Mäki, Netta; Penttinen, Esko; Rinta-Kahila, Tapani

A Domino Effect: Interdependencies among Different Types of Technical Debt

Published in:
Proceedings of the 56th Annual Hawaii International Conference on System Sciences

Published: 01/01/2023

Document Version
Publisher's PDF, also known as Version of record

A Domino Effect: Interdependencies among Different Types of Technical Debt

Netta Mäki
Aalto University
netta.maki@aalto.fi

Esko Penttinen
Aalto University
esko.penttinen@aalto.fi

Tapani Rinta-Kahila
The University of Queensland
t.rintakahila@uq.edu.au

Abstract

The paper examines the accrual of technical debt, which represents an increasingly pressing concern for many organizations. To advance understanding of how this debt-accumulation process unfolds, an in-depth case study was conducted with a large manufacturing firm for identifying particular types of technical debt and potential interdependencies among them. The findings point to architecture debt being "the root of all evil" at the case company, setting in motion dynamics that led to the development of other types of technical debt. Scholarship should benefit from this nuanced articulation and illustration of interdependencies across the various types of technical debt.

Keywords: technical debt, architectural debt, legacy systems, manufacturing, case study

1. Introduction

Many organizations, especially those operating in traditional fields such as manufacturing and banking, continue to rely on old legacy systems for mission-critical applications. While these systems provide business continuity and support reliability, their operators are awakening to the fact that such systems often accrue significant technical debt – future information technology (IT) maintenance obligations, most often through suboptimal shortcuts taken in information systems' development.

Much as financial debt is part and parcel of business, technical debt in information systems has been described as practically unavoidable (Casey, 2020). Furthermore, just as with financial debt, there is a threshold at which technical debt becomes problematic. While technical debt cannot be avoided entirely, it needs to be managed consciously – otherwise it becomes expensive and hinders operations and innovation. Indeed, company CIOs interviewed in a McKinsey survey (Dalal et al., 2020) stated that up to 20 percent of their new-product budget ended up going toward resolving existing issues related to technical debt. In other words, money set aside for new IT systems and solutions had to be directed to fixing problems that had accumulated in the systems already in use. Worryingly, reports cite technical debt as diminishing innovation and change within companies while also negatively influencing both security risks and business-continuity risks (McClure, 2018). Technical debt can be especially harmful in the long term, and, if ignored for an extended period, it starts bringing negative consequences and rendering companies more vulnerable. Therefore, focused efforts to understand how technical debt accumulates in organizations are highly relevant. Managers need to be able to identify the mechanisms that lead to debt accrual to be able to consciously manage it.

Considerable attention has been devoted to addressing the origins of technical debt (e.g., intentional vs. unintentional or internal vs. external) and its types (e.g., architecture debt, code debt, and infrastructure debt). However, recent research has found that technical debt can be contagious: increases in one type of debt (cause) often result in increases in another type (effect), making them interdependent (Martini et al., 2015; Rinta-Kahila et al., forthcoming.; Rolland & Lyytinen, 2021). This suggests that repercussions of technical debt may be dynamic in nature. Recent information systems (IS) literature indicates that many challenges facing managers are systemic: they require holistic understanding of how different social and technical elements interact within an organization over time (Burton-Jones et al., 2015; Rinta-Kahila et al., 2022). As such, vicious feedback cycles caused by seemingly innocent decisions during system development and maintenance may unconsciously feed the accrual of technical debt in an organization's information-systems architecture (Rinta-Kahila et al., forthcoming). Such dynamics seem to receive little attention in the literature on technical debt, however. Similarly, managers have been found to exhibit limited awareness of how their decisions contribute to the accrual of various types of technical debt (Holvitie et al., 2018; Martini et al., 2015; Rinta-Kahila et al., forthcoming). Therefore, we

URI: https://hdl.handle.net/10125/103356
978-0-9981331-6-4

attempted to improve understanding of how the individual types of technical debt may fuel each other, by considering the question *"what kinds of interdependencies exist among different types of technical debt?"*

To tackle this research question, we examined a manufacturing firm where old legacy systems were still in use, a potentially rich empirical context for studying the accrual of technical debt. Interactions with the case company for other research projects had produced a good general sense of the company's system landscape, and we built on this understanding via nine interviews targeted specifically at understanding how various kinds of technical debt unfold.

Our discussion is anchored on the types of technical debt presented in prior literature. After providing a brief literature review, we describe how we collected and analyzed the empirical data. We then turn to our empirical findings and present the study's implications for theory and management practice.

2. Technical debt

The introduction of the concept of technical debt, by Ward Cunningham in the early 1990s, coincided with that of agile methods (Ampatzoglou et al., 2015). Originally, its scope was quite limited, with Cunningham defining technical debt as badly written code (Cunningham, 1992). Technical debt parallels with financial debt, and much of the terminology reflects this. For instance, studies often refer to principal and interest: technical debt's principal is the original cost of fully eliminating the debt, and all additional issues and costs accrued through acquiring the original debt are interest, with one example being productivity reductions linked to having technical debt (Alves et al., 2016).

2.1. Types of technical debt

Technical debt comes in various types, depending on where in the IT system the debt accumulates. Next, we discuss technical debt at the levels of software code, design, testing, requirements specification, IT infrastructure, IT architecture, data, documentation, and people.

Code debt covers problems that make source code hard to understand, maintain, and read. "Code smell" and "grime build-up," or characteristics of program code that may betray deeper problems, are the most oft-cited indicators of code debt (Ramasubbu & Kemerer, 2019). Code debt may be either chosen or accrued inadvertently. In the former case, developers' comments in the code state explicitly that the implementation will not suffice in the long run and needs further attention at some point (Maldonado & Shihab, 2015). **Design debt** too refers to source-code problems, with its biggest giveaway being disregard for common principles of object-oriented design (Guo & Seaman, 2011). Linked to software development, design debt arises if, for instance, changes are made without proper refactoring of existing code (Neill & Laplante, 2006). **Test debt**, in turn, encompasses issues with information systems' testing, mainly utter lack of testing but also poorly executed tests and inappropriate testing (Guo & Seaman, 2011). One symptom of test debt in information systems is a sudden need for formerly automated tests to be performed manually (Tom et al., 2013). Also, difficulties in finding every critical defect even with rigorous testing could imply test debt in the systems, since something is rendering the tests clearly ineffective (Li et al., 2015). Test debt is a type of debt wherein human error and input play an important part: testing readily gets forgone amid pressure for speed and efficiency. Lack of motivation constitutes another cause of information-system test debt—if testing is not done on schedule and properly, the probability of system failure rises significantly, and system security may be utterly compromised also (Shull, 2011). **Requirements debt** accumulates from trade-offs in what requirements are met, when, and how (Alves et al., 2016). It can arise if some needs are not fully addressed in the final system or if the implementation impinges on meeting other requirements (Kruchten et al., 2012).

Infrastructure debt, on the other hand, refers to issues that leave information systems falling short of modern standards and requirements. Lack of continuous integration between old and new production systems could fall under this category (Li et al., 2015). Infrastructure debt is usually connected with legacy systems. Yet, resolving such debt by replacing legacy systems with more modern ones is notoriously difficult and risky (Rinta-Kahila, 2018; Rinta-Kahila et al., forthcoming). **Architecture debt** accrue through suboptimal choices in the software architecture that increase complexity and decrease stability, such as creating technological gaps by integrating old technology with new one (Kruchten et al., 2012). It also accumulates through efforts to make systems more flexible and adaptable than necessary. Feature creep can render future changes needlessly complex (Kruchten et al., 2012). One typical indicator of architecture debt is violations related to modularity (Alves et al., 2016), and one can investigate dependencies at architecture level to ascertain whether architecture debt is an issue. The parts of a healthy information system work seamlessly together, without their core-level functions necessarily requiring the other parts' presence, while systems' excessive dependence on each other makes maintenance very difficult and increases risk (Martini et al., 2015). The likelihood of unnecessary additional architecture

debt is higher with heavily customized information systems than with less tailored systems (Ramasubbu & Kemerer, 2016). Business evolution and new information can make even a highly functional architecture suboptimal with time (Martini et al., 2015).

Furthermore, low-quality data embody **data debt** (Foidl et al., 2019), e.g., data that is incorrectly formatted, inaccurate, redundant, or absent altogether. In knock-on effects, employee workarounds to bad data may increase complexity, thereby impairing the whole information system's maintainability (Foidl et al., 2019). Inability to utilize data when needed may point to information systems suffering from data debt, whether employees find it inconvenient to access the necessary data, workers have no access at all, or data are stored in an inconvenient location or hidden completely. Systems' data debt may manifest itself also in processes being hard to manage, because controlling the data for their execution is impossible. Despite its potential significance, data debt has thus far received very sparse attention in the academic discourse.

Documentation debt accrues when developers neglect clear indications to others of their decision processes or the changes made (Alves et al., 2016). Missing or incomplete project documentation can form documentation debt (Zazworka et al., 2013). Likewise, poor-quality or outdated documents may be regarded as documentation debt (Guo & Seaman, 2011). Such documents often contain valuable information, so companies with documentation debt can face surprisingly dire consequences if important enough information disappears or is not available in a suitable form when needed. When projects, changes, etc. are not documented clearly enough, negative impacts on future upgrades and maintenance may ensue, and any future changes may end up unnecessarily difficult: in the absence of references to earlier decisions and reasons for something having succeeded or failed, every individual decision and factor must be researched again, from the beginning.

People debt manifests in insufficient training and hiring practices (Guo & Seaman, 2011) and gaps in knowledge distribution (Tom et al., 2013). The term covers technical debt arising from issues of transmitting knowledge between company-internal parties. If knowledge evaporates, it could grow difficult to stay on top of even the simplest tasks and processes, let alone later improve the systems or make other advances without additional time and effort.

2.2. Origins of technical debt

Technical debt may accumulate intentionally or unintentionally. Intentional technical debt can be seen as investment in an asset that should yield gains, usually short-term benefits (e.g., a competitive edge). Such gains are often conceptualized via digital options – potentially transformative IT-enabled action possibilities that would otherwise remain beyond reach (Rolland et al., 2018; Woodard et al., 2013). Potential access to these impactful opportunities is one key reason for intentionally taking on technical debt, alongside settling prior debt (as prior debt could hamper the realization of options). However, pursuing digital options without adequate planning can impede progress and build unintended debt in the systems, and such debt can restrict digital options' realization and the benefits (Rolland et al., 2018). This is especially the case when the debt accumulates latently outside the organization's awareness. For this reason, it is important for organizations to systematically manage their technical debt (Ramasubbu & Kemerer, 2019).

Technical debt has internal and external sources. It may stem from an organization's internal actions that prioritize strategic digital options over systematic IT management. The more a firm must resort to such compromises, elevating or overlooking anything from time or money to project-specific staffing, the more technical debt it is probably building into its information systems. The passing of time and general technological progress are significant external sources of technical debt, as often systems that were once cutting-edge and lean will ultimately become obsolete and wieldy (Kruchten et al., 2012). Similarly, mounting competition may create pressure to save time and keep up, rendering incumbent systems less relevant. Further, loss of knowledge via departure of important employees may result in people debt for reasons external to the organization (Fairley & Willshire, 2017). While such external factors may be outside one organization's control, managers can track developments and attempt to predict them, to avoid hasty decisions amid scrambling to keep up.

Finally, technical debt may in itself stimulate the accumulation of further debt. For instance architecture debt has been found to increase other types of debt: when complex and non-standard architectural solutions are left undocumented (documentation debt), when only few people understand them (people debt), and when those solutions increase the entrenchment of old technologies (infrastructure debt) (Rinta-Kahila et al., forthcoming.; Rolland & Lyytinen, 2021). However, this aspect of debt accumulation has not received much attention in prior literature. Hence, the present study provides a systematic investigation of the matter.

3. Methods

Technical debt can be identified by means of human knowledge and experiences or via automated tools

(Zazworka et al., 2013). Our project employed the former since a qualitative approach was deemed the best way to gain rich and empirically-grounded insights on the occurrence of different types of debt, allowing us to approach our under-explored research question on the potential interdependencies among them. We conducted an exploratory single-case study (Yin, 2018), for the richest, most detailed view possible of how technical debt might be manifested in information systems.

3.1. The case company

We carried out the study at a Finnish manufacturing site that focuses on the production of electric engines and similar equipment. The site's parent corporation, which employs over 100,000 people around the world and has customers in diverse industries, focuses predominantly on the production of technology but offers related services too, such as machine maintenance. The current investigation was conducted as a part of a larger research project in this case organization. Our prior engagement with the organization had indicated that their IT environment is heavily affected by various types of technical debt (see (Rinta-Kahila et al., forthcoming). Managers at the site were concerned about potential implications of incurring technical debt, but they were lacking a holistic understanding on how and where debt accumulated. Hence, we agreed to conduct a more focused examination of technical debt and its different manifestations.

As we discussed potential approaches to the research with the site's IT managers, a decision was made to examine how technical debt manifests in the design processes of two specific types of motor, here referred to as types *A* and *B*. This process involves creating and fulfilling orders, as well as product and design development in general. The two electric motors operate on the same fundamental principle. However, *type-A* motors are highly customizable to the customer's needs, even down to the specific nuts and bolts used. *Type-B* motors offer less flexibility for customization (they follow standardized designs, to guarantee shorter lead times and a lower price) though they can be modified to some extent. The reason for focusing on these processes was that they are managed with a variety of information systems, both legacy ones and newer ones. The systems in question were found likely to exhibit different types of technical debt, considering that they have varying amounts of customization done to the software architecture and functionality, with concerns of insufficient documentation and skills to maintain them. Often, the demands of a customer would trigger a need to customize the product, and if the information system in its standard form could not support this, it might receive potentially debt-incurring modifications.

3.2. Data collection

Data for this research were collected by means of semi-structured interviews specifically designed to probe the manifestation of different types of technical debt. A set of guiding questions was compiled to guarantee that the interviews stayed on track and keep their scope under control. The questions covered basic information on the interviewees' tasks and job role at the case company and on how the design process for the two types of motors works, then continued on to more specific questions about process functionality. The questions (interview protocol in Appendix A) were designed not only to chart where any technical debt resides in the information systems but also to help divide the debt into the types recognized in the literature. Open questions and opportunities to probe more about certain answers aided in collecting an in-depth set of data for what people perceive and experience.

Our contact persons at the site identified a set of relevant employees to be interviewed. From there, we continued to identify additional informants by snowballing. All the interviews (see Table 1), conducted in December 2020, advanced from general questions to more in-depth, detailed ones, and every interviewee was asked all of the questions, for comparability of the data. Every interview was recorded and transcribed.

Table 1. Data collection

Informant	Role	Interview length
Chris	IS manager	66 min
Bob	Manager, engineering tools and processes	69 min
Mark	Head of product platforms and R&D	56 min
John	Team leader, mechanical-design engineer	74 min
Alex	System engineer	64 min
Tom	Team leader and product-owner, R&D	65 min
Matt	Senior software-development engineer	80 min
Dave	Project manager	57 min
Luke	Manager, sales tools and processes	46 min

3.3. Data analysis

Transcribing the interviews yielded about 130 pages of text. The semi-structured approach produced material that already had a somewhat systematic form, as could be expected. Two of the authors read the transcripts inductively to identify important themes and get a sense of the occurrence of debt types. This was

followed by a more systematic analysis by one of the authors to identify different types of technical debt and interdependencies among them. The data was analyzed via the lens of existing debt types per Alves et al. (2016) and Tom et al. (2013) by assigning codes to interview excerpts that reflected the types described in the said studies. The informants' statements were examined critically against previous definitions and empirical manifestations of technical debt to ensure a sound understanding of debt occurrence in the case company. The material was tabulated into a single set of data by recording every indication of technical debt into a conceptual matrix. The matrix covered debt types established in the prior literature, deliberativeness of debt accrual, its causes and consequences, and characterization of the case company's decision-making process in relation to technical debt.

When assigning causes to debt accrual based on the informants' testimonies, we noted that in many cases the primary cause for accruing one type (e.g., leaving integration of two systems undocumented) seemed to boil down to the prior accrual of another type (e.g., resorting to a complex, non-standard architectural integration of two systems). These emergent findings guided our attention to the interdependencies between different debt types. We then conducted a more specific analysis of the interdependencies by re-examining every segment of data where the two or more debt types were indicated together. This final analysis allowed us to verify the causal connections between different types of debt.

4. Findings

4.1. Evidence of technical debt

We found evidence of various types of technical debt. The most prominent debt type was architecture debt. High architectural complexity, large number of different systems, lack of smooth integration, and excessive system customization were said to slow down work processes, disrupt data flows, and make updates difficult to execute. One interviewee, when asked whether the data flow well, answered thus: *No. Every move requires a manual trigger for the data to move onwards.* (John, team leader for mechanical-design engineering). Issues of data formatting signaled also of data debt, with some interviewees seeing problems in transferring data between separate locations, plants, and facilities. The international nature of the case company was tied in with this type of data debt, and examples of formatting issues included difficulties in translating between languages or alphabets. Differences in data formats were brought up also in relation to the internal design process.

We conceived of infrastructure debt mainly in terms of the issues arising because legacy systems were excessively interwoven in the process. The case company's architecture for the design process for both motor types features a mix of newer, more modern tools with legacy systems that have been in place for many years. Three of the interviewees mentioned issues with the interaction between the old and new systems—for example, because the new systems were seen as too rigid and the old ones too customized. These issues were cited as rendering the whole process slower than necessary, with the mix of systems demanding more manual labor than either set might on its own. There were mentions also of how the overall design process for both motor types is not up to modern standards. These interviewees continued by pointing to the legacy systems as a possible central factor in this: *[Satisfaction with the design process] is a multifaceted issue. The first problem is that the tools we use in the design are not necessarily up to the standards that are required in design and product structure in the 2020s.* (John, team leader for mechanical-design engineering)

People debt was mentioned mainly in connection with the older legacy systems. Their maintenance and use require a very specific skillset, possessed by only a few people at the case company and even in the world. Since these systems are, in essence, reliant on a single person, they are a risk and a possible bottleneck to the whole design process. This becomes an issue. *How much do we want to put behind one person? [...] we are screwed if a guy gets hit by a bus...* (Matt, senior software-development engineer) Also lack of documentation, especially of smaller changes to the process, was brought up often. It was attributed mainly to lack of time but also to individuals' attitudes. There were mentions of documentation being missing too on account of storage in illogical places but also because some documents were reportedly corrupted, in incompatible format, or completely illegible. This was true mainly of older documentation, material that describes legacy systems still in use. Finally, interviewees mentioned lags in documentation, arising for the same reasons cited for lack of documentation.

4.2. Interdependencies among types of technical debt

Existing technical debt in the case company was recognized as leading to accumulation of additional debt, leading us to uncover signs of cause–effect relationships among different debt types. Importantly, architecture debt was the dominant element in many of the dependencies recognized, as both cause and consequence. Next, we discuss the observed interdependencies (summarized in Figure 1).

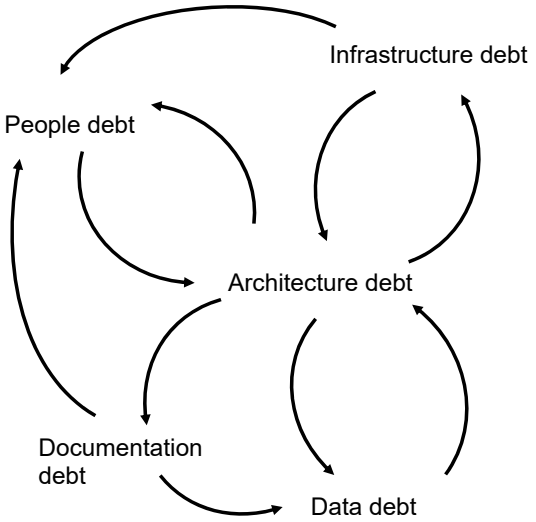

Figure 1. Empirically observed interdependencies among types of technical debt

4.2.1. Architecture debt and infrastructure debt. In a textbook example of architecture debt, interviewees mentioned that the numerousness of steps and IT tools in the motor-design process has made the system architecture increasingly complex. The reason for the ever-increasing number of systems was the fact that the company was accumulating infrastructure debt: it was rarely able to fully remove its legacy systems when implementing new replacement systems. Retaining archaic systems has forced the site to resort to suboptimal architectural solutions. As the new systems got integrated into old ones, resolving infrastructure debt had become more difficult due to the increasing stickiness of decaying but still functioning legacy systems. This represents a vicious cycle of architecture debt and infrastructure debt fueling each other.

4.2.2. Architecture debt, people debt, and documentation debt. The company's employees were described as tending to specialize in only a few of the tools and systems in this process, or just one. This is a concrete manifestation of people debt. One of the causes mentioned for this specific issue was that, with so many tools to keep up with (and, furthermore, some of them being difficult to operate and update), it is easier to confine oneself to a single specific tool. Because people at the company recognized the complex architecture and systems as pushing employees toward limiting their know-how to just a few systems, architecture debt can be viewed as adding to the undesired debt burden at the firm, here in the form of people debt.

Another link visible between architecture and people debt was that of customization. The case company's highly customized systems—both tailor-made legacy systems built in-house and commercial systems created specifically for the company's environment—not only can complicate maintenance and updates but also, according to interviewees, limit employees' knowledge of the tools. Lack of detailed documentation may enter in, further contributing to restricted expertise. Thus, documentation debt emerges as an interesting third link in the chain of dependencies. Since the customizations cater specifically to the case company, no standard manual covers how to operate and update the systems. Neither is such a thing prioritized: attitudes toward documenting minor-seeming changes were described as lackluster. Hence, some information may be missing or completely lost on many occasions, so managing the process is rendered harder in general. Changing the information systems is trickier too, in the absence of proof of what happened the last time something was changed. Though larger changes reportedly were documented fairly well, with no further issues of consequence arising, the interviewees definitely saw the lack of reporting on smaller issues as a source of layers of problems.

The dependencies between architecture debt and people debt do not end with the examples presented above. For one thing, effects can flow in both directions: people debt may be viewed as a possible cause for architecture debt. Again, where know-how and skills are limited to a few employees per system, making genuinely optimal changes to the process is harder. This, of course, applies to bigger changes, but the effects extend to even the smallest of everyday decisions. With a complete system replacement being extremely difficult, the case company decided to build on top of its legacy systems, keeping the parts that cannot be easily replaced. This decision retained the liability from the older tools, operable by only a few employees. Another possible cause of complexity, their presence in the process could be considered a loop of sorts—limited know-how can arise from systems being complex to operate from the outset.

4.2.3. Architecture debt and data debt. Architecture debt is stimulating additional data debt too. The complex architecture demands extensive manual effort for entering data in the course of the motor-design process (e.g., as noted above, there were instances wherein a given data item had to be input multiple times). One of the reasons listed was the systems being thoroughly entangled with each other and not operating in absolute harmony. Additionally, interviewees stated that the master data-entry point was sometimes unknown or inaccessible, thanks again to the information-systems architecture being so serpentine.

We identified an interdependency between architecture and data debt in the process. Poor data flow and data even disappearing (whether from view or from existence) were among the other effects of the design process comprising too many stages. There is some

irony here: some of these steps had been added to the process to improve data flow and keep the process functioning. In one example, an older legacy system had to be kept alive for compatibility: data from earlier parts of the process did not interface with the commercial systems introduced for its later stages. This is clearly not the best solution from a wider perspective, since adding still more stages to a process that is already somewhat of a tangle just increases complexity. With things as they stand, the architecture debt in the system is creating data debt, which, in turn, is piling on further architecture debt. A vicious circle of technical debt has been etched into the systems. Yes, the process functions, but the issue could end up growing much more extensive in the long run than just a need for employees to navigate a few extra steps in their daily work.

4.2.4. Documentation debt, infrastructure debt, and people debt. Both lack of documentation and prevalence of old systems results in IT tools' heavy reliance on a shrinking pool of personnel. Individuals' choices of how thoroughly and methodically to document changes, processes, etc. is making it harder to train new employees at the case company. This is particularly relevant in relation to the legacy systems, and the company has struggled to make sure employees responsible for these parts of the process have others to share the load. The knowledgeable personnel must devote considerable time to teaching these peers, though, because system documentation is non-existent, hard to find, or in an unreadable format. While this relationship between people debt and documentation debt is not necessarily bringing the company immense amounts of further debt, it does hamper efforts to eliminate existing technical debt.

4.2.5. Documentation debt and data debt. Additionally, some interviewees noted that, in the absence of clear guidelines on where and how to save data, they had established their own ways of working. With no comprehensive way of storing all the data for the design process, and with people generally operating as they see fit, it can be nearly impossible to locate particular data when needed. The data might even be stored on an employee's personal computer. When the data are in a hard-to-reach location, the search takes time, drawing resources away from other tasks and duties.

5. Discussion and conclusions

The literature lists characteristics of technical debt that can render it contagious (Martini et al., 2015; Rinta-Kahila et al., forthcoming.; Rolland & Lyytinen, 2021). Delving further into such relations, we found many examples of one type of technical debt fueling more of some other type of debt and identified concrete examples of how technical debt can "infect" information systems further, in a seemingly unending loop. In contrast, prior research has confined its examination of such relationships mainly to a single type of technical debt (for instance, complex architectures leading to even greater complexity). Foidl et al. (2019) are among the few to have looked at the relationship between two distinct types of technical debt, by describing how bad data can prompt people to find shortcuts that may produce increased complexity. Similarly, Rolland and Lyytinen (2021) and Rinta-Kahila et al. (forthcoming) have identified architectural debt as a major cause for the incurrence of some other debt types. Continuing in this direction, we systematically found further indications that technical debt, of various sorts, can produce further debt of a completely different type. Furthermore, we explored how these relationships can result in vicious circles.

Researchers looking at technical debt have pointed to vicious circles involving architecture debt (Martini et al., 2015), and architecture debt likewise played a role in many of the cause–consequence relationships recognized in our case study. We identified architecture debt as the main culprit in fueling (and being fueled by) infrastructure debt, data debt, people debt, and documentation debt. In addition, we found indication of architecture debt's potential to generate test debt too. Customization work done on the commercial systems renders it significantly harder and more expensive to test the systems for robustness to impending change. While the informants did not indicate that shortcuts had been taken in testing, the increasing complexity of testing suggests that test debt might become an issue in the future. Given the huge negative effect these vicious circles can produce, restricting one's view to only a single type of technical debt could obscure key patterns. Our contribution lies in unraveling these reinforcing feedback loops, which we have illustrated empirically in the findings section and visually in Figure 1. In addition, we provide empirical evidence on data debt, a type of technical debt that has received surprisingly little attention in academic literature.

Furthermore, our findings address the contention of Brown et al. (2010) that technical debt may accumulate in companies for its potential to settle debt incurred earlier. In our study, the case company's complex information architecture was rendering the data flows imperfect, so elements were introduced to make sure the data flow smoothly. These additional elements complicated the overall architecture still further. Even though such situations were recognized in the case company, settling other debt was not, in fact, the most prominent reason for technical-debt co-dependencies. Rather, the types of debt created in consequence of existing debt tended to be more an unintentional side

effect than fruit of a deliberate decision. The various dependencies identified speak directly to the question of intentionality. The dataset revealed a surprisingly large quantity of intentional technical debt in the case company, mainly from items of architecture debt accepted on purpose, or at the very least in the knowledge that the decision would bring on debt.

5.2. Management implications

An understanding of how technical debt can lead to more debt is crucial for any company that wishes to fully control and manage the technical debt it incurs. It is important for an organization's decision-making to consider not only how types of technical debt have been interdependent in the past but also how the decisions and changes could create and interact with further debt in the future. Especially in settings such as the case company's, wherein things are "ready to ignite" according to the interviewees, no unwanted technical debt should be taken on if it is avoidable. Awareness of the possibility of debt creating more debt can minimize the risk, also long before a possible ignition point.

Evaluating the interest on technical debt entails examining the likelihood of any kind of interest (Alves et al., 2016). Because there can be cause–consequence relationships aplenty between debt types, the probability of mounting interest is well worth considering. When the chances are fairly significant, action to address it should be taken without delay. Likewise, the interest already accumulating on technical debt must not be taken lightly. The case company illustrates this well.

5.3. Limitations and further research

This study leaves ample room for further research. It is hard to generalize findings from a single-case study, so similar work could be done usefully at multiple manufacturing companies to validate or challenge the results. Our findings point toward a number of interesting interdependencies between different types of technical debt but do not offer (analytically) generalizable mechanisms. Future research could continue probing these insights and apply systems-theorizing (Burton-Jones et al., 2015) to reveal general feedback loops between organizational behaviors, structures, and accumulation of different types of technical debt.

References

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

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: a systematic literature review. *Information and Software Technology*, *64*, 52–73.

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. *FoSER 2010*, 1–5. https://doi.org/10.1145/1882362.1882373

Burton-Jones, A., McLean, E. R., & Monod, E. (2015). Theoretical perspectives in IS research: From variance and process to conceptual latitude and conceptual fit. *European Journal of Information Systems*, *24*(6), 664–679. https://doi.org/10.1057/ejis.2014.31

Casey, K. (2020). *What causes technical debt - and how to minimize it*.

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

Dalal, V., Krishnakanthan, K., Münstermann, B., & Patenge, R. (2020). *Tech debt: Reclaiming tech equity*. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/tech-debt-reclaiming-tech-equity

Fairley, R., & Willshire, M. (2017). Better Now Than Later: Managing Technical Debt in Systems Development. *Computer*, *50*(5), 80–87.

Foidl, H., Felderer, M., & Biffl, S. (2019). Technical Debt in Data-Intensive Software Systems. *45th Euromicro Conference on Software Engineering and Advanced Applications*.

Guo, Y., & Seaman, C. (2011). A portfolio approach to technical debt management. *Proceedings of the 2nd Workshop on Managing Technical Debt, MTD 11*, 31–43.

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*(December 2017), 141–160. https://doi.org/10.1016/j.infsof.2017.11.015

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

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

Maldonado, E., & Shihab, E. (2015). Detecting and quantifying different types of self-admitted technical debt. *7th International Workshop on Managing Technical Debt, Montreal, Canada*.

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
McClure, D. (2018). *Decouple to Innovate: How federal agencies can unlock IT value & agility by remediating technical debt, Accenture*.
Neill, C., & Laplante, P. (2006). Paying down design debt with strategic refactoring. *Computer*, *39*(12), 131–134.
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). Caught in Between: How an Organization Became a Prisoner of Its Legacy System after IS Change. *International Conference on Information Systems (ICIS)*, 1–17.
Rinta-Kahila, T., Penttinen, E., & Lyytinen, K. (forthcoming). Getting Trapped in Technical Debt: Socio-Technical Analysis of a Legacy System's Replacement. *MIS Quarterly (Forthcoming)*. https://doi.org/10.25300/MISQ/2022/16711
Rinta-Kahila, T., Someh, I., Gillespie, N., Indulska, M., & Gregor, S. (2022). Algorithmic decision-making and system destructiveness: A case of automatic debt recovery. *European Journal of Information Systems*, *31*(3), 313–338. https://doi.org/10.1080/0960085X.2021.1960905
Rolland, K. H., & Lyytinen, K. (2021). Managing Tensions between Architectural Debt and Digital Innovation: The Case of a Financial Organization. *Proceedings of the 54th Hawaii International Conference on System Sciences | 2021 Datafication*, *0*, 6722–6732. 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
Shull, F. (2011). Perfectionists in a World of Finite Resources. *IEEE Software*, *28*(2), 4–6.
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
Woodard, C. J., Ramasubbu, N., Tschang, F. T., & Sambamurthy, V. (2013). Design capital and design moves: The logic of digital business strategy. *MIS Quarterly: Management Information Systems*, *37*(2), 537–564. https://doi.org/10.25300/MISQ/2013/37.2.10
Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods* (6th ed.). SAGE Publications Inc.
Zazworka, N., Spinola, R., Vetro, A., Shull, F., & Seaman, C. (2013). A case study on effectively identifying technical debt. *17th International Conference on Evaluation and Assessment in Software Engineering*, 42.47.

Appendix A: The interview protocol

Introduction: why this interview is conducted, who is conducting the interview and research project, confidentiality and anonymity (recorded), structure briefly (semi-structured), time allocated.

Interviewee Background

1. What is your role in the company? What are your main responsibilities?

2. How is your work related to the type A and type B motor design process?
a. Have any of your previous work tasks been related to the design process?
b. How long have you been involved?

3. Which of the different systems are you most familiar with? Which do you use the most?

Type A and Type B Motor Design Process

4. Could you briefly walk me through what happens in the design process when an order is placed for a type A or a type B motor?

5. Are you content with the current design process?
a. What's good? What's bad? (Are the issues generally unique instances, or constant problems?)

6. Do you find the design process simple? Logical? Streamlined?

7. What kind of solutions and decision have been made in the past that are currently causing issues in the design process (regarding the different systems and data)?
a. On the whole process level, individual systems level, code level etc.
b. Are you missing out on something based on previous decisions?

8. What kind of solutions and decisions have been made regarding the design process systems that could cause problems/additional work and obligations/loss of effectiveness in the future?
a. On the whole process level, individual systems level, code level etc.

9. Why do you think these decisions were made?
a. Who was making these decisions?
b. What factors do you think were considered? (Time, money, resources, attitudes, prioritization etc.)

Types of Technical Debt

10. Do you think the different systems and tools used in the design process work seamlessly together? (architecture and data debt)
a. If you think of this particular design process as a chain, is there a certain weak link? A system that works particularly

well in the chain? (Does the weakness in one link affect the rest of the links significantly?)
b. Are the different systems very customized for the product's specific needs? Does this cause any issues in the whole architecture (now, in the future)?
bi.Why are they customized?
bii. Is the upkeep of the systems harder because of the customizations? How?
c. Do you think information and data flows easily between the different systems?
ci. Is the data in the right format? Is everything synchronized?
d. Is master data easy to control?
e. Is the data needed for the design process easy to access? Can it be utilized by everyone? Is it stored in an appropriate manner?
f. Is there necessary data missing?
g. What is the quality of the data like?

11. Do you have any older systems in place in the design process? (infrastructure/architecture debt)
a. Do you find it/them outdated? Is this causing problems?
b. Should the system(s) be replaced soon? Should they already have been replaced in the past?
bi. If not, are other modifications needed?
c. Is the interplay between older and newer systems working?

12. Are there known defects within the design process information systems? (defect debt)
a. Are they being immediately fixed? (Prioritization? Why?)

13. Is every change and detail regarding the systems documented actively? (documentation debt)
a. Is everything documented in a clear and understandable way?
b. Is there any paperwork missing?

14. Do you know if these design process systems were tested extensively before they were taken into use? (test debt)
a. Are they still being tested regularly?
b. Could they have been tested more?
c. Are the tests done in an appropriate manner?
d. Is testing considered to be very expensive?

15. Do people seem generally content with the design process and its systems? (people debt)
a. Do you feel you have enough knowledge and expertise to efficiently use the design process systems you need?
b. Have you run into people who feel they have inadequate knowledge?

16. Do the different systems in the design process meet all the requirements to design type A/type B engines efficiently? (requirements debt)

17. Are you familiar with the coding of these different systems? (code debt, design debt)
a. Have you run into any problems with the system source code?
b. Have you been notified of any problems with the system source code? (What kind of problems?)
c. Are there any computer-assisted methods in use to recognize "bad code"?

Origins of Technical Debt

18. Do you feel the issues mentioned are known consequences of intentional decisions?
a. If yes, were they made to be proactive or reactive?
b. Is/was everyone aware of these issues when important decisions are done/were made?
c. Who made the decisions? Would you have considered another way?

19. How familiar are you with the concept of technical debt?
a. Are you aware of any technical debt specifically in the design process?
b. Is anything actively done to address this debt?
c. Do you think this debt will need to paid off at some point?

Concluding Questions

20. Is there anything we have not yet touched on, that you would like to bring up regarding technical debt and the type A and B motor design process and its information systems?

21. Do you have any specific people in mind that could be interviewed next regarding these topics?