Robotic Process Automation from the Design-Capital Perspective – Effects on Technical Debt and Digital Options

Lauri Ruha
Aalto University
lauri.ruha@aalto.fi

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

Esko Penttinen
Aalto University
esko.penttinen@aalto.fi

Abstract

Robotic process automation (RPA) is an instantiation of lightweight automation that allows organizations to automate manual business processes quickly and at low cost without modifying the organization's underlying deep information-systems structures. While RPA endows organizations with digital options (e.g., automation ability, cost savings), its implementation is bound to incur technical debt (i.e., accumulate unwarranted complexity in the IT architecture). The paper reports on an action research study shedding light on how RPA ties in with these two notions of a firm's design capital: digital options and technical debt. Findings indicate that RPA can create digital options through improvements in knowledge reach, knowledge richness, and process richness. These benefits come at the cost of accumulating technical debt which stems from additional technical complexity and maintenance obligations.

Keywords: Robotic Process Automation, design capital, technical debt, digital options, action research

1. Introduction

As organizations undergo digital transformation and with work tasks having become digitized in growing numbers, a need for workflow-automation tools to perform these tasks has arisen. These automation-based tools promise more efficient operations, cost savings, and reduction in errors by enabling automation of tasks that were previously conducted manually. One such tool that has been increasingly gaining attention in both scholarship and practice is robotic process automation (RPA). It functions by mimicking the actions a human worker would take to carry out a task. An RPA workflow could be configured to consist of, for example, interacting with elements of an application's user interface that involve entering text or clicking. Compared to other automation tools, RPA is regarded as more lightweight automation technology (Bygstad, 2016) and an easy-to-configure solution that can be implemented and maintained with less investment of resources. As RPA delivers various cost- and efficiency-related benefits and enables human workers to focus on tasks that require more abstract thinking, the technology can be seen as a prominent source of digital options – novel IT-enabled capabilities with transformational potential (Rolland et al., 2018; Sambamurthy et al., 2003).

RPA's main advantage over more heavyweight automation (such as back-end ERP automation) lies in its relatively simple implementation process, which is a result of RPA's non-invasive nature (i.e., not requiring significant changes in the firm's IT architecture). However, implementing RPA on top of the existing IT architecture (Asatiani et al., forthcoming) will inadvertently increase complexity of the organization's IT environment. Hence, RPA could contribute to the accumulation of technical debt – inertia caused by technical systems' rigidity and increased IT maintenance obligations (Ramasubbu & Kemerer, 2016). Technical debt, if allowed to grow uncontrollably, may eventually hamper organization's agility and ability to unlock the digital options brought by RPA. However, previous RPA literature has mainly focused on RPA's benefits (e.g., Lacity & Willcocks, 2016), alternative RPA operating models (Asatiani et al., forthcoming), and RPA selection criteria (Asatiani & Penttinen, 2016) without considering its implications on digital options versus technical debt. In light of this and the growing interest in RPA, we strove to uncover how implementing RPA to automate a business process affects an organization's digital architecture and its ability to exploit that architecture.

We approach this question via the lens of design capital (Woodard et al., 2013), which refers to the cumulative stock of digital designs owned or controlled by a firm. Design capital incorporates both digital options and technical debt, suggesting that the balance between the two state can either allow for greater organizational agility or present challenges to efforts of developing designs and commercializing existing ones (Woodard et al., 2013). Hence, we asked: *"How does implementing RPA to automate a business process affect a company's design capital?"*

To address this question, we conducted an action-research project at a Finnish engineering company. We tracked the evolution of the firm's design capital by following and participating in an RPA development project. We identified option-creating, debt-creating, and debt-mitigating effects on design capital that may stem from an RPA implementation. The findings presented below indicate that the resulting change in the state of design capital can be considered fairly complex and involves numerous contributing factors. We reflect on these effects and, additionally, the extent to which one can consider them unique to RPA.

2. Background

2.1. Robotic process automation

RPA can be defined as a software-based solution configured to carry out repetitive operation-oriented tasks and procedures traditionally handled by humans. It is used to automate rules-based business processes that involve routine tasks, structured data, and deterministic outcomes, allowing employees focus more on higher-value work (Aguirre & Rodriguez, 2017). These tools typically interact with the user interfaces of other computer systems in the way a human would (van der Aalst et al., 2018) – for instance, automatically clicking and typing into Web-form fields. An RPA tool operates via mapping the actions needed to carry out a business process to the RPA tool's language such that the software robot can follow the rules set. The tools apply `if`, `then`, and `else` statements to structured data, typically using a combination of user-interface interactions (Tornbohm & Dunie, 2017).

RPA is implemented in an outside-in manner, where, instead of deep integration, the automation tool is "plugged" on top of the underlying system architecture. This approach differs from the classical inside-out method of business-process management, which often entails deeper integration between the automation tools and the organization's information systems (IS). Hence, setups employing RPA leave the IS unchanged (van der Aalst et al., 2018): RPA tools operate largely with the same user interface as a human, reacting to events on a computer screen instead of communicating with a system's application programming interface (API). The key difference, therefore, is that RPA tools typically act as front-end solutions while traditional automation tools act as back-end solutions (Asatiani & Penttinen, 2016). Another apparent differentiating factor is the relative ease and speed of RPA's configuration. Instead of programming, RPA interfaces work by means of simple logical statements and interactive elements, which are dragged, dropped, and linked to represent steps in the process. Hence, end-users can make the necessary modifications to the tool's configuration without having programming skills (Lacity & Willcocks, 2016). This, in turn, bridges the knowledge gap that can form when the implementers are more technically oriented personnel who lack deep understanding of the underlying process (Cooper et al., 2019). However, it still requires a basic understanding of information-system functionality – e.g., the structure of rule-based systems (loops, conditions, parameters, etc.), the use of data, and interfaces to applications (Hofmann et al., 2019). Also, RPA does not create a new application or store any transaction data, so there is no need for a data model or a database as there is in most business-process management systems.

2.2. Design capital

Woodard et al. (2013) provide a useful framework for examining the effects of RPA implementation on an organization's design capital – the cumulative stock of non-physical designs owned or controlled by the company. One may regard this, alongside other types of capital stock, as an economic factor of production. It comprises the capabilities of the various IS the organization possesses along with their interaction with complementary business processes and assets (Woodard et al., 2013), making its value highly firm-specific (Teece, 1986). As other company-specific resources do, design capital may function as a major source of competitive advantage or disadvantage, depending on the extent to which the IT architecture and the related digital artifacts enable creation of economic value. Improving the state of a company's design capital brings it significant advantages such as ability to launch new digitally enabled products and services swiftly and at lower cost, react more rapidly to market opportunities and competition/threats, and more effectively shape the business ecosystems (Woodard et al., 2013). To assess the state of a company's design capital and gauge how deliberate actions may affect it, Woodard et al. (2013) utilize the concepts of digital options, technical debt, and design moves.

2.2.1. Digital options. New designs intrinsically hold economic "option value," which reflects both the direct value the product may bring and indirect value from enabling the creation of other designs (Baldwin & Clark, 2006). Therefore, one can regard digital options as a proxy for the value of the opportunities afforded by the organization's design capital. They are a means of preserving the chance to capitalize on a new technology or practice (Woodard et al., 2013).

Digital options play a crucial part in increasing organizational agility (Overby et al., 2006; Sambamurthy et al., 2003). Since an option reflects a right to invest later on without any obligation to make the full investment, digital options offer companies a degree of flexibility in their operations (Sambamurthy et al., 2003). While the company can capitalize on its digital options if the need arises, it is not bound by any decision in the present. The agility created through digital options allows companies to respond to changes in the market more swiftly and to gather and better interpret externally generated knowledge (Overby et al., 2006).

Prior research has identified four distinct areas wherein digital options are created: knowledge reach, knowledge richness, process reach, and process richness (Sambamurthy et al., 2003). The first involves the accessibility of codified knowledge available to a company. Well-architected IT systems can assist firms in accessing data from a wide range of sources (Overby et al., 2006). Knowledge richness, in turn, may be defined as the quality and timeliness of the accessible data (Evans & Wurster, 2000). Finally, process reach and process richness are largely analogous to these but tie in with business-process level: the former notion refers to how well various systems and stakeholders are mutually integrated, while the latter is a measure of the quality of the information at business processes' disposal. While some IT designs can improve both knowledge and process-related factors, most technologies tend to be either knowledge-oriented or process-related (Overby et al., 2006).

2.2.2. Technical debt. The second key concept refers to compromises, errors, lapses, and omissions arising during the design and implementation of software that unnecessarily increase its complexity (by introducing unwarranted dependencies) and thereby add to maintenance obligations (Cunningham, 1992). It can manifest itself through, for example, low-quality code, lack of documentation, or poorly structured IT architecture (Li et al., 2015). Technical debt accumulates over time as a natural byproduct of the design process, creating obligations that must be "repaid" when one strives to make new changes to the system. While it is often incurred inadvertently, especially when there is a shortage of resources, it can be taken on deliberately in a similar way as financial debt. An example of this would be using design shortcuts to allow for faster product launches.

Analogously to financial debt, technical debt can either create or destroy economic value, in line with how it is managed. Typically, companies with technical debt have two options for addressing it: they can continue paying the "interest" and endure its consequences or pay back the "principal" by reducing their debt via re-architecting (Brown et al., 2010). If the debt is handled in an appropriate manner, the short-term benefit gained from shortcuts during development can outweigh the maintenance costs and long-term drawbacks. On the other hand, technical debt tends to mount superadditively rather than linearly, which means that building up too much can drag systems into an irreparable state (Brown et al., 2010). Tackling technical debt can be rendered difficult by its largely invisible nature. Also, a large proportion of technical debt is related to structural/architectural choices or to technological gaps rather than the quality of the code itself. In most cases, it does not manifest itself in visible defects but, rather, creates barriers to future development (Kruchten et al., 2012).

2.2.3. Design moves. There is a strong interconnection between digital options and technical debt. For instance, taking design shortcuts to get access to digital options faster tends to increase technical debt. Maintaining access to digital options may come bundled with greater technical debt accumulation when it involves increasing IT maintenance obligations (Woodard et al., 2013). Then again, letting go of digital options (e.g., eliminating specific system functions) may decrease technical debt as there is less complexity and maintenance needs. Such actions, which may have increasing or decreasing impacts on digital options and/or technical debt, are referred to as design moves. One can plot the state of an organization's design capital on two dimensions (Figure 1) – digital options and technical debt – to form a 2×2 evaluation matrix (Woodard et al., 2013). The axes separating the quadrants run from high to low values for the two. The resulting "design capital map" is a simple but powerful analytic framework for assessing the effects of deliberate actions on an organization's design capital. Design moves may shift an organization from one quadrant to another, depending on the moves' implications of options and debt.

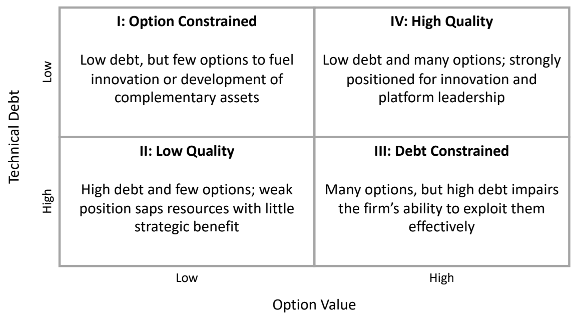
I: Option Constrained
Low debt, but few options to fuel innovation or development of complementary assets
IV: High Quality
Low debt and many options; strongly positioned for innovation and platform leadership
II: Low Quality
High debt and few options; weak position saps resources with little strategic benefit
III: Debt Constrained
Many options, but high debt impairs the firm's ability to exploit them effectively
Technical Debt
Low
High
Low
High
Option Value

Figure 1. Design-capital map.

3. The method – action research

To shed light on how the implementation of RPA to automate business processes affects a firm's design capital, we deemed action research a suitable approach, on account of its pragmatic and iterative nature (Baskerville & Myers, 2004). We were able to access a Finnish engineering company that had decided to configure an RPA tool to automate the process of validating outbound invoices. For our purposes, assessing design capital at the level of the company's IT infrastructure as a whole was infeasible, so we confined our analysis to the level of the company's customer relationship management (CRM) system. This system is used for the majority of the steps in the target business process.

In practice, the action research project reported upon here unfolded through four stages, from initial planning to the creation of a minimum viable product (MVP). The first stage (February 2021) consisted of establishing the requirements for RPA solution and scoping the possible RPA tools. The second (March 2021) entailed building the proof-of-concept (POC) version and reviewing and testing the results. Then, the third stage (April 2021) comprised building the pre-MVP version and validating the tool with real invoice data. The final stage (May 2021) consisted of finalizing the MVP version and probing areas for future development. The first author on the three-person research team was directly involved in the project as a customer-service coordinator and was the main person responsible for the task of configuring the RPA tool for the work process. The author took part in scoping the feasibility of using RPA for the business process, setting requirements for the software robot, and selecting the RPA vendor, along with the project team. In later stages, the author was responsible for configuring the software robot, developing the workflow, and initializing testing procedures.

The body of empirical data constituting the basis for the empirical narrative and subsequent analysis covers initial planning work, RPA development, and internal weekly meetings with project participants and other stakeholders. Further, in our design-moves analysis, we identified modifications which changed the company's design capital specific to their CRM system. When assessing the effects of design moves on options and debt, we drew on Woodard et al.'s (2013, p. 543) operationalization of the concepts. Regarding options, we identified the extent to which moves increased or decreased knowledge reach, knowledge richness, process reach, and process richness (Sambamurthy et al., 2003). Regarding debt, we probed the extent to which moves increased or decreased complexity and IT maintenance obligations.

4. The empirical narrative

We present the empirical narrative via the four above-mentioned stages of the action research project: planning, POC, pre-MVP, and final MVP.

4.1. Planning

The planning process, constituting the first stage in the action research process, began in February 2021, when the idea of using RPA to validate the information related to outbound bills was proposed. Flexibility, ease of use, price, and reconfigurability formed the reasons for choosing RPA as a potentially good method for this task.

The first step in planning for development of the RPA implementation was to assess how RPA was going to be used, how it would affect the workflow, and its limitations. Three key criteria were set for the performance of the RPA solution: 1) it must be able to save on human labor in a cost-efficient way, 2) its performance should be somewhat comparable to a human worker's, 3) a low error rate was to be prioritized over extensive automation, and 4) the solution should not need extensive post-implementation maintenance.

The next step in the planning was to prepare a detailed description of the billing process. This aided in creating an overview of the process, assessing what parts of it could be automated, and addressing potential concerns. The process was articulated via a flowchart-like structure, which formed a good basis for initiating automation. This stage identified validation of contact information as the key subprocess for automation of the billing workflow. This subprocess relied partly on external data, some not accessible via APIs; hence, RPA was envisioned as serving the retrieval of some data from the company-internal information system that were not readily available in the desired form from the existing reporting tool. In addition, some types of engineering inspection require submitting an official document on the inspection to the customer's contact point alongside the invoice, with most of these documents being emailed to a property manager. The goal for RPA in this subprocess was to largely mimic the workflow of manual validation. In the latter flow, firstly, the information about property management is checked against both the customer information stored in the CRM tool and a Web service of the Finnish Patent and Registration Office. This information gets compared, further, to the contact information in the work order. If all these sources match, the contact information is considered valid; otherwise, closer manual inspection is needed. An additional challenge with this subprocess stemmed from the fact that not all data consulted shared the same format. The property-management information from the customer data and the external source used the property manager's name, while the contact information was retained in the form of an email address. Also, minor variations in the person's name, such as abbreviations or corporate identity markers, were common. This meant that successful automatic matching across the data sources required additional steps.

4.2. Building the POC

Building of test versions of the software robot began in March 2021. This marks the start to the second stage in the action research process. Initial trials employed Python's RPA library (TagUI), but this was quickly ascertained to be a non-ideal solution for the final implementation, for performance and maintenance reasons. The library had trouble accessing some data sources, execution was fairly slow, and the code would not have been the easiest to maintain. Therefore, the staff decided that the development should utilize a trial version of a commercial RPA tool – if the project did not succeed, no tool costs would have arisen. For this, the project personnel chose the vendor UiPath, the market leader for RPA software and generally regarded as providing powerful, user-friendly solutions. It also is among the lowest-cost options for software licensing.

The first part of the testing focused on collecting information from the various data sources automatically. As noted above, the software robot had to consider two sources, the Finnish Patent and Registration Office's free-to-use service YTJ, containing basic data on Finnish enterprises, and the case company's CRM system. Obtaining data from either of these required its own combination of actions, but collecting the data from both was determined to be possible. The approach chosen for data retrieval was to store the search terms used to obtain the data in an Excel-file column, then let the software robot retrieve the information and add it to other columns in the same file. Then, the data would get processed Excel-internally for indication of which bills have valid contact and billing information. Excel was chosen for this task for three reasons: many of the invoice-related data items could be easily pulled into Excel files from the company's internal systems, the RPA software interfaced well with it, and this seemed to be the option affording the simplest maintenance. If everything matched on the file, the tool would mark a bill ready for automatic sending.

Once these sub-procedures were created in a UiPath environment, testing in a sandboxed version of the CRM system was undertaken, to avoid any chance of unintended actions by the software robot affecting live billing data. Sandboxing is designed precisely for this type of testing, to prevent any of the actions influencing data in the production environment. The result of the testing was a software robot capable of collecting all the necessary data and navigating the CRM system to mark the validation checkbox. At this stage, the software robot could be characterized as a POC version: it had reached the internal testing phase but not yet been tested beyond the project team. This version was largely unoptimized and still required a large amount of manual work.

The testing involved using unvalidated bills, for which contact data were collected and mapped to a property manager, with the results later getting cross-referenced with the bills validated by Customer Service. Whenever errors or edge cases were detected, they were noted for future development attention. Also, the POC version's performance was tested against known exceptional cases, for seeing how it would perform in certain tricky situations.

Another key element in the testing was to make sure the RPA could not perform unintended actions when left unmonitored. This was especially important with regard to the company's CRM system, where an edge case or bug could have led to considerable damage (e.g., accidental deletion of outbound bills). Therefore, the developers set out to create a separate account for the software robot, with restricted system access: it would have read-only access to everything but the single specified checkbox.

Overall, the results from testing of the POC version were positive. The software robot was found capable of collecting the necessary data and could handle exceptions reasonably well. One of the reasons behind the attention to exception-handling capabilities was the strict requirements for when a validation checkmark could be given. The software robot also was observed to not perform any unintended actions.

4.3. Building the pre-MVP

With the POC version's testing complete, the software robot entered further development in preparation for testing with the ultimate end users in Customer Service. This development included combining the sub-procedures into a single entity, creating logic to handle known exceptions, simplifying the software robot's workflow, and making various minor optimizations. The resulting version of the software robot was much more fully optimized and had a better-defined automation workflow. It can be viewed as a pre-MVP version.

In its testing with Customer Service and a live version of the CRM system, the pre-MVP version used unvalidated bills, as the POC version had, but now customer-service coordinators were presented with the system's proposed checkmark when validating a bill. A coordinator who saw a checkmark and identified the information as incorrectly validated would assign the bill a keyword; otherwise, human validation of a bill that had a checkmark was seen as indicating a correct validation decision. Additional information was gathered via periodic collection of feedback from customer-service workers about their impressions of the software robot's performance.

The results from the pre-MVP version's testing too were found to be positive, with only a few edge cases getting identified. Addressing these proved to be relatively straightforward. Upon completion of the pre-MVP testing stage, the software robot was determined to have an adequate feature set for forming a successful automation workflow. Thus, it reached MVP stage.

4.4. Configuration of the MVP

Throughout the MVP artifact's development process, the progress of the software robot was tracked via project-team meetings held 2–3 times per month. At these meetings, personnel evaluated the current state of the software robot, presented new ideas, and addressed any potential issues identified. Depending on the progress and the possibilities and challenges that emerged, the initial plans for the software robot's configuration were adjusted to better reflect the current state of the software robot.

The MVP artifact's configuration was broken into four distinct steps. The first pulled a report containing the information on outbound bills in the customer-service queue from the company's CRM system. This report featured the bill's unique identity code, the corporate identifier of the customer billed, and the contact information from the customer's billing-contact details (again, for a property manager in most cases). These data would get copied to a standard-form Excel sheet integrated with the UiPath solution. Using said details, the UiPath system would search for and return a corresponding "c/o" address from the YTJ service. This step's speed was roughly 20 searches a minute. After this, the software robot would search the company's CRM system for contact information matching each bill-identifier code (the system structure did not allow retrieving the contact information in report form). The contact details included the email address of the person receiving the documentation, the company where the recipient works, that person's inspection-related role (e.g., property manager or maintenance person), and the email address for the contact persons associated with the location of the inspection. UiPath's solution automatically inserted these data in the corresponding columns in the Excel file. While this step was relatively slow, at 6 searches/minute, it was still many times human speed.

Recall, however, that the CRM data gathered for the Excel file are not in the same form as the data pulled from YTJ; as described above, some of the data related to contact points were expressed as "c/o" names and some as email addresses. For matching the two types of data with each other, a lookup table was created to map each "c/o" address to a corresponding email-address domain. Next was a check of whether the corresponding domain matches the email addresses for the recipients and the location's contact persons. If both lists of email addresses contained the relevant domain and if the internally and externally fetched "c/o" addresses mapped to the same domain, the contact information was regarded to be valid.

The lookup table contained "c/o" addresses of property-management companies and each one's corresponding email-domain name. Because the data often featured slight variations in the "c/o" addresses (abbreviations etc.), sometimes multiple entries had to be made for a single property manager. Though built manually, the table was amenable to constant reuse, and it permitted small updates and other necessary changes. Since including all possible versions of every property manager's "c/o" address was impossible, sometimes the domain matching the given "c/o" address could not be found. However, roughly 90% of the "c/o" addresses could be matched to a domain, with this number naturally increasing further with updates to the table and as it grows from its initial 500 rows. Those cases wherein a match is impossible would simply be directed to manual checking, so the only disadvantage should be a slight decrease in the percentage of automatic validation.

This solution, while less elegant than writing a special program to handle fuzzy data-matching, was deemed preferable for the time being. While a custom program could have led to a higher rate of automatic handling via better matching of data, building such a system from scratch could well have taken significantly more time, and the solution would have been harder to maintain. Nonetheless, it was discussed as a possibility for future improvement to the software robot's workflow.

Because a low error rate for the RPA solution was crucial, bills were to get marked for automatic sending only if all the information, from different sources, matches. Bills not reaching this threshold should be handled manually. Channeling them into the same flow as before was a way to make sure the probability of errors would be small. Naturally, conservatism presented the downside of a lower automatic-processing rate, with fewer bills being eligible for automation. The requirements could be gradually relaxed, however, if error rates remained low, thus allowing for higher percentages later.

The final step for the software robot was to mark the checkbox in the corresponding work order in the company-internal system. The software robot would read the Excel file row by row, marking the box and clicking `Save` if the contact-information check yielded a Boolean value of `TRUE`. This checkbox could then inform the workflow in the CRM system such that those bills with a checkmark need not undergo validation by Customer Service.

5. RPA's effects on design capital

Against the background of detailed description of the RPA tool's development and configuration, we can now discuss its effects on the organization's design capital. Firstly, we set the stage related to the initial state of design capital, after which we provide an account of the effects on digital options and technical debt found. We then elaborate on how the design capital developed in the course of the project, before, finally, reflecting on the MVP's fulfillment of the initial requirements from a design-capital angle.

5.1. The initial state of design capital

We confined our analysis to the level of the company's CRM system which is used for the majority of the steps in the target business process. We found no strong interdependencies with other internal systems to affect the business process scrutinized so one can regard the CRM system as operating largely as an independent entity. Although one of the biggest providers in the marketplace supplied the CRM system and it features some advanced automation capabilities, integration with many external sources is not fully supported. This led to a need for repetitive manual work in many processes that rely on external input. Limits arose, accordingly, in how quickly material such as outbound bills could be processed. Therefore, in this regard the system can be described as featuring a relatively low level of digital options. The amount of technical debt, in turn, was considered manageable in the CRM system. Debt was present in some designs but did not greatly affect regular use. Hence, we deemed the initial state of design capital as option constrained.

The MVP project work identified RPA's clear potential to create new digital options while managing technical debt by addressing the four key criteria set for the project (see 4.1. Planning). The options would arise from enabling a previously manual process to be automated. The first criterion related to cost-efficiency is tied to an increase in digital options brought by allowing the company to save costs by increasing its level of automation. Meeting the second criterion (RPA's performance must be comparable to a human worker), in turn, guarantees maintaining incumbent digital options by minimizing the lost value derived from suboptimal process performance. The third criterion directs the types of digital options to be created by prioritizing error prevention over efficiency. Finally, setting the fourth criterion (the solution cannot require too much maintenance after implementation) helps make sure that the level of technical debt stays manageable. Since any technical debt created by RPA would be largely siloed in the functions associated with the billing workflow, the expected effect on debt was deemed reasonable.

5.2. Design moves during the project

Next, we track the changes in the company's design capital through three design moves reflecting the introductions of different versions of the RPA solution: the POC, pre-MVP, and MVP artifact. Each version involved a series of smaller interventions and changes which cumulatively constitute a design move.

Firstly, as discussed above, the initial design capital was in an option-constrained state, with low levels of both digital options and technical debt. From here, the development of the POC significantly increased technical debt, in that the solution was largely unoptimized and increased complexity was introduced to the company's IT architecture, gets introduced here. This change was coupled with an increase in digital options through improvements to knowledge reach and knowledge richness. Increased knowledge reach stemmed from allowing the company's systems to access to access the external (YTJ) data automatically. The improved reach allowed customer data to be enriched by means of external sources, hence resulting in higher data quality and greater knowledge richness. This move drew the design capital temporarily into the low-quality quadrant.

Second, as the system entered the pre-MVP stage, the more robust and streamlined workflow, the software robot's better exception-handling capabilities, and resolution of the issues identified in the POC stage mitigated a portion of the technical debt. Also, the digital options increased from unlocking access to higher-quality data for the billing process, thus enhancing process richness. The RPA enabled data to be both extracted from the CRM system and structured in ways not possible with the previous, standard configurations.

Finally, the MVP version moved the design capital to a debt-constrained state via slight net increases in both digital options and technical debt. Increases in digital options and decreases in technical debt stemmed from improving the software robot's usability and addressing the issues encountered in the pre-MVP stage, respectively. However, most of the technical debt remained unaddressed. For example, the daily process in this configuration still was triggered manually on a desktop computer instead of via the desired automatic, timed operation. The workstation involved was virtually unusable at runtime, since the process would fail were another program to be opened in addition to UiPath and the Web browser it was using. Also, the Excel configuration for mapping property managers to "c/o" details featured a highly customized *ad hoc* setup. In the lookup process matching the "c/o" data from the external source to the property managers, the most common variations of the property managers' "c/o" names were checked against existing values. A process of this type is difficult to maintain in that a property manager's name may change and new ones may enter the market. There was a possibility of the process yielding a lower percentage of automatic processing through "c/o" values not being recognized as belonging to the relevant property manager.

In sum, the design-capital map in Figure 2 represents a path from an initial option-constrained state toward the middle of the matrix through the low-quality state. Assessing the overall change in design capital features many subjective elements, but one can conclude that the digital options created here should outweigh the downside of the technical debt.

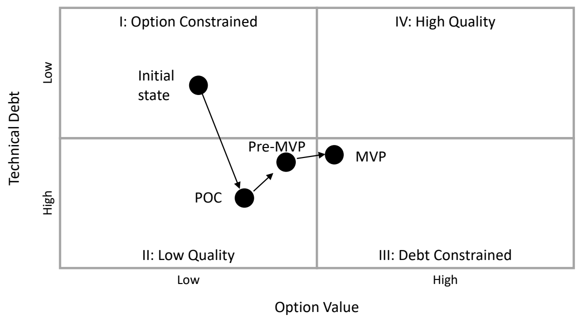

Figure 2. RPA's effects on CRM system's design capital.

5.3. Reflections on digital options and technical debt

Overall, the MVP was found to comply well with the initial project requirements for the software robot. It could handle the validation of billing information for roughly 30% of the outbound bills. This represents significant potential to reduce the amount of manual work related to their validation process. The speed of execution was also more than sufficient for the number of bills sent out: the software robot could clearly outperform a human worker in speed. Therefore, the digital options referred to in the first and second criteria could be realized. Next, analysis found the error rate to be low. During development, no cases were observed wherein the RPA tool would have marked the validation checkbox for any bill that, per the logic of the automation workflow, should not have had one. That workflow logic also seemed to function well overall in dividing the bills between fully automatable and non-automatable cases based on validation. That said, the number of invoices used in testing was somewhat low, totaling about a thousand, so these observations must be taken with a degree of caution. There could always be undiscovered edge cases that might result in unforeseen behavior of the software robot and the workflow. The same uncertainties apply to any claims about maintenance requirements after implementation. Hence, the constraint set by the third criterion is satisfied while analysis related to the fourth criterion is inconclusive.

RPA represented a potentially powerful method of accelerating digital transformation within the company. The software robot could allow the company to digitalize the remaining manual tasks in the billing workflow for a large number of cases and offers potential to automatically generate data on the billing process – data that can inform improvements to both the process and the software robot. The audit trail created could be used, for example, aid in identifying edge cases and monitoring performance.

Adopting RPA as a part of the organization's IT architecture has rendered the company able to utilize RPA for other use cases in a more agile manner, in that it already supports the tool and can benefit from the initial implementation. Indeed, several potential cases for using automation in comparable business processes were discussed during the project. In such a manner, implementing RPA as a new tool showed potential to create digital options outside the context of the billing workflow alone. These options could be considered as "carryover" from the implementation.

Many of the digital options created can be characterized as unique to RPA in relation to other, comparable technologies (such as business-process management tools). Among them are the ability to access data sources that lack an API and the possibility of quickly reconfiguring the tool to alter the workflow of the automation process. The project team considered these to be major upsides to RPA.

A few sources of technical debt were identified. The most noteworthy ones lay in the addition of a new technology requiring support. This inherently brings more complexity to the IT architecture and requires the company to have an employee with the associated understanding and skills in (re)configuring the software robot. In addition to this, daily processing via RPA requires creating and maintaining a dedicated process.

Another potential source of future technical debt is RPA's low level of technical robustness. This may lead to an increased need for maintenance work and reconfiguration. For instance, changes to the user interface of the external data source could result in the software robot being unable to access the required data. Thanks to RPA's flexible nature, typically one should be able to reconfigure it fairly quickly, but the total amount of maintenance work could still amount to something significant over time.

At the same time, participants found factors helping to mitigate technical debt. Because RPA operates on top of the CRM system rather than being deeply integrated with it, relatively little technical debt gets transferred between RPA and the CRM system. That is, the technical debt related to the CRM system does not strongly affect the RPA's operation, and *vice versa*. This was evident in the RPA's ability to collect and structure any combination of data items, from across disparate parts of the CRM system, by simply navigating the user interface. Replicating this within the CRM system itself would have likely required a significant amount of back-end configuration work. Also, again, the technical debt created by RPA could be largely isolated in the CRM-system functions associated with the billing workflow.

One additional element that might serve to mitigate the effects of technical debt is the low amount of organizational reliance on the software robot and remaining flexible for contingency strategies in case of downtime. Since human workers can handle the process if the software robot breaks down, the risk associated with low technical robustness is small. Assuming the workers manage to maintain the required skills (see Rinta-Kahila et al., 2018), brief downtime would produce only minor losses in productivity and a need for slightly more work hours and time for processing outbound bills. This reduces the cost of paying "interest" related to technical debt.

6. Conclusions

Prior work has looked into the development of design capital in the context of IT product management (Woodard et al., 2013) and heavyweight enterprise system implementations and replacements (Rinta-Kahila et al., forthcoming; Rolland et al., 2018). Our study investigated a less-explored situation of lightweight IT development by probing into RPA's effect on design capital. RPA was found to have a clear and distinct effect on both digital options and technical debt at the case organization. Firstly, option-creating effects tie in with RPA's potential to create digital options by allowing a company to improve its level of digitalization; increase knowledge reach, process richness, and knowledge richness; and exploit the organizational learning that occurred during implementation. Such fine-grained consideration of digital options expands on previous design capital studies which have given less attention to different forms and functions of digital options (e.g., Rinta-Kahila et al., forthcoming; Woodard et al., 2013). Secondly, the notion of debt-creating effects helped us articulate the company's level of technical debt increasing through the implementation of RPA. Such change takes place mostly because of the accompanying increase in complexity of IT infrastructure and, in the case company, RPA's low technical robustness. Debt-mitigating effects help to offset some of the technical debt created during the implementation. In our study, among these were the low transferal of technical debt between RPA and other systems and the mitigating effects of maintenance, optimizations, and future development. While our study did not conceptually separate different types of technical debt, future studies can enrich this understanding by considering how different debt types (e.g., code, architecture, documentation, etc.) might interact with different types of options (e.g., knowledge reach/richness).

Practitioners can benefit from these findings in their evaluations of RPA's effects on design capital via sensitization and by identifying ways to maximize the creation of digital options and address technical debt in advance. By accounting for changes in design capital and acknowledging the relevant causal relationships, practitioners can optimize the value that RPA brings to the organization while limiting the associated downsides. They can reap benefits from RPA's flexibility and rapid implementation, to gain from the ensuing digital options across multiple use cases in a more agile manner.

7. References

Aguirre, S., & Rodriguez, A. (2017). Automation of a business process using robotic process automation (RPA): A case study. *Communications in Computer and Information Science*, *742*.

Asatiani, A., Copeland, O., & Penttinen, E. (forthcoming). Deciding on the robotic process automation operating model: A checklist for RPA managers. *Business Horizons*.

Asatiani, Aleksandre, & Penttinen, E. (2016). Turning robotic process automation into commercial success – Case OpusCapita. *Journal of Information Technology Teaching Cases*, *6*(2), 67–74.

Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems. *Understanding Complex Systems*, *2006*.

Baskerville, R., & Myers, M. (2004). Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice: Foreword. *MIS Quarterly*, *28*(3), 329–335.

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.

Bygstad, B. (2016). Generative Innovation: A Comparison of Lightweight and Heavyweight IT. *Journal of Information Technology*, *32*(2), 180–193.

Cooper, L. A., Holderness, D. K., Sorensen, T. L., & Wood, D. A. (2019). Robotic process automation in public accounting. *Accounting Horizons*, *33*(4), 15–35.

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

Evans, P. B., & Wurster, T. S. (2000). *Blown to Bits: How the New Economics of Information Transforms Strategy*. Harvard Business School Press.

Hofmann, P., Samp, C., & Urbach, N. (2019). Robotic Process Automation. *Electronic Markets*, *30*(1), 99–106.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software*, *29*(6), 18–21.

Lacity, M. C., & Willcocks, L. P. (2016). Robotic process automation at telefónica O2. *MIS Quarterly Executive*, *15*(1), 21–35.

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.

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.

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.

Rinta-Kahila, T., Penttinen, E., & Lyytinen, K. (forthcoming). Getting Trapped in Technical Debt: Socio-Technical Analysis of a Legacy System's Replacement. *MIS Quarterly*.

Rinta-Kahila, T., Penttinen, E., Salovaara, A., & Soliman, W. (2018). Consequences of Discontinuing Knowledge Work Automation – Surfacing of Deskilling Effects and Methods of Recovery. *Proceedings of the 51st Hawaii International Conference on System Sciences*, 5244–5253.

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.

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.

Teece, D. J. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, *15*(6), 285–305.

Tornbohm, C., & Dunie, R. (2017). *Market Guide for Robotic Process Automation Software*.

van der Aalst, W. M. P., Bichler, M., & Heinzl, A. (2018). Robotic Process Automation. *Business and Information Systems Engineering*, *60*(4), 269–272.

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.