiting the realization of digital options. This points toward examining the phenomenon via a systems perspective that assumes reciprocal and temporal causal relationships among components (e.g., goals, decisions, IT systems) of a holistic system (e.g., an organization's overall sociotechnical system) (Sterman, 2001). Further, most prior studies of system implementation and technical debt have approached the replacement problem as a single-level issue, yet the need to recognize the presence of both centralized global control *and* heterogeneous local operations calls for a multilevel perspective involving emergent and constraining interactions between lower and higher levels (Kozlowski & Klein, 2000). Adopting such a perspective helps capture the complex, emergent system behaviors that follow local system replacement and how they interact with the global responses.

The Dynamics of Legacy Systems' Replacement

Encouraged by our analysis identifying paradoxical and nonintended outcomes associated with local system replacement, we decided to apply system dynamics (SD) to synthesize the key relationships among relevant elements into an SD model of legacy system replacement (Baker & Singh, 2019; Fang et al., 2018; Martinez-Moyano et al., 2014). SD is a modeling tool grounded on the systems perspective that enables one to model holistic system structures with circular and delayed causality between different components (Sterman, 2001). We thus engaged in an abductive process appropriate to understand paradoxical findings (Sætre & Van de Ven, 2021), which called us to reflect upon our empirical findings against the backdrop of introduced theoretical concepts. Drawing on the PSIC and design-moves analysis, we integrated the key elements into a multilevel sociotechnical framework that revealed primary causal links between them (e.g., between customizing COTS systems and local technical debt). Per Fang et al. (2018) and Martinez-Moyano et al. (2014), we determined the polarity (negative vs. positive effect) of each causal link in line with our assessment of the direction of causal effect (e.g., customization has a positive effect on local technical debt). In assessing polarity, we triangulated among our empirical findings, previous research, and logical reasoning.

Next, we articulated the main feedback loops by integrating sets of chained (recursive) causal effects (see Fang et al., 2018). Then we looked at the net effect of each feedback loop and denoted each loop as either a reinforcing or a balancing one. A reinforcing loop (with positive net effect, indicated by "R") features either an even number of negative links or no links that are self-sustaining. In contrast, a balancing loop (with a negative net effect, denoted by "B") has an odd number of negative links so that it counterweighs the reinforcing loops to which it is connected. Each loop was given a descriptive label based on its expected main effect (Martinez-Moyano et al., 2014). For example, the Global-Option-Pursuit Loop R1 in Figure 4 expresses a reinforcing loop of how digital options are pursued at a global level. In line with the multilevel perspective of the PSIC analysis, we positioned the loops across the focal levels of analysis: the local site and global organization. The multilevel nature of the phenomena observed and our case findings led us to conceptualize both technical debt and digital options as manifested at both a local and global level in the organization. In line with the PSIC approach, we recognized the effects originating from the external environment.

The resulting SD model illustrates how social inertia, technical debt, and digital options interact and influence (and are influenced by) one another during system replacement at each level of analysis. It explains the persistence of legacy systems during the replacement process and the consequent accumulation of technical debt. Next, we elaborate on the model's organization and logic. Drawing on this theoretical synthesis, we next formulate four propositions focused on main causal loops, which we expect to hold more generally across contexts. The propositions communicate the theoretical logic of how technical and social inertia affect legacy system discontinuance—the focal phenomenon of our study.

We begin our presentation of the SD model by examining the dynamics of pursuing global-level options related to agility and reducing global technical debt. Then, we complete our outline by connecting these dynamics to those of the local-level replacement, which accumulated technical debt locally as a result of technical and social inertia faced during system replacement. The "full" SD model covers both levels and shows how local inertia ultimately struck back and shaped EngineGroup's global debt and options. Appendix B outlines the model's key constructs and the logic of their causal links.

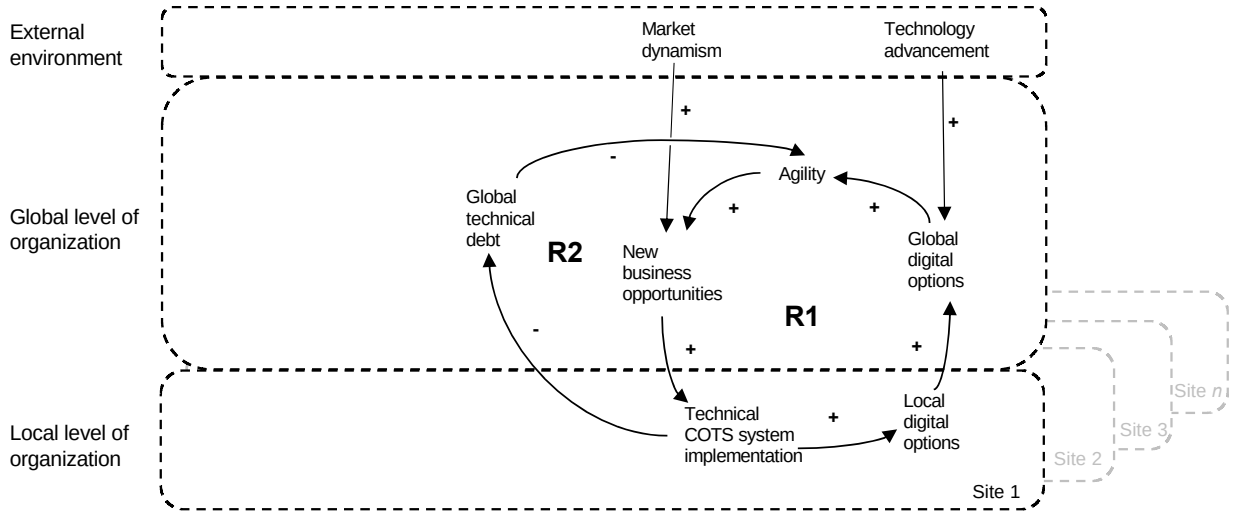

Figure 3. Global Pursuit of Agility via a Common COTS Solution

Global COTS Implementation Dynamics

Market dynamism driven by globalization and turbulent competition creates new business opportunities for global organizations like EngineGroup. At the same time, technological advancements offer new IT solutions, such as COTS systems, that help them seize those opportunities. For example, the standardization of data and processes by means of COTS solutions grants global managers greater control over local sites (Rolland & Monteiro, 2002). This allows more flexible and efficient coordination of global product engineering and manufacturing. These IT-enabled coordination mechanisms extend the digital options globally for increased agility (Overby et al., 2006; Rolland et al., 2018; Sambamurthy et al., 2003). Creation of such options was the main driver triggering the sociotechnical change narrated in our case. EngineGroup pursued these options through a direct top-down effect (Kozlowski & Klein, 2000) by mandating a common COTS solution for all local sites, where the solution would generate similar options locally. Because the local sites of such global organizations have distinct characteristics, histories, and capabilities, global digital options can be expected to emerge via a bottom-up "compilation" process (Kozlowski & Klein, 2000).[2] This assumes that the *realized* local options can be brought together across local sites (e.g., Finland's and Estonia's using shared product-data structures) and then configured globally in a process that is inherently complex and discontinuous. The salience of the available options at each site depends on the site's unique characteristics and position in the larger organization. In sum, EngineGroup's global pursuit of new opportunities via digital options entailed a top-down mandate to implement COTS solutions at each local site (EngineShop and all others). From an SD perspective, this endeavor should produce the hoped-for reinforcing cycle of wider digital options enabling agility: the *global-option-pursuit loop (R1)* in Figure 3. Another motive for implementing a common COTS solution was EngineGroup's need to resolve its globally accumulating technical debt created by heterogeneous local systems. Harmonizing the global system architecture via a common COTS solution creates the second self-sustaining cycle, the *global-debt-elimination loop (R2)*.

Local COTS Implementation Dynamics

Without the technical and social inertia inherent to local system replacements, R1 and R2 would yield agility and reduce technical debt ad infinitum through recurrent (frictionless) COTS solution implementation. In complex reality, however, local, heterogeneous legacy systems are rife with technical inertia originating from multiple types of technical debt. Debt-infested legacy systems impose heightened maintenance obligations and hinder local operations (Brown et al., 2010; Holvitie et al., 2018). If unaddressed, these issues can eventually weigh down global operations as well (Banker et al., 2020; Tom et al., 2013).

[2] While we do not have detailed data on system implementations at local sites other than EngineShop, this is a logical deduction.

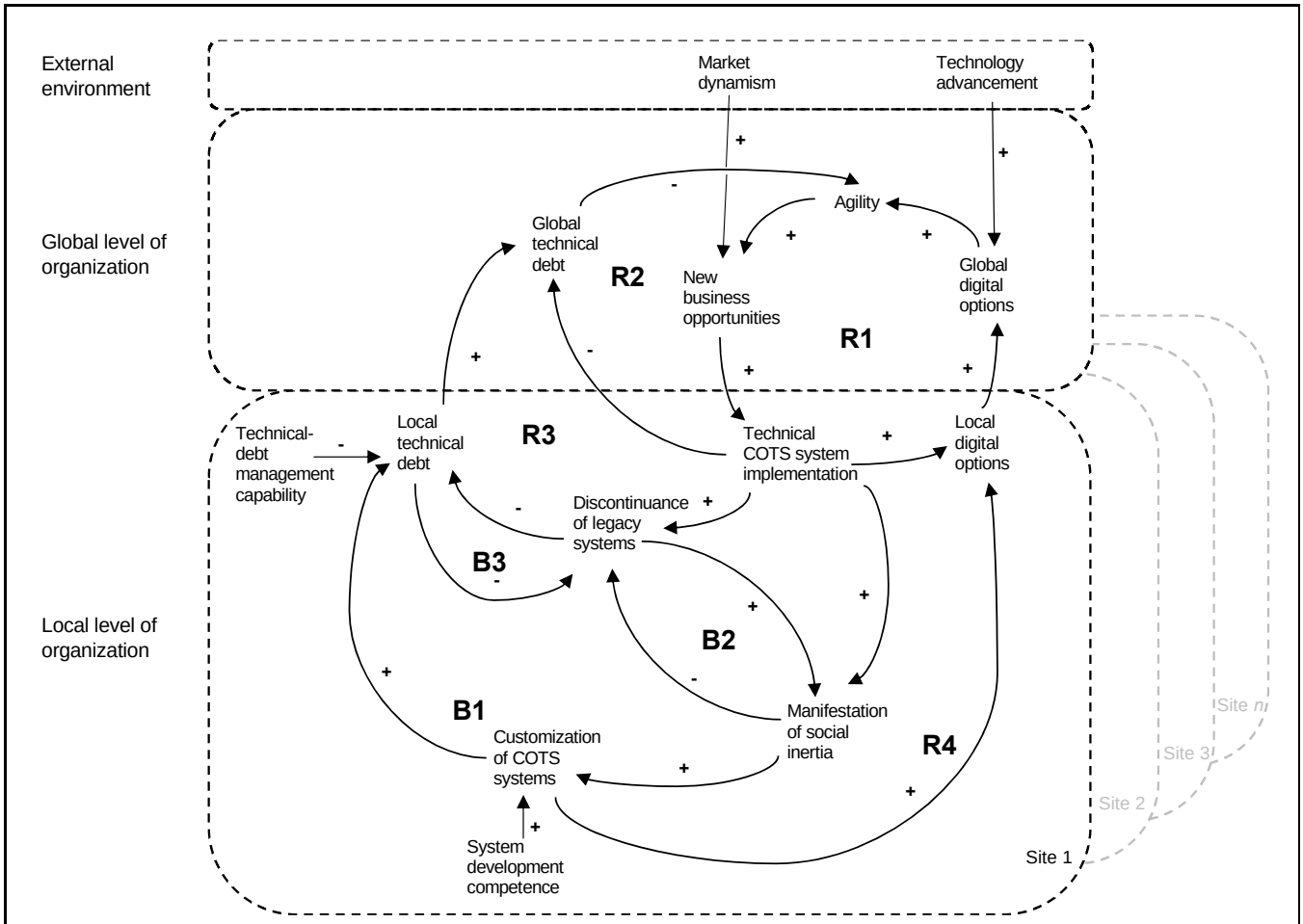

Figure 4. A Multilevel SD Model of Technical Debt and Legacy System Replacement

In organizations such as EngineGroup, local-level technical debt emerges at a global level via a compilation effect (Kozlowski & Klein, 2000), similar to realizing digital options: local sites pile up varying levels of multiple types of technical debt, with the extent of global debt then determined through a complex configuration of debts across sites. In such a setting, achieving global agility would require addressing these forms of locally accumulated debt while implementing the common COTS solution.

As Figure 4 depicts, implementing a common COTS solution necessitates withdrawing the local site's legacy systems (Furneaux & Wade, 2011). Prior literature (Rolland et al., 2018) and the EngineShop case indicate that the discontinuance of legacy systems mired in technical debt allows local actors to evaluate and resolve the debt by reducing technology gaps, removing system dependencies, and shifting maintenance obligations to COTS systems' providers. If such a local decrease in technical debt is carried out diligently across local sites (as was initially intended at EngineShop), this effort would alleviate global technical debt through a compilation effect (Kozlowski & Klein, 2000). From the SD standpoint, such decreases in global technical debt would also increase global agility. Generally, we expect the replacement of legacy systems to accordingly create a self-reinforcing cycle that spans the local and global levels of the organization. This is the *local-debt-elimination loop (R3)* in Figure 4. Hence, we offer:

Proposition 1: *In global, multi-site organizations, the discontinuance of local legacy systems increases the global organization's agility by resolving technical debt locally and globally.*

Together, three self-reinforcing cycles (R1-R3) enhance global agility. But their effects are curtailed by the local social inertia that these cycles trigger. Replacing a local legacy system with a common COTS solution pushes the

social system out of equilibrium and initiates significant social change: actors, tasks, and structures need to be adjusted to the new system's use and the global imperatives that accompany it. Also, the local legacy system has become entrenched in the local site behaviorally (indwelled), politically (legitimized), cognitively (learned), materially (resourced), and economically (routinized) (Mehrizi et al., 2019). These systems provide locally relevant options that users and managers are reluctant to let go of. Hence, they will resist the change that comes with a COTS solution; therefore, both COTS implementation and consequent legacy system discontinuance have to overcome significant social inertia (Berente et al., 2019; Holland et al., 1999) because they tear down local institutionalized models of operation incongruent with the proposed COTS solution.

The key consequence of social inertia is increased pressure to customize the common COTS solution to local needs. Customization may generate site-specific options that are unavailable with the common COTS solution, thereby creating a comparative advantage for the local site if its strategy is geared to differentiation and locally unique high-value products (Davenport, 1998). In contrast, EngineGroup's chosen strategy prioritized global flexibility and coordination, demanding harmonization of systems across all sites. Our case analysis suggests that given the contradictory demands across the global-local divide alongside local managers' bounded rationality and cognitive entrenchment (Arvidsson et al., 2014), COTS implementations share a tendency to seek to maintain the local status quo. Fear of growing social inertia in combination with the presence of inherited local development competencies led EngineShop to customize the proposed COTS solution for a better fit with its current operations environment. This, however, accumulated technical debt by introducing nonstandard and latent dependencies with additional maintenance obligations and leaving solutions poorly documented and weakly understood. Local accumulation of technical debt at EngineShop and other sites contributed to growth in technical debt globally for EngineGroup, making it difficult to reap the rewards of coordination options afforded by the shared solution. These dynamics suggest that local customizations accumulate technical debt locally creating a balancing *customization-debt loop (B1)* that reduces the positive effect of the three reinforcing loops (R1-R3). Hence, we posit:

Proposition 2: *Implementing a globally shared COTS system at local sites triggers local social inertia. The inertia increases the local sites' predisposition to customizing the COTS system producing increased technical debt both locally and globally.*

It should be noted that later system customizations designed to provide lacking functionalities and render local systems compatible with other sites' COTS implementations helped EngineShop obtain the options needed for collaborating globally (though entailing additional technical debt). This helped realize the initially desired global options and thereby permitted EngineGroup to gain global agility as represented in the *customization-options loop (R4)*.

The second consequence of local social inertia is the heightened "stickiness" of legacy systems. At EngineShop, parts of the legacy system were never discontinued because their functionality allowed the site to continue operating effectively, as called for by the global management. Generally, local sites can anticipate high social inertia during a common COTS implementation project when critical legacy system components have become woven into the organization's social fabric with self-reinforcing mechanisms (Mehrizi et al., 2019). Fully withdrawing these components also introduces high technical risks in the form of missing or low-quality data, technical errors, or difficulties in migrating operations to the new system, which in extreme cases leads to significant financial losses (Turban et al., 2008, pp. 316-317), even bankruptcy (Scott, 1999). Leaving legacy systems in place, even just partly, helps alleviate such concerns, but this risk-averse action yields nonstandard dependencies and technology gaps, adding to technical debt. A balancing *social-discontinuance-inertia loop (B2)* is generated, counteracting the effects of the local-debt-elimination loop (R3). Hence, we posit:

Proposition 3: *Discontinuing an entrenched local legacy system triggers local social inertia. The inertia decreases the unit's predisposition toward fully discontinuing legacy systems leading to (latent) increases in technical debt both locally and globally.*

These two local responses to social inertia created by the new COTS system—customization of COTS solutions and leaving legacy system components in place—increase technical inertia by adding technical debt locally, which over time reduces the organization's global agility. While the global digital options pursued through the COTS solution may become available over time, the organization's ability to realize these options remains unknown and limited throughout the implementation process. Local sites are likely to continue customizing the COTS solution to overcome social inertia. They may fail to discontinue parts of their legacy systems, given the presence of both social and technical inertia and the lure of digital options such systems afford locally. The extent to which local COTS implementations enable global agility and reduce technical debt depends on the local sites' abilities to manage their technical debt (Ramasubbu & Kemerer, 2016). Successfully

managing such debt requires adopting systematic global approaches and processes (Ramasubbu & Kemerer, 2019). Regrettably, debt often rises in an unconstrained and hidden manner because most organizations are weakly prepared to manage it both locally and globally. This was the case at EngineShop, where few conscious efforts were made to manage technical debt over the years of the study.

Escalating technical inertia due to the innocuous-seeming accumulation of technical debt is likely to trap the local site in a Catch-22 situation. Though the site can, in principle, resolve its technical debt by removing the remaining components of legacy systems, the accrued debt resulting from ongoing COTS-system customization and the legacy components' entrenched nature renders the legacy systems technically and socially inert. Their decommissioning turns into a dauntingly complex task and a leap of faith. The constant accumulation of technical debt increases technical inertia such that the "temporary" architectural solutions integrating the COTS solution with the legacy components become permanent. This can be represented as a negative balancing loop, here labeled the *technical-discontinuance-inertia loop (B3).* The loop counteracts the positive effect of the local-debt-elimination loop (R3) and reduces the local site's ability to settle its technical debt. We thus posit the following:

Proposition 4: *Accumulating technical debt reduces a local site's likelihood of discontinuing its legacy systems. By corollary, this reduces the site's predisposition toward reducing its accumulated technical debt.*

In sum, the SD model (Figure 4) addresses our first research question. It synthesizes how social inertia, technical inertia (expressed as technical debt), and digital options interact dynamically when firms replace legacy systems with COTS systems in a global, multi-site setting. The relationships between the components can produce alternative outcomes (sociotechnical equilibria), depending on the choices made during implementation and the level and variation of each factor (technical/social inertia) during the process. With regard to the second research question—i.e., how and under what conditions does the state of being "caught between" emerge and stabilize during this process—the model and the propositions identify two critical pathways, which increase the likelihood of a legacy system replacement process to result in a prolonged caught-between situation. The likelihood increases significantly when the local site accrues technical debt by (1) customizing the COTS system to fit it into the (sociotechnical) legacy environment (task, technology, structure, people), and/or (2) leaving its legacy system or parts thereof in place (temporarily or indefinitely) while implementing the COTS system. These choices are more likely in the presence of strong social inertia (against COTS logic and/or legacy system discontinuance created by deep entrenchment), high local system-development competence, and a low (or nonexisting) capability to manage technical debt. Alternatively, if an organization replaces the legacy system with a vanilla COTS system (void or with minimal customization) following a "big-bang" approach (full replacement at once), the likelihood of getting "caught between" is lower. However, the choice comes with different risk profile in that it increases the technical risks of poorly functioning systems and heightened social inertia and thus demands management attention.

Discussion and Conclusion

Theoretical Contributions

Our findings contribute to research at the intersection of legacy system discontinuance and COTS systems' implementation. We are among the first to break apart the dynamic interactions between technical inertia, social inertia, and digital options during the replacement process that follows COTS system implementation. Moreover, the study contributes to a dynamic account of local and global drivers of technical debt and digital options by integrating their interactions into a multilevel sociotechnical framework (Figure 4). Finally, the results offer new insights into the IT governance literature's debates around the benefits of centralization vs. decentralization.

First, our findings extend the work on legacy system discontinuance (Mehrizi et al., 2019, 2022), which has suggested that, in addition to deploying "ceasing mechanisms" that discredit and deroutinize legacy systems, organizations must develop mechanisms that "intensify" the legacy system's use so that the old and the new system can jointly support the business for some time and guarantee smooth migration of data and functions to the new system. However, our findings highlight a dark side to such intensification: leaving portions of the legacy system in place by relegitimizing them encourages technical debt. If this debt is not settled, the organization may innocently pursue self-reinforcing mechanisms in a vicious circle, which makes the legacy system increasingly "sticky" and impenetrable. Quantitative discontinuance studies have identified the salience of such technical inertia influencing managers' discontinuance decisions (Furneaux & Wade, 2011, 2017). Our process analysis expands on this insight by revealing the sources and dynamics of these inertial factors, contributing to the emerging understanding of IS discontinuance as part of the system replacement process (Soliman & Rinta-Kahila, 2020).

Second, our study pinpoints some prominent questions for COTS implementation scholarship (Davenport, 1998). Whether to impose a vanilla system or, instead, customize it for local needs in a multi-site, multilevel context remains one of the key questions in this domain. While we cannot offer a definitive answer, the benefit of hindsight suggests that implementation decisions aimed at reducing social inertia contribute to technical debt to such an extent that a socially risky vanilla implementation may turn out to be a better alternative in the long term. We expect this to be the case when the social and technical implementation risks can be anticipated and managed properly and when adequate resources are available to push the vanilla implementation to completion. Notorious cases of failed vanilla COTS implementations (e.g., Hershey and Whirlpool; see Turban et al., 2008) reveal a pattern of decisive short-term damage in which *both* social and technical risks materialize. But they also point to significant mismanagement of such risks before and during the process. In contrast, our study heeds the need to balance short-term implementation risks with the often-overlooked long-term risk of accruing technical debt. Focusing on mitigating short-term social inertia only may increase long-term technical inertia, thereby ultimately hampering the realization of COTS systems' digital options.

Third, the proposed SD model is novel in that it applies dynamic digital-option/debt-ratio analysis (Woodard et al., 2013) to legacy system replacement in a multilevel sociotechnical setting. The analysis expands upon nascent sociotechnical explanations of the origins of debt/option ratio dynamics found in IS literature (Rolland et al., 2018; Rolland & Lyytinen, 2021) by recognizing the mutual dependencies of technical and social inertia and how they shape these ratios at the local and global level. The results shed light on critical trade-offs that managers face in wrestling with such ratios: if global directors impose strict demands that back local sites into one approach (e.g., by mandating a vanilla system and dictating that productivity not decline), the implementation is likely to have unintended, ironic outcomes that solidify into an "in-between" state precluding realization of digital options. Those analyzing how digital options are created and realized with COTS systems are advised to recognize the multilevel dynamics that emerge from the presence of multiple, heterogeneous operation logics across levels. This highlights the criticality of legacy system discontinuance: it is an important but often overlooked element driving the accumulation of technical debt.

Finally, our study provides novel contributions to IT governance literature. Several approaches have been proposed for federated, multisite organizations in terms of distributing decision authority over IT activities between local and central authorities (Sambamurthy & Zmud, 1999; Weill & Ross, 2005). Decentralized governance modes where local units are allowed to run customized IT infrastructures tend to accrue technical debt from the global governance perspective. Therefore, shifting to central governance in systems and architectures can be expected to reduce technical debt. However, such transitions bear a risk of growing social inertia in local system implementation, which can result in a further accumulation of local technical debt. In a federal governance mode, like that of EngineGroup, matrix-based reporting seeks to find a balance between local and global IT authority. But the federal solution also increases complexity that can overwhelm managers and limit their decision-making effectiveness (Weill & Ross 2005, p. 33), preventing the resolution of technical debt both locally and globally.

Management Implications

From a managerial angle, EngineShop's case sheds light on dilemmas that managers encounter when tackling complex, long-term decisions around sociotechnical change. The propositions stemming from our model can help managers understand the trade-offs they face during legacy system replacement. These highlight the importance of managing system architectures in conjunction with tackling social inertia. The business case for legacy system replacement needs to address the social dimension of change and its inherent connection to accumulating technical debt. Managers must be aware of the constraints that the incumbent business model embodied by a legacy system imposes (Holland et al., 1999). They need to decide how much, if at all, the current operation logic should dictate the new implementation's scope and goals. Also, managers must respect the strong hidden path dependencies that grow from seemingly innocuous choices during COTS implementation. Our SD model should help managers balance trade-offs between such short- and long-term risks by exposing early implementation decisions' inevitable path-creating consequences at the global level (e.g., for the availability of digital options) and the local level (accrual of technical debt). By illuminating these dynamics, the SD model provides a sensemaking framework for analyzing potential long-term implications of implementation choices and building business cases that can help resolve technical debt.

As noted above, our analysis does not suggest that heavy-handed process reengineering around a common COTS system is always the best course of action. The answer depends on surrounding conditions (Davenport, 1998). The choice should be made with conscious anticipation, rather than a reactive response to chaotic implementation dynamics. Our model invites managers to look beyond the seeming comfort of sticking with legacy systems and to examine them critically, extending their analysis horizon to the broader sociotechnical system and related strategic objectives. This does necessitate an unpleasant balancing act between the needs of the local sites and the needs stemming from the organization's global strategic thrusts and how they are linked to short- and long-term risks.

Concluding Remarks

We acknowledge several limitations. Most events chronicled were historical and captured via informants' retrospective accounts. Informants may not have recalled all events shaping the process and its outcomes, and some might exhibit personal biases in their recall of the events and in interpreting them. We invested significant effort in offsetting such source bias (Huber & Power, 1985; Klein & Myers, 1999), sampling all relevant stakeholder groups (which offered diverse perspectives) and triangulating the events identified against archival documents. Still, tracing a complex replacement process in real time could have painted a more faithful picture of the change. Also, the complex matrix structure of the EngineGroup organization and the system architecture at EngineShop made distinguishing and interpreting some interactions between project events and contextual factors challenging; therefore, our process analysis may have missed some salient considerations. To guarantee a sound understanding of the environment, we consulted the site's managers and other personnel multiple times.

We further note that in federated organizations, globally imposed implementation policies will inevitably be influenced by global social inertia stemming from cognitive bias and political dynamics. Global decision makers, who are detached from different local implementation contexts, may hold myopic views of what the global system solution can provide strategically and what such implementation would entail (Avison et al., 2006). Moreover, local sites have varying degrees of political power, and each will seek to influence global decision-making to advance their own interests (Berente et al., 2019; Markus, 1983). Such global social factors are not addressed in our model because the case study drew on a data corpus collected mostly on a single local site. Future research could increase the generalizability and comprehensiveness of the model and propositions via a multicase design that encompasses more data at the global level.

Also, the study's temporal scope was limited to the time span explicitly covered by the dataset. We cannot be sure that the accumulation of technical debt observed will have negative impacts on the organization later. EngineShop may have gained a competitive edge offsetting the future costs of having to manage the debt, and it might resort to recent technical solutions such as microservice architectures (Bozan et al., 2021) to help resolve the challenges related to the accrued debt. The extent to which the organization can and will exploit such opportunities requires further analysis in the future. Further, due to our focus on a single site, we had limited access to the IT governance processes that bind both global managers and local-level stakeholders at various sites. The challenge of improving multilevel IT governance processes to optimize the options-debt ratio remains an important future research area.

Several additional research opportunities arise from the study. Further study could offer a finer-grained longitudinal treatment of the accumulation of various types of technical debt. It should delve deeply into reciprocal interactions—how individual debt types affect each other, the availability of digital options, and architectural choices. Since economic and operational risks render legacy systems notoriously hard to replace, our study is among the first to recognize the full gamut of sociotechnical challenges that accompany replacement through treating it as a multilevel, multi-site, and path-dependent process involving both technical *and* social change with their associated inertia. As organizational systems grow larger and the range of digital assets shaping business solutions widens, such difficulties of replacement are likely to grow still more complex and challenging, calling for increased research into this overlooked yet important topic.

Acknowledgments

We are grateful to the case organization for generous access and support. We thank the SE, AE, and three anonymous reviewers for their highly constructive comments that significantly improved the manuscript. We are grateful for the valuable feedback from the participants of The University of Auckland's Qualitative Paper Development Workshop 2019, the Aalto University ISS research seminar, and ICIS 2018. We wish to especially thank Ilan Oshri, Chee-Wee Tan, Elena Karahanna, Matti Mäntymäki, Mark Bremhorst, and Malshika Dias for their insightful comments on the early versions of this manuscript. Our special thanks go to Ning Nan (UBC) for her generous help in validating the system-dynamics models. As always, all remaining errors are ours. This research was partially supported by Finland's Foundation for Economic Education (Liikesivistysrahasto), Grant: [180292]. We remain grateful for the foundation's generous support.

References

Aanestad, M., Grisot, M., Hanseth, O., & Vassilakopoulou, P. (2017). Information Infrastructures and the Challenge of the Installed Base. In M. Aanestad, M. Grisot, O. Hanseth, & P. Vassilakopoulou (Eds.), *Information health care within european infrastructures: Working with the installed base* (pp. 25-33). Springer.

Adolph, W. S. (1996). Cash cow in the tar pit: Reengineering a legacy system. *IEEE Software*, *13*(3), 41-47. https://doi.org/10.1109/52.493019

Advanced. (2021). *Mainframe modernization business barometer report*. https://modernsystems.oneadvanced.com/globalassets/modern-systems-assets/resources/reports/advanced_mainframe_report_2021.pdf

Almonaies, A. A., Cordy, J. R., & Dean, T. R. (2010). Legacy system evolution towards service-oriented architecture. *International Workshop on SOA Migration and Evolution*, 53-62. https://doi.org/10.1.1.212.5383

Alves, N. S. R., Mendes, T. S., De Mendonça, M. G., Spinola, R.

O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, *70*, 100-121. https://doi.org/10.1016/j.infsof.2015.10.008

Arvidsson, V., Holmström, J., & Lyytinen, K. (2014). Information systems use as strategy practice: A multi-dimensional view of strategic information system implementation and use. *Journal of Strategic Information Systems*, *23*(1), 45-61. https://doi.org/10.1016/j.jsis.2014.01.004

Avison, D., Gregor, S., & Wilson, D. (2006). Managerial IT Unconsciousness. *Communications of the ACM*, *49*(7), 88-93. https://doi.org/10.1145/1139922.1139923

Baker, J., & Singh, H. (2019). The roots of misalignment: Insights on strategy implementation from a system dynamics perspective. *Journal of Strategic Information Systems*. https://doi.org/10.1016/j.jsis.2019.101576

Banker, R., Liang, Y., & Ramasubbu, N. (2020). Technical debt and firm performance. *Management Science*. https://doi.org/10.1287/mnsc.2019.3542

Berente, N., Gal, U., & Yoo, Y. (2010). Dressage, control, and enterprise systems: The case of NASA's Full Cost initiative. *European Journal of Information Systems*, *19*(1), 21-34. https://doi.org/10.1057/ejis.2009.47

Berente, N., Lyytinen, K., Yoo, Y., & King, J. L. (2016). Routines as shock absorbers during organizational transformation: integration, control, and NASA's enterprise information system. *Organization Science*, *27*(3), 551-572. https://doi.org/10.1287/orsc.2016.1046

Berente, N., Lyytinen, K., Yoo, Y., & Maurer, C. (2019). Institutional logics and pluralistic responses to enterprise system implementation: A qualitative meta-analysis. *MIS Quarterly*, *43*(3), 873-902. https://doi.org/10.25300/MISQ/2019/14214

Berente, N., & Yoo, Y. (2012). Institutional contradictions and loose coupling: Postimplementation of NASA's enterprise information system. *Information Systems Research*, *23*(2), 376-396. https://doi.org/10.1287/isre.1110.0373

Besker, T., Martini, A., & Bosch, J. (2018). Managing architectural technical debt: A unified model and systematic literature review. *Journal of Systems and Software*, *135*, 1-16. https://doi.org/10.1016/j.jss.2017.09.025

Besson, P., & Rowe, F. (2012). Strategizing information systems-enabled organizational transformation: A transdisciplinary review and new directions. *Journal of Strategic Information Systems*, *21*(2), 103-124. https://doi.org/10.1016/j.jsis.2012.05.001

Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy information systems: Issues and directions. *IEEE Software*, *16*(5), 103-111. https://doi.org/10.1109/52.795108

Bozan, K., Lyytinen, K., & Rose, G. M. (2021). How to transition incrementally to microservice architecture. *Communications of the ACM*, *64*(1), 79-85. https://doi.org/10.1145/3378064

Brown, N., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., Zazworka, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., & Nord, R. (2010). Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. https://doi.org/10.1145/1882362.1882373

CRN. (2012). *How to kill off legacy systems*. https://www.crn.com/blogs-op-ed/channel-voices/240000330/how-to-kill-off-legacy-systems.htm

Cunningham, W. (1992). The WyCash portfolio management system. In *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications* (pp. 29-30).

Davenport, T. H. (1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, *July-August*, 121-132. https://hbr.org/1998/07/putting-the-enterprise-into-the-enterprise-system

Deloitte. (2013). When companies become prisoners of legacy systems. *Wall Street Journal: CIO Journal* https://deloitte.wsj.com/articles/when-companies-become-prisoners-of-legacy-systems-1380600092?tesla=y

Deloitte. (2021). *What is blocking your progress towards increased business agility?* https://www2.deloitte.com/nl/nl/pages/human-capital/articles/what-is-blocking-you-progress-towards-increased-business-agility.html

Fang, Y., Lim, K. H., Qian, Y., & Feng, B. (2018). System dynamics modeling for information systems research: Theory development and practical application. *MIS Quarterly*, *42*(4), 1303-1329. https://doi.org/10.25300/MISQ/2018/12749

Furneaux, B., & Wade, M. (2011). An exploration of organizational level information systems discontinuance intentions. *MIS Quarterly*, *35*(3), 573-598. https://doi.org/10.2307/23042797

Furneaux, B., & Wade, M. (2017). Impediments to information systems replacement: A calculus of discontinuance. *Journal of Management Information Systems*, *34*(3), 902-932. https://doi.org/10.1080/07421222.2017.1373013

Fürstenau, D., Baiyere, A., & Kliewer, N. (2019). A dynamic model of embeddedness in digital infrastructures. *Information Systems Research*, *30*(4), 1319-1342. https://doi.org/10.1287/isre.2019.0864

Gangadharan, G. R., Kuiper, E. J., Janssen, M., & Luttighuis, P. O. (2013). IT innovation squeeze: Propositions and a methodology for deciding to continue or decommission legacy systems. *IFIP Advances in Information and Communication Technology*, *402*, 481-494. https://doi.org/10.1007/978-3-642-38862-0_30

Gómez, G. (2020). The biggest reason companies are stuck in the past—and how to escape. *Forbes*. https://www.forbes.com/sites/forbestechcouncil/2020/02/25/the-biggest-reason-companies-are-stuck-in-the-past-and-how-to-escape/?sh=75b81709199d

Hemon-Laurens, A. (2016). Banks don't have to kill legacy systems to be more agile. *American Banker*. https://www.americanbanker.com/opinion/banks-dont-have-to-kill-legacy-systems-to-be-more-agile

Holland, C. P., Gibson, N., Light, B., & Kelly, S. (1999). A business perspective of legacy information systems. *Communications of the Association for Information Systems*, *2*, Article 7.

Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., Buchan, J., & Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, *96*, 141-160. https://doi.org/10.1016/j.infsof.2017.11.015

Huber, G. P., & Power, D. J. (1985). Retrospective reports of strategic-level managers: Guidelines for increasing their accuracy. *Strategic Management Journal*, *6*(2), 171-180.

Klein, H., & Myers, M. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, *23*(1), 67-93. https://doi.org/10.2307/249410

Kozlowski, S. W. J., & Klein, K. J. (2000). A multilevel approach to theory and research in organizations: Contextual, temporal, and emergent processes. In K. J. Klein & S. W. J. Kozlowski (Eds.), *Multilevel theory, research and methods in organizations: Foundations, extensions, and new directions* (pp. 3-90). Jossey-Bass.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software*, *29*(6), 18-21. https://doi.org/10.1109/MS.2012.167

Leavitt, H. J. (1964). Applied organisation change in industry: structural, technical and human approaches. In W. H. Cooper, H. J. Leavitt, & M. W. Shelly II (Eds.), *New perspectives in organization research*. Wiley, 1144-1170.

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *The Journal of Systems and Software, 101*, 193-220. https://doi.org/10.1016/j.marpolbul.2018.09.025

Luo, W., & Strong, D. M. (2004). A framework for evaluating difficulties in ERP implementation. *IEEE Transactions on Engineering Management*, *51*(3), 322-333. https://doi.org/10.5220/0002612504600465

Lyytinen, K., & Hirschheim, R. (1987). Information systems failures - a survey and classification of the empirical literature. *Oxford Surveys in Information Technology*, *4*, 257-309.

Lyytinen, K., & Newman, M. (2008). Explaining information systems change: A punctuated socio-technical change model. *European Journal of Information Systems*, *17*(6), 589-613. https://doi.org/10.1057/ejis.2008.50

Lyytinen, K., & Newman, M. (2015). A tale of two coalitions: Marginalising the users while successfully implementing an enterprise resource planning system. *Information Systems Journal*, *25*(2), 71-101. https://doi.org/10.1111/isj.12044

Lyytinen, K., Newman, M., & Al-Muharfi, A. R. A. (2009). Institutionalizing enterprise resource planning in the Saudi steel industry: A punctuated socio-technical analysis. *Journal of Information Technology*, *24*(4), 286-304. https://doi.org/10.1057/jit.2009.14

Mäki, N., Penttinen, E., & Rinta-Kahila, T. (2023). A domino effect: Interdependencies among different types of technical debt. In *Proceedings of the 56th Hawaii International Conference on System Sciences.* https://hdl.handle.net/10125/103356

Markus, M. L. (1983). Power, politics, and MIS implementation. *Communications of the ACM*, *26*(6), 430-444. https://doi.org/10.1145/358141.358148

Markus, M. L., & Robey, D. (1988). Information technology and organizational change: Causal structure in theory and research. *Management Science*, *34*(5), 583-598. https://doi.org/10.1287/mnsc.34.5.583

Martinez-Moyano, I. J., McCaffrey, D. P., & Oliva, R. (2014). Drift and adjustment in organizational rule compliance: Explaining the "regulatory pendulum" in financial markets. *Organization Science*, *25*(2), 321-338. https://doi.org/10.1287/orsc.2013. 0847

Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating architectural technical debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology*, *67*, 237-253. https://doi.org/10.1016/j.infsof.2015.07.005

Mehrizi, M. H. R., Modol, J. R., & Nezhad, M. Z. (2019). Intensifying to cease: Unpacking the process of information systems discontinuance. *MIS Quarterly*, *43*(1), 141-165. https://doi.org/10.25300/MISQ/2019/13717

Mehrizi, M. H. R., van den Hooff, B., & Yang, C. (2022). Breaking or keeping the habits: Exploring the role of legacy habits in the process of discontinuing organisational information systems. *Information Systems Journal*, *32*(1), 192-221. https://doi.org/10.1111/isj.12341

Newman, M., & Zhao, Y. (2008). The process of enterprise resource planning implementation and business process re-engineering: Tales from two Chinese small and medium-sized enterprises. *Information Systems Journal*, *18*(4), 405-426. https://doi.org/10.1111/j.1365-2575.2008.00305.x

Ning, J. Q., Engberts, A., & Kozaczynski, W. V. (1994). Automated support for legacy code understanding. *Communications of the ACM*, *37*(5), 50-57. https://doi.org/10.1145/175290.175295

Overby, E., Bharadwaj, A., & Sambamurthy, V. (2006). Enterprise agility and the enabling role of information technology. *European Journal of Information Systems*, *15*(2), 120-131. https://doi.org/10.1057/palgrave.ejis.3000600

Patton, M. Q. (1990). Qualitative evaluation and research methods. In. *Qualitative Evaluation and Research Methods* (2nd ed., pp. 169-1860. SAGE. https://doi.org/10.1002/nur.4770140111

Poon, P., & Wagner, C. (2001). Critical success factors revisited: Success and failure cases of information systems for senior executives. *Decision Support Systems*, *30*, 393-418.

Ramasubbu, N., & Kemerer, C. F. (2016). Technical debt and the reliability of enterprise software systems: A competing risks analysis. *Management Science*, *62*(5), 1487-1510. https://doi.org/10.2139/ssrn.2523483

Ramasubbu, N., & Kemerer, C. F. (2019). integrating technical debt management and software quality management processes: A normative framework and field tests. *IEEE Transactions on Software Engineering*, *45*(3), 285-300.

Rinta-Kahila, T. (2018). *Pulling the plug: The concept, process, and outcomes of organizational information system discontinuance* [Doctoral dissertation, Aalto University]. Aalto University Publication Series: Doctoral Dissertations, 208/2018. https://aaltodoc.aalto.fi/handle/123456789/34658.

Rios, N., Mendonça Neto, M. G. de, & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, *102*(May), 117-145. https://doi.org/10.1016/j.infsof.2018.05.010

Rivard, S., & Lapointe, L. (2012). Information technology implementers' responses to user resistance: Nature and effects. *MIS Quarterly*, *36*(3), 897-920. https://doi.org/10.2307/41703485

Rolland, K. H., & Lyytinen, K. (2021). Managing tensions between architectural debt and digital innovation: The case of a financial organization. In *Proceedings of the 54th Hawaii International Conference on System Sciences*. https://hdl.handle.net/10125/71427

Rolland, K. H., Mathiassen, L., & Rai, A. (2018). Managing digital platforms in user organizations: The interactions between digital options and digital debt. *Information Systems Research*, *29*(2), 419-443. https://doi.org/10.1287/isre.2018.0788

Rolland, K. H., & Monteiro, E. (2002). Balancing the local and the global in infrastructural information systems. *The Information Society*, *18*(2), 87-100. https://doi.org/10.1080/01972240290075020

Sætre, A. S., & Van de Ven, A. H. (2021). Generating theory by abduction. *Academy of Management Review*, *46*(4), 684-701. https://doi.org/10.5465/amr.2019.0233

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping agility through digital options: reconceptualizing the role of information technology in contemporary firms. *MIS Quarterly*, *27*(2), 237-263. https://doi.org/10.2307/30036530

Sambamurthy, V., & Zmud, R. W. (1999). Arrangements for information technology governance: A theory of multiple contingencies. *MIS Quarterly*, *23*(2), 261-290. https://doi.org/10.2307/249754

Sandborn, P. A., & Prabhakar, V. J. (2015). The forecasting and impact of the loss of critical human skills necessary for supporting legacy systems. *IEEE Transactions on Engineering Management*, *62*(3), 361-371. https://doi.org/10.1109/TEM.2015.2438820

Sarker, S., & Lee, A. S. (1999). IT-enabled organizational transformation: a case study of BPR failure at TELECO. *The Journal of Strategic Information Systems*, *8*(1), 83-103. https://doi.org/10.1016/S0963-8687(99)00015-3

Scott, J. E. (1999). The FoxMeyer Drugs' bankruptcy: Was it a Failure of ERP? *Proceedings of the Americas' Conference for Information Systems*.

Shang, S., & Seddon, P. B. (2002). Assessing and managing the benefits of enterprise systems: The business manager's perspective. *Information Systems Journal*, *12*(4), 271-299. https://doi.org/10.1046/j.1365-2575.2002.00132.x

Soliman, W., & Rinta-Kahila, T. (2020). Toward a refined conceptualization of IS discontinuance: Reflection on the past and a way forward. *Information and Management*, *57*(2), Article 103167. https://doi.org/10.1016/j.im.2019.05.002

Sterman, J. D. (2001). System dynamics modeling: tools for learning in a complex world. *California Management Review*, *43*(4), 8-25.

Strong, D. M., Johnson, S. a, Tulu, B., Trudel, J., Group, R. M., Volkoff, O., Pelletier, L. R., Bar-on, I., & Garber, L. (2014). A theory of organization-EHR affordance actualization. *Journal of the Association for Information Systems*, *15*(2), 53-85. https://doi.org/10.17705/1jais.00353

Study 1. (2008). *Anonymized title* [Master's thesis.] Aalto University.

Study 2. (2014). *Anonymized title* [Master's thesis.] Aalto University.

Study 3. (2010). *Anonymized title* [Thesis]. VAMK University of Applied Sciences.

Study 4. (2015). *Anonymized title* [Thesis]. VAMK University of Applied Sciences.

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *The Journal of Systems & Software*, *86*, 1498-1516. https://doi.org/10.1016/j.jss.2012.12.052

Turban, E., McLean, E., & Wetherbe, J. (2008). *Information technology for management: transforming organizations in the digital economy* (4th ed.). Wiley.

Van Maanen, J. E. (1988). *Tales of the field: On writing ethnography*. The University of Chicago Press.

Wagner, E. L. (2010). Understanding project survival in an ES environment: a sociomaterial practice perspective. *Journal of the Association for Information Systems*, *11*(5), 276-297. https://doi.org/10.17705/1jais.00227

Weill, P., & Ross, J. (2005). A matrixed approach to designing IT governance. *MIT Sloan Management Review*, *46*(2), 26-34.

Woodard, C. J., Ramasubbu, N., Tschang, F. T., & Sambamurthy, V. (2013). Design capital and design moves: The logic of digital business strategy. *MIS Quarterly*, *37*(2), 537-564. https://www.jstor.org/stable/43825922

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). SAGE.

About the Authors

Tapani Rinta-Kahila (Ph.D., information systems science, Aalto University) is a lecturer of business information systems at the University of Queensland Business School. His research addresses topics related to IS discontinuance, organizational implementation of artificial intelligence and automation, and the unintended consequences of IS. Rinta-Kahila's core expertise lies in qualitative in-depth case studies. His research has appeared in *Journal of the Association for Information Systems*, *European Journal of Information Systems*, *Information & Management*, and *MIS Quarterly Executive*.

Esko Penttinen (Ph.D., information systems science, Helsinki School of Economics), is an associate professor of information systems at Aalto University School of Business. Penttinen pursues a phenomenon-driven approach to information systems research, which has led him to work on a wide array of topics related to the development, implementation, and deployment of information systems. Penttinen's main expertise lies in the assimilation and economic implications of inter-organizational information systems, and he has focused on application areas such as financial systems, government reporting, and electronic invoicing. His research has appeared in *MIS Quarterly*, *Journal of the Association for Information Systems*, *Information Systems Journal*, *Journal of Information Technology*, and *MIS Quarterly Executive*.

Kalle Lyytinen (Ph.D., computer science, University of Jyväskylä; Dr. h.c. Umeå University, Copenhagen Business School, Lappeenranta University of Technology) is a Distinguished University Professor at Case Western Reserve University and a Distinguished Visiting Professor at Aalto University, Finland. He is among the top five IS scholars in terms of his h-index (96) and has the highest network centrality. He is an AIS Fellow (2004) and LEO Award recipient (2013), and the chair of IFIP WG 8.2 "Information Systems and Organizations" (1992-1996). He has published over 400 refereed articles, edited or written over 30 books or special issues, and won Best Paper awards from AoM, AIS/ICIS, and other societies. He has served as the EIC of the *Journal for the Association of Information Systems* (2005-2010), VP of Publications at AIS (2010-2013), and been either a SE or editor of all Basket-of-Eight IS journals as well as several leading organization theory and innovation journals. He currently conducts research on digital innovation, concerning its nature, dynamics, and organization; complex design work; requirements in large systems, and the emergence and growth of digital infrastructures.

Appendix A

The PSIC Analyses on the Implementations

Figures A1 and A2 depict the Teamcenter and SAP system's implementation, respectively, as narrated in the "Findings" section. The two uppermost levels feature events within the environment and the wider organizational context, such as decisions made by EngineGroup's global management. The next row presents local events related to the project, and the rows below depict the implementation in terms of sociotechnical building-system (B) and work-system (W) balances, respectively. The two-way arrows between sociotechnical components (A, T, S, and Te) denote their interrelationships, with those between components with a white background indicating a balanced relationship whereas dotted arrows connecting gray-colored components indicate gaps. For instance, a technology-task gap arose in the work system when an incumbent information system was found incapable of supporting employees' work tasks (W7 in both figures). Unidirectional arrows between sociotechnical systems indicate critical events, such as the legacy system's inadequate digital options triggering the system-replacement project (W1→B1). Bidirectional arrows between work systems and building systems indicate interaction between the systems—e.g., the legacy work system's complex structure rendering the building-system structure inadequate for the implementation task (B2↔W2 in Figure A1). The last row lists the design moves identified, with their implications for architectural debt and digital options. Bracketing along the horizontal axis shows the temporal progression of events across the episodes.

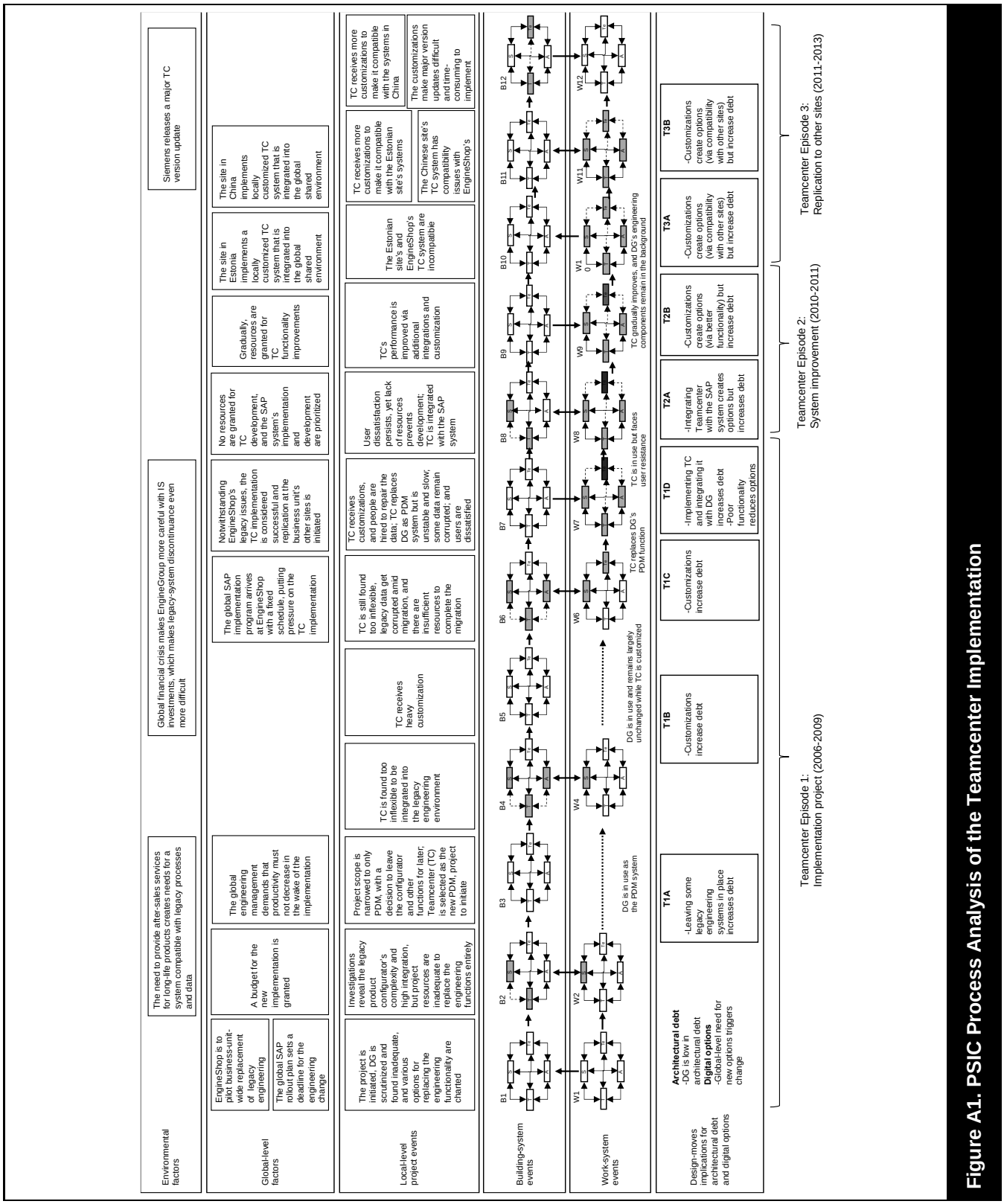

Figure A1. PSIC Process Analysis of the Teamcenter Implementation

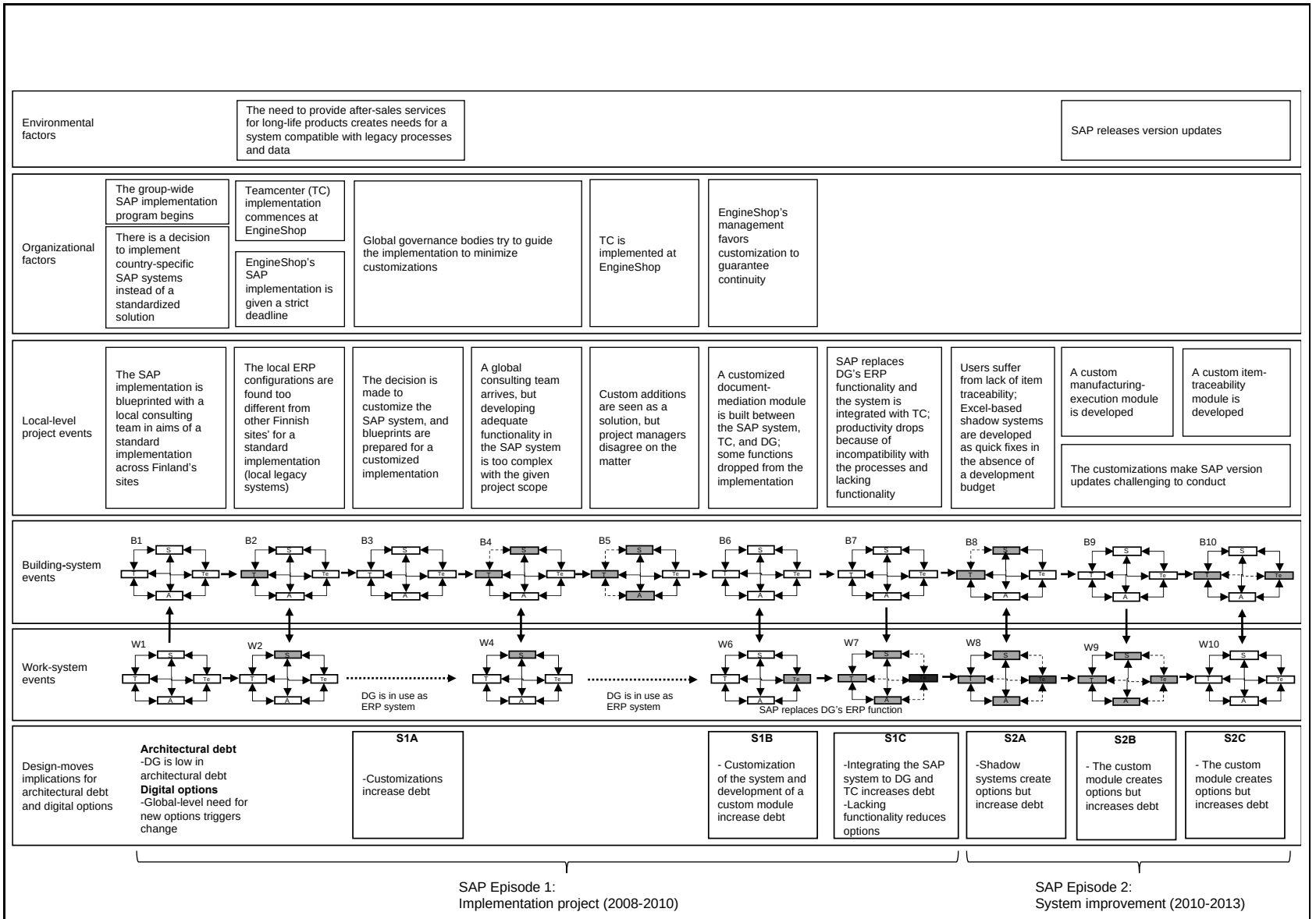

Figure A2. PSIC Process Analysis of the SAP System Implementation

Appendix B

The System-Dynamics Model's Constructs and Relationships

Tables B1-B3 explain the key constructs and their relationships in the proposed system-dynamics (SD) model (Figure 4).

Table B1. A Summary of the SD Model's Constructs and Relationships for Loops R1 and R2

Causal construct	Description	Response construct	Dynamic relationship
Market dynamism	The rate of change in the organization's competitive environment (globalization, disruptive competitors, etc.)	New business opportunities	Changing dynamics of market demand and competition create opportunities to do business in a novel manner or in new areas (Baker & Singh, 2019; Sambamurthy et al., 2003).
Technology advancement	Emergence and availability of new technologies	Global digital options	Technology advancement in the environment produces new digital options, manifested in new IS solutions (Sambamurthy et al., 2003). In increasingly globalized business environments, these include global coordination options afforded by commercial technologies that are standardized and scalable.
Global digital options	A set of IT-enabled capabilities in the form of digitized enterprise work processes and knowledge systems accessible to a global organization's management	Agility	A global organization may achieve agility by exploiting digital options that provide new capabilities (Overby et al., 2006; Sambamurthy et al., 2003).
Agility	The organization's ability to change rapidly or adapt in response to changes in its environment	New business opportunities	Agility enables pursuing new business opportunities since the organization can better respond to environmental changes and harness the consequent opportunities (Sambamurthy et al., 2003).
New business opportunities	The opportunity to initiate competitive actions in a novel manner or in new areas	Technical COTS-system implementation	Seizing new business opportunities often reveals a need for new systems that produce additional value for the new business area/model, thereby increasing pressure to implement new systems at local sites.
Technical COTS-system implementation	The degree of implementing COTS systems at a local site	Local digital options	IT is a "digital options generator" (Overby et al., 2006; Rolland et al., 2018; Sambamurthy et al., 2003): implementing new systems produces new digital options for the local site, such as the ability to collaborate with another site.
Local digital options	A set of IT-enabled capabilities in the form of digitized enterprise work processes and knowledge systems accessible at a global organization's local site	Global digital options	Complex configurations and combinations of digital options at different local sites create options that are accessible only at the global level (e.g., coordination options).
Technical COTS-system implementation	The degree of implementing COTS systems at a local site	Global technical debt	A globally operating organization can reduce its IT maintenance obligations by deploying identical COTS systems at its different sites. Such harmonization of global IT governance can help to increase organizational agility.
Global technical debt	The global IT management's IT maintenance or governance obligations caused by non-harmonious global IT architecture—i.e., maintaining unique/non-standard IT systems at local sites	Agility	Global-level technical debt (especially architectural debt due to unique local systems) hinders the global organization's agility as the debt renders implementing organization-wide IT changes increasingly difficult.

Table B2. A Summary of the SD Model's Constructs and Relationships for Loops R3 and R4

Causal construct	Description	Response construct	Dynamic relationship
Technical COTS-system implementation	The degree of implementing COTS systems at the local sites	Discontinuance of legacy systems	Implementing a COTS replacement system typically involves (or is intended to involve) the discontinuance of an incumbent legacy system (Furneaux & Wade, 2011; Mehrizi et al., 2019).
Discontinuance of legacy systems	The extent to which the organization decommissions its incumbent legacy systems	Local technical debt	Discontinuance of legacy systems gives the organization an opportunity to settle the technical debt embedded in those systems.
Local technical debt	Local site's IT maintenance obligations that will need addressing in the future, caused by sub-optimal shortcuts in system design or implementation	Global technical debt	Technical debt manifested at various local sites builds up into debt that hinders IT governance at the global level of an organization. This is a bottom-up process by which complex configurations of various debt types at different sites create technical rigidity globally.
Customization of COTS systems	Technical configuration, extension, or modification of a COTS system that causes it to deviate from its standard form, function, or structure	Local digital options	Customization of COTS systems unlocks new functions that are not available in the standard COTS system, potentially providing competitive advantage (Davenport 1998).

Table B3. A Summary of the SD Model's Constructs and Relationships for Loops B1, B2, and B3

Causal construct	Description	Response construct	Dynamic relationship
Technical COTS-system implementation	The degree of implementing COTS systems at the local sites	Manifestation of social inertia	Implementing COTS systems in local legacy environments shakes the social system out of its static state and stimulates some social changes, and the social system responds to this disruption by exhibiting social inertia—resisting change (Besson & Rowe, 2012).
Manifestation of social inertia	Resistance by the organization's social components (people, processes, and hierarchies) to the change introduced	Customization of COTS systems	Social inertia incentivizes managers to customize the new technology to match the incumbent social system, so as to avoid resistance and let business operations proceed with minimal disruption (Arvidsson et al., 2014; Luo & Strong, 2004).
System-develop-ment competence	The technical competence and expertise available for software-system development and maintenance	Customization of COTS systems	Locally available competence for developing and modifying software systems creates favorable conditions for customizing COTS systems.
Customization of COTS systems	Technical configuration, extension, or modification of a COTS system that causes it to deviate from its standard form, function, or structure	Local technical debt	Customizing COTS systems plants technical debt locally, because it strays from standard functionality, increases complexity, and renders changing the IT infrastructure more laborious and difficult over time; this often involves increasing architectural debt via unsystematic and latent dependencies, which then tends to result in increases in other types of debt (code debt, people debt, documentation debt, etc.) (Luo & Strong, 2004; Rolland et al., 2018).

Discontinuance of legacy systems	The extent to which the organization decommissions its incumbent legacy systems	Manifestation of social inertia	Removing legacy systems that are deeply embedded in their organization shakes the social system out of its static state and stimulates some social changes, and the social system responds to this disruption by exhibiting social inertia—resisting change (Besson & Rowe, 2012).
Manifestation of social inertia	Resistance by the organization's social components (people, processes, and hierarchies) to the change introduced	Discontinuance of legacy systems	If the change stimulates strong enough social inertia, it will stall, hamper, or prevent the full removal of legacy systems.
Local technical debt	Local site's IT maintenance obligations that will need addressing in the future, caused by sub-optimal shortcuts in system design or implementation	Discontinuance of legacy systems	Accumulating technical debt makes complex and dramatic technical changes increasingly difficult, thereby inhibiting the discontinuation of legacy systems.