MIS Quarterly

RESEARCH ARTICLE

ACHIEVING EFFECTIVE USE WHEN DIGITALIZING WORK: THE ROLE OF REPRESENTATIONAL COMPLEXITY[1]

Jens Lauterbach
effectivIS GmbH,
Munich, GERMANY {lauterbach@effectivis.de}

Benjamin Mueller
Faculty of Business and Economics (HEC), University of Lausanne,
Lausanne, SWITZERLAND {benjamin.mueller@unil.ch}

Felix Kahrau
KMU sicher digital,
Venningen, GERMANY {kahrau@kmusicherdigital.de}

Alexander Maedche
Institute of Information Systems and Marketing, Department of Economics and Management,
Karlsruhe Institute of Technology, Karlsruhe, GERMANY {alexander.maedche@kit.edu}

In times of accelerated digital transformation, many organizations still struggle to put enterprise systems to effective use quickly. While prior work suggests either system or task complexity as a source for these difficulties, this case study of a major system implementation at a European bank reveals the most important source to be the complexity arising from co-dependency between the system and the task. We conceptualize this co-dependency as inherent in system-enabled tasks by proposing system dependency (the extent to which a task is supported by a system) and semantic dependency (the degree to which semantic understanding is required for task completion). Together, these dependencies create representational complexity, which constrains users from achieving effective use in system-enabled tasks and can explain differences in achieving effective use through variations in learning effort. The concepts and insights emerging from this study provide researchers and practitioners with a deeper understanding of what complexity means and why, in some contexts, learning how to use systems effectively takes longer.

Keywords: Effective use, complexity, representation theory, system dependency, semantic dependency, exerting representational complexity, case study, critical realism, representation analysis

Introduction

Fast achievement of effective use soon after the implementation of enterprise systems is essential if organizations are to realize the relevant system benefits (Burton-Jones and Grange 2013). However, many organizations struggle to attain this effectiveness. To investigate why, we conducted an in-depth case study to learn how and why differences in achieving effective use materialize following a large-scale system implementation.

In our case, we observed three departments within an organization that were provided with the same system. In one department users achieved effective use quickly, whereas in the other two departments users struggled with this challenge

[1]Andrew Burton-Jones was the accepting senior editor for this paper.

for much longer. To explain this discrepancy we turned to the literature on effective use, which suggests that completing a task effectively requires learning to apply the system effectively (Burton-Jones and Grange 2013). However, as we will show, differences in achieving effective use occurred despite identical training and support in the three departments. We also found that extant conceptualizations of learning effective use could not explain our findings.

On closer inspection, we found that the missing piece to our puzzle was complexity. Complexity has long been a fundamental concept in both the Information Systems and Organization Sciences disciplines, but the two disciplines have largely focused independently on either the complexity of systems (e.g., Beese et al. 2016; Liang et al. 2015; Sharma and Yetton 2007; Volkoff et al. 2007) or the complexity of tasks (e.g., Campbell 1988; Wood 1986). On the Information Systems side, for instance, Brooks (1987, p. 11) stresses that "the complexity of software is an essential property, not an accidental one .... From this essential complexity … comes the difficulty of invoking function, which makes programs hard to use." However, our in-depth case study suggests that an independent treatment of the two in the two disciplines is insufficient. Rather, the co-dependency between task and system is critical, a finding supported by recent literature (Hærem et al. 2015).

New concepts are needed to further this line of thought. We propose *system dependency* and *semantic dependency* and use these novel concepts to describe a mechanism we call *exerting representational complexity* in use. As we will outline, this mechanism explains how and why one department achieved effective use more quickly than the other two. Our results contribute to the literature on effective use and bring together the literatures on task and system complexity. Further, our concepts offer managers insight into how to improve training and support to address the complexity of system-enabled tasks so as to advance individuals' ability to quickly master the use of new systems and facilitate firms' digitalization efforts.

As we proceed, the paper is structured as follows. We begin by presenting our study's conceptual foundations. These foundations link our work to the literature on system use and effective use and serve as a scaffold for our analysis of the case data. We then introduce our case study approach to develop the theoretical explanations for the differences we observed in achieving effective use. We then extract our new concepts from the case data and use these concepts to extend our conceptual framework. Finally, we discuss our results and their implications for theory and practice.

A Framework for Understanding Effective Use and Complexity

We made two decisions early in the study that influenced how we viewed effective use and complexity. First, we decided that to understand *how and why* users vary in how quickly they achieve effective use, we needed to focus on the underlying *mechanisms* involved. Researchers can study mechanisms in various ways (Avgerou 2013). We chose to take a critical realist approach given that it focuses extensively on mechanisms (Volkoff and Strong 2013) and because researchers studying effective use have used it in the past (Burton-Jones and Grange 2013; Burton-Jones and Volkoff 2017). Second, given that a theory of effective use had been published shortly before we began the study (Burton-Jones and Grange 2013), we decided to see if those ideas could help us in our study. The result of these two decisions was that we began the study with a range of concepts from critical realism and from the recent theory of effective use. Accordingly, our first step was to draw these concepts together into a conceptual framework that could guide our analysis. We do not propose that our conceptual framework offers the only perspective available; other researchers might well draw on different literatures or use the same literatures as us in different ways. Nonetheless, the conceptual framework we developed offers one approach that we believe can work.[2]

At its core, our framework is built around the conceptual entities that form system use (Burton-Jones and Straub 2006). We refer to these conceptual entities as structures (Sayer 1992, p. 92). As we will outline later, the relationships and interactions between these structures allow us to extract mechanisms from our case data, which provide the theoretical explanations we are looking for (Archer 1995; Bhaskar 2008). Below, we first define system use and its effectiveness as a basis for extracting the structures at the core of our explanations. We then describe two main components of use in detail— system and task—and discuss how the user interacts with these in learning effective use. As a final point, we establish our view on the role of complexity in our framework.

System Use and its Effectiveness

Effective use has its roots in definitions of system use. System use is the extent to which users employ a system to carry out a task (Burton-Jones and Straub 2006). Accordingly, we conceptualize *use* as a structure in which user, task, and system interact to achieve a relevant goal (Burton-Jones and Straub 2006).

[2]Given that the focus of this study is theoretical rather than philosophical, we do not discuss our critical realist perspective extensively in the paper.

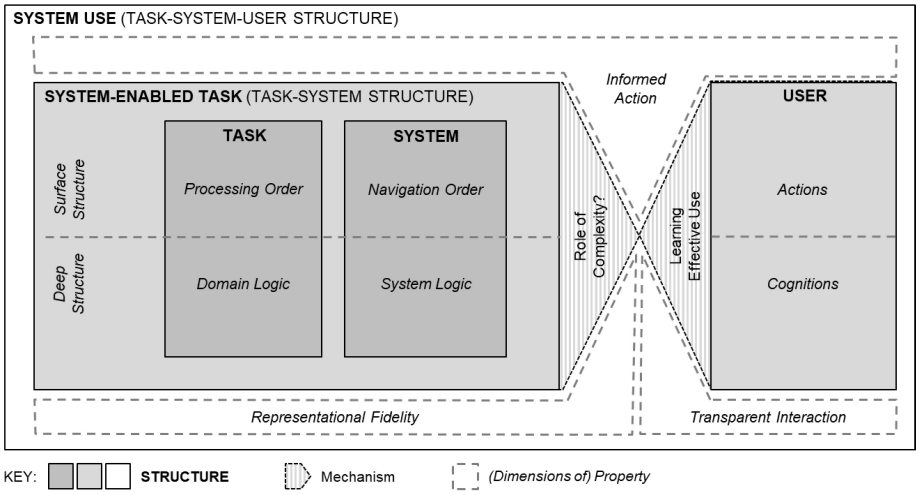

Figure 1. The Role of System-Enabled Tasks in Achieving Effective Use

Use becomes effective when users employ a system "in a way that helps achieve the goals for using the system" (Burton-Jones and Grange 2013, p. 633). In business, use becomes effective once users can, by using the system, complete their task and achieve the associated business goals. Effectiveness of use is not simply inherited from the component parts. Rather it arises as a property from a process in which system, task, and user interact to generate use (Elder-Vass 2005; Mingers and Standing 2017). Understanding of this process depends on comprehending the structures that constitute use and their interactions. Figure 1 provides an overview of the relevant structures and how their interactions generate use.

Surface and Deep Structures

There are a number of ways that researchers might conceive of these structures and their interactions. We draw on representation theory (Wand and Weber 1995), which is also at the heart of the effective use concept as proposed by Burton-Jones and Grange (2013). However representation theory is mainly concerned with the system structure in our framework; it says little-to-nothing about task structure. We thus provide a detailed discussion of this issue below in order to provide the conceptual basis needed for developing a matching task conceptualization later. The level of detail that follows helps us ensure that the contribution we seek to make is clear.

Following representation theory, we assume that an information *system* consists of surface, deep, and physical structures (Wand and Weber 1995). On an abstract level, surface structures contain syntax while deep structures carry semantic meaning (Wand and Weber 1995). In information systems, surface structures are the interactive interface of the system whereas deep structures are the actual meaning of a real-world entity represented in the system (Wand and Weber 1995). System surface structures confront users with what we define as system *navigation order*, which is composed of entities visible to the user and the logical relationships among them (e.g., in digitized forms). The navigation order prescribes users' interaction with the system during use. At the deep structure level, we propose a corresponding concept—*system logic*—defined as how users are provided with system functionality through the functions (algorithms) and data objects implemented in the system. The key point for our paper is that learning effective use requires users to master both navigation order and system logic (Burton-Jones and Grange 2013).[3]

With respect to the *task*, representation theory states that a real-world phenomenon such as a task can be represented in an information system, but the theory does not provide an exact conceptualization of task structures nor does it describe how they can be represented in a system. We argue that a

[3]These concepts of navigation order and system logic, and the associated concepts of processing order and domain logic in Figure 1, are not existing concepts in representation theory. Rather, they are proposed here. As readers of representation theory will know, the theory does not provide many concepts for studying the use of information systems in practice (see Burton-Jones et al. 2017). The concepts and approach we offer in this paper, therefore, reflect contributions to representation theory and its use in IS research. The same applies to our analysis approach later in the paper (see Table 3). We return to these contributions to research using representation theory in our "Discussion" section.

better understanding of what a user does while using a system for a task requires understanding how a task is structured and how the system represents it.

The idea of matching structures between tasks and systems has been acknowledged in IS research for a while. For example, in his editorial introduction to DeSanctis and Poole (1994), Zmud stressed the existence of deep structures in both the technology and the work environment. However, the implications of this have not been acted upon by researchers in detail yet. While a small number of studies have dealt with more general organizational (deep) structures (e.g., Leiser and Hirschheim 2007), these studies do not discuss deep structures on a task level. Others mention task (deep) structures (e.g., Burton-Jones and Straub 2006), but do not explicitly conceptualize them or relate them to system (deep) structures. We therefore developed our own task conceptualization, building on basic task components and properties presented in the literature (Campbell 1988; Goodhue and Thompson 1995; Hackman 1969). In particular, a task creates outputs from inputs by following a sequence of certain acts supplemented by information cues, which are facts task doers process to make judgments during their task (Wood 1986).

Linking this conceptualization to representation theory, we conceptualize tasks as consisting of surface and deep structures. The task surface structure is the task syntax. The required elements (like symbols or letters in a language) are information cues, acts or activities, and inputs/outputs. Like syntax in language, their sequence must be well defined to make sense. As indicated by Wood's (1986) task definition, information cues have to order the task acts into an adequate sequence. This ordering results in the *processing order* as a task surface structure. Similarly, the task deep structure *domain logic* contains information about what distinct acts mean in a given context and how they depend on one another to achieve the overarching task outcome. Here, information cues serve as the container of semantics giving meaning (not just order) to the activity.

Learning Effective Use

Having discussed the structures of use, we now turn to the issue of learning how to effectively use the system. This allows us to integrate the user as the third key constituent of use. Relying on task and system as key structures of use, we conceptualize the process of *learning effective use* of a system for a task in three stages, with one stage for each of the dimensions of effective use, that is, transparent interaction, representational fidelity, and informed action (Burton-Jones and Grange 2013).

First, users learn to interact with the system by getting to know its surface and physical structures. This learning gradually improves users' *transparent interaction* with the system in use, which ensures that users can access the system representations unimpeded by the surface and physical structures of the system (Burton-Jones and Grange 2013). Second, users' understanding of the system improves with respect to what the system does and how it works. Over time, users advance their level of *representational fidelity*, which refers to "how well the representations the user obtains from the system faithfully reflect the domain" (Burton-Jones and Grange 2013, p. 642). As faithful reflections "provide a more informed basis for actions" (Burton-Jones and Grange 2013, p. 636), users can apply their understanding of the system to the task and learn how to leverage its representations. Third, users can now bolster their ability to take *informed action*, which is the extent to which users achieve the goals for using the system (Burton-Jones and Grange 2013). In business, users' goals are linked to the performance criteria for their tasks and roles. For instance, to determine clerks' effectiveness, managers could measure the number of records a clerk entered into a system.

Against this backdrop, our task is to discern the causal mechanism through which, over time, a level of effectiveness emerges that can objectively be classified as "effective use." In our framework, the three stages described above form the mechanism *learning effective use*, which generates effectiveness as a property of use (Danermark et al. 2002; Mingers and Standing 2017; Sayer 1992). When learning effective use, users get to know the system (navigation order and system logic) and the domain it represents (processing order and domain logic), get to improve their ability to obtain faithful representations of that domain from the system, and get to know how to leverage those representations for informed actions (Burton-Jones and Grange 2013). Through the process constituted by these learning actions, users become familiar with not only the system and the task but also the co-dependency of the two structures.

The Role of Complexity

As users interact with the system to take concrete actions to perform a task, they also create abstract cognitive conceptualizations of these experiences, the underlying task and system structures, and their relationships (Kolb and Kolb 2009). This is where we suggest that complexity plays a role. When learning effective use, especially when learning how to achieve representational fidelity, users have to map the system and the task to one another to understand the system–task co-dependency. In our conceptual framework, we introduce the

system-enabled task structure to capture this co-dependency. In our conceptualization it is a structure just like the system or the task, but this structure emerges from the relationship between the task and the system. We suggest that the entities and properties of the system-enabled task create another mechanism—*exerting complexity*—which influences the cognitive demands that task-doers experience as complexity. This complexity has a counteracting effect on learning effective use. As we will show later, this insight was central to understanding what occurred in our case study.

Methodology

With the framework in hand, we now introduce our case and discuss how we collected and analyzed our data.

Case Description

Our case site is the retail banking division of BANK (pseudonym), a global financial services provider with roots in Europe that undertook a multi-year system-enabled transformation program. The banking industry is particularly suitable for studies such as ours because the information systems are tightly interwoven with almost every task.

BANK's transformation program replaced its custom-built, core banking system and the surrounding systems with an integrated standard software solution. The new solution was rolled out in several releases, one of which concerned BANK's credit service unit, Credit Factory (CF). CF is a shared service center that supports BANK's branches with the post-processing of credit business, such as mortgage loans. At CF, three departments are responsible for carrying out the two core tasks of credit processing (Table 1).

These departments suit our research well because they are three different groups of loan management system business users whose individual members cover the two main tasks of post-processing credit business. PSN is responsible for opening new contracts and entering all relevant data into the system; PSS processes all changes to contracts in stock. All users within these two departments have the same task. This within-group homogeneity and cross-group heterogeneity allows the differences between the groups (i.e., in terms of their tasks) to contribute to an explanation of the observed outcomes (Klein et al. 1994). Since SSG carries out both tasks, we are able to obtain corroborating evidence because if observations from PSN and PSS also occur at SSG, we can rule out non-task-related departmental characteristics as contributing to an explanation for the differences we observe.

BANK's transformation program included implementing a new loan management system (LMS) across all three departments. LMS is a banking-specific solution provided by an international software vendor and is a customizable standard software integrating information and business processes (e.g., across units within CF and linking them to BANK's branch employees). The old loan management system (OLMS) had been in place for over 30 years, and the introduction of the new LMS led to the typical scope and complexity of enterprise system projects (Devadoss and Pan 2007; Markus and Tanis 2000). Although all CF clerks could use OLMS effectively, its maintenance and scalability had become cumbersome. New regulatory requirements, which had become mandatory after the 2008 financial crisis, were almost impossible to implement.

Project activities to replace OLMS and implement LMS at CF started early in 2013 with requirements engineering, followed by configuration, implementation, and testing, and lasted until November 2013. Change management and training took place from September to December 2013, when the legacy data migration occurred, with the final release of LMS in early January 2014.

Data Collection

We used multiple data collection methods to enable triangulation and generate the data needed to address our research objective. Analysis of documents (e.g., project documentation, intranet pages, organizational charts, role and process descriptions) helped identify the initial topics and relevant stakeholders across all organizational levels for interviews and observations. The documents also provided insight into CF's official perspective on how tasks should be structured and executed. Semi-structured interviews with managers and clerks were sources of primary data in the form of personal experiences and thoughts (Schultze and Avital 2011). We interviewed clerks to capture the experiences of those who used (O)LMS in their daily work and interviewed managers across all three departments to determine how the change affected the groups' performance. In addition, we used participant observation to understand the clerks' context and what they actually do in their daily interactions with (O)LMS.

From early 2013, three of the authors were extensively on site at BANK to explore the transformation program overall. We analyzed documents and talked to the IT department's project team. In this phase, we identified CF as our case site and negotiated access prior to the go-live. Beginning in November 2013, we intensified our field presence, with two authors regularly visiting CF to follow its change management activities. We captured data in three waves (Figure 2).

Table 1. Core Tasks and Departmental Responsibilities at Credit Factory

Task	Description	Department		
		PSN	PSS	SSG
New loans	Processes new loans (e.g., open new account, create customer record, enter data for new contracts into the loan management system)	R	–	S/E
Existing loans	Processes changes to existing loans in stock (e.g., prolongations, repayments)	–	R	S/E

Departmental Roles: [R] department responsible for carrying out task and fully dedicated to its execution; [S] department provides support for carrying out task (e.g., in case of high workload); [E] department handles exceptions (e.g., incomplete or faulty data).

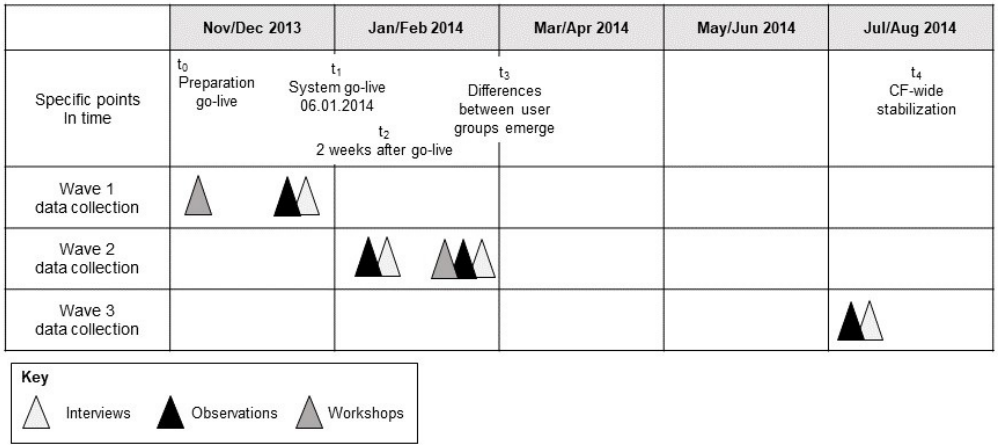

Figure 2. Data Collection

Prior to adoption (wave 1; up to t_1), one author participated in the basic LMS training (November 2013) and two authors held interviews and undertook participant observation to capture the status quo with OLMS still in place (December 2013). Our observations focused on clerks because we were interested in how they executed their system-enabled tasks. Rather than observing managers, we interviewed them for triangulation and as a source of contextual information.

Wave 2 captures the weeks after go-live in January 2014 (from t_1 to t_3). The same two authors were on site to observe the wave 1 respondents and undertake selective interviews (t_2). We held another round of interviews at the end of February 2014 (t_3), and again focused on clerks to study their reaction to the change from OLMS to LMS.

Wave 3 covered July and August 2014 (weeks around t_4), when we executed the same data collection activities as in wave 2. With one exception in PSN, and four exceptions in SSG, the subjects remained the same during waves 1, 2, and 3 (Table 2). After August 2014 (past t_4), we spoke with CF members to discuss and confirm our findings, but at that point data collection aimed at corroborating our findings from the earlier data collection.

We held the interviews, which lasted between 30 and 120 minutes, in German. After ensuring interviewees' informed consent, we tape-recorded and transcribed all interviews. We had two interview guides: one for managers and one for employees/clerks who were current or prospective (O)LMS users. To gather information on all the stages, we used different interview guides during waves 1, 2, and 3. In each stage, the interview questions were largely semi-structured and focused on how the systems were being used, what impacts and outcomes were being seen, and how the learning process was unfolding. We also asked for temporal perspectives (with questions about the status quo now, the previous status quo, and expectations for the future).[4] Complementary to our interview data, we captured all observations in memos

[4]A longer version of the paper with the six interview guides (3 waves × 2 roles) can be obtained from the authors.

Table 2. Interviews and Observations*

Area/Role	Persons	Wave 1		Wave 2		Wave 3		Total	
		Int.	Obs.	Int.	Obs.	Int.	Obs.	Int.	Obs.
Senior Management	4	3	0	5	0	2	0	10	0
Process Management	1	1	0	1	0	1	0	3	0
PSS Manager	1	1	1	1	1	1	1	3	3
PSS Clerk	3	3	3	3	3	3	3	9	9
PSN Manager	1	1	0	1	0	1	0	3	0
PSN Clerk	4	4	4	3	3	3	3	10	10
SSG Manager	2	1	–	2	–	1	0	4	0
SSG Clerk	8	8	8	7	4	4	4	19	16
Total	24	22	16	23	11	16	11	61	38

*Interview transcripts ~ pp. 1,200; observation memos ~ pp. 400; documents ~ pp. 600 (e.g., status reports, organizational charts, training material).

during or after participant observation and at the end of each day on site (Volkoff et al. 2007). Three of the authors discussed their experiences in daily debriefings and documented insights in analytical memos.

Data Analysis

Our data analysis followed recommendations for critical realist research (Wynn and Williams 2012), which suggest an iterative approach to data collection and analysis similar to that used in the grounded theory method (Corbin and Strauss 1994). In our data collection and analysis, we initially drew on the effective use model (Burton-Jones and Grange 2013) to understand how effective use is achieved over time. At first, we applied open coding to make the data transparent.[5] This process allowed us to identify and describe the principal events and to build a timeline. The first round of coding started after the wave 1 interviews and we extended the coding iteratively, with more interviews and observations of the LMS impacts after the go-live. We repeated this approach until all data were collected and analyzed, resulting in over 780 open codes associated with more than 2,000 quotes.

In our initial analyses, we focused on the users' learning actions as described in the effective use literature. We coded our data accordingly and identified distinctive patterns that signified different efforts in users' attempts to achieve effective use, and we extracted candidate mechanisms to explain the relevant differences across the user groups (Wynn and Williams 2012). Analysis revealed that the learning actions and associated effort were symptomatic of an underlying mechanism. For example, "exploiting learning experience" was a candidate mechanism because all user groups could exploit the knowledge gained from the training. However, when opening new loans, clerks seemed to apply their learning experiences more directly than when processing existing loans. Consequently, mechanisms like this cannot explain why the knowledge helped clerks to complete some tasks effectively but not others.

The apparent lack of causal efficacy of such candidate mechanisms compelled us to continue our discovery and elimination of competing explanations. We therefore reengaged with our data to decompose the observed events into meaningful component parts and describe the events in a theoretically meaningful way, leading to gradual refinement and extension of our conceptual framework. Continued coding to further categorize the data according to the developing framework (axial coding) allowed arrangement of the data to support more refined conceptual analyses (selective coding) and subsequent theorizing (Morton 2006; Wynn and Williams 2012).

Throughout the analysis, we returned to the data repeatedly to verify the preliminary results, such as codes, categories, and concepts (Wynn and Williams 2012). Revisiting the data allowed us to challenge our analyses and corroborate the plausibility of proposed mechanisms, and ensured that we remained grounded in the data while challenging and further developing our concepts.

[5]Coding examples can be obtained from the authors.

Theorizing Differences in Achieving Effective Use

In our case study, differences in achieving effective use materialized between the clerks carrying out tasks related to new and existing business. In presenting our results, we focus mainly on the differences between PSN (clerks responsible for processing new business) and PSS (clerks responsible for processing existing loans). As indicated, we used SSG to replicate our analysis and confirm our findings. SSG clerks carry out their PSN- and PSS-related tasks in a manner that makes them structurally similar to how work is done at PSN and PSS, allowing us to deepen our understanding of where troubles emerged (e.g., missing information) or support was required. After the go-live of LMS, problems and support centered mainly on the task related to existing loans. We detected an escalation of troubles across PSS and whenever SSG clerks executed a task related to existing business. This awareness allowed us to treat SSG similarly to PSS with regard to complexity but still as a separate case, which helped to corroborate our findings.

In the absence of a ready-made approach in the literature to analyze our data, we developed our own. We first examined either purely system- or purely task-related mechanisms. While these approaches did not produce sufficient causal efficacy to explain our findings, they allowed us to analyze the pertinent system and task properties in detail. This helped us uncover and analyze the co-dependency between task and system and allowed us to demonstrate why only our complexity-based mechanism possesses the causal efficacy to explain the differences we observed in learning effective use. We then focus on the new structural properties of system-enabled tasks we have developed and present case evidence for these properties in Tables 5 and 6. We then show how the structural properties form the complexity-based mechanism that, in use and over time, makes learning effective use difficult. This process is depicted for PSS (Figure 3) and for PSN (Figure 4). Table 3 summarizes the steps of our analysis.

Table 4 presents the empirical evidence for claiming that differences in learning effective use exist between the clerks processing new loans and those processing existing loans. A look at the events across the three departments reveals that within two months after the go-live, clerks in all departments had achieved similar levels of transparent interaction (TI) in terms of timing and effort. However, clerks in PSS and SSG struggled more than expected, especially regarding achieving desirable levels of representational fidelity (RF) and informed action (IA).

Our assessment was supported not only by interviews and observations but by CF's organizational performance measures, such as the number of open cases (new or changed loan contracts still to be processed) at each day's end, with PSN returning to pre-implementation levels about four weeks after the LMS go-live.

In contrast, clerks working on tasks related to the processing of existing loans took much longer to return to normal operations. Although they employed learning actions similar to those of colleagues working on tasks related to new loans (e.g., consulting training material, receiving support from change agents, talking to peers, etc.), they particularly struggled with learning how to achieve representational fidelity. In effect, clerks in PSS and SSG who worked on processing existing loans were unable to achieve effective use. Problems at PSS and SSG escalated (measured by the number of open cases) and clerks' organizational performance deteriorated. As an example, prior to the LMS release, 400 to 500 open cases at the end of a week were considered alarming, but about eight weeks after the go-live (t_3), PSS's open case levels reached 10,000. These difficulties persisted and in follow-up interviews CF's management suggested that PSS and SSG only returned to pre-implementation performance about eight months after implementation (t_4). Clearly, the two groups had very different paths to effective use.

System and Task Properties

In searching for explanations of the case events, we first focused on system properties. The change to LMS was the trigger for the transformation we observed in the case. The literature suggests that the new navigation order and the new system logic constrain individuals when working (Volkoff et al. 2007). In our case, users had to learn how to use the newly released LMS effectively to accomplish their task-related goals.

In line with the literature, we observed constraints related to transparent interaction in all the user groups. As standard software, the LMS GUI (graphical user interface) differed greatly from that of OLMS. For instance, LMS has screens and data fields that are not relevant for CF clerks. We found that all groups learned the new GUI relatively smoothly. Our data showed no significant differences in the groups' efforts. In contrast, clerks working on the task of processing existing loans (in both PSS and SSG) faced constraints regarding the system logic. For instance, PSS clerks were initially unable to work owing to technical problems caused by the data migration (e.g., some loan data fields contained incomplete or wrong values) or because the new functionality of the system proved challenging (e.g., loan-related calculations in LMS). This prevented them from doing their work:

Table 3. Summary of Theoretical Analysis Steps

#	Description	Input (case examples)	Output (case examples)
1	Analyze **system properties** to understand and document the navigation order and system logic.	Screen shots, system documentation, user training, interview data, observations, etc.	**Analysis of system structures** (see the "System and Task Properties" section)
2	Analyze **task properties** to understand and document the processing order and business logic.	Process diagrams, business domain glossary, interview data, observations, etc.	**Analysis of acts and information cues** (see Tables 5 and 6, columns 3-4)
3	Analyze properties of the **system-enabled task** to understand and document the **extent of dependencies** between system and task acts and information cues.	Interview data, observations, training cases	**Identification of representations** (see Tables 5 and 6, column 5)
4	Analyze properties of the **system-enabled task** to understand and document the **degree of dependencies** in terms of deep structure salience.	Output Step 3	**Identification of semantics** (see Tables 5 and 6, column 6)
5	Compare different system-enabled tasks and infer their **relative complexities in use**.	Output Step 4	**Identification of representational complexity** (see Figure 3 vs. Figure 4)

> OLMS, our [old] program, ticks differently than LMS. You must understand this booking logic [how LMS implements booking on accounts in its system logic]. How does LMS tick? How does LMS process this? What's going on in the background? (Clerk 3, PSS, t_3)

We found similar issues where SSG clerks worked on the task of processing existing loans. Here, clerks' work was constrained owing to a lack of knowledge about the LMS system logic:

> LMS is actually very difficult, because in spite of the preparation, in spite of the scenarios that we had practiced [training cases], we still don't really know … well … how it works [system logic]. (Clerk 2, SSG, t_3)

Interestingly, similar system-related problems affected PSN clerks far less. They did not have to learn much about the underlying system functionality to do their tasks:

> It was problematic that we did not know one-hundred percent what data we had to enter, whether we had to incorporate certain cost items, for example, for loans or not … [However, we] had relatively few problems. We had had the training; we were well prepared. There were few technical problems. Of course, sorting out what we did not know from the start took time. But there were no major problems. (Clerk 1, PSN, t_3).

Consequently, we found that constraints imposed by the system logic played a bigger role in PSS and SSG than in PSN. However, we also found that system properties alone cannot explain why deep structures are more salient in the learning actions of clerks processing existing loans. As all the clerks used LMS—the same system with the same properties to conduct tasks they had previously understood and completed successfully—we concluded that differences in learning effort related to system logic were linked to the structure of the respective tasks.

Consequently, we analyzed the groups' respective tasks, but without taking their links to the system into account yet. In line with the literature on task properties, we found that adding learning actions to existing tasks affected the task complexity (Wood 1986). Individuals across all three user groups used learning actions in response to their inability to do their work (Burton-Jones and Grange 2013). From our data, we identified different actions to engage in learning: clerks used the training material and referred to what they had learned in training sessions, explored and tried out the system, or asked their peers and external consultants for help. We also observed team-internal, lateral support, with team members gathering around a colleague's computer to collectively find a solution to a problem:

> First, I looked if there was anything I could do differently and if I did not progress [in the system] after one or two attempts, I asked my deputy team leader … he also knew LMS pretty well, and this is how we solved problems. (Clerk 1, PSN, t_3)

Table 4. Different Outcomes for Effective Use (EU)

EU	Clerks Processing New Loans		Clerks Processing Existing Loans	
	Quote	**Interpretation**	**Quote**	**Interpretation**
Transparent Interaction (TI)	"Previously, it [the meaning of data fields in OLMS] was clearer, because, each field had a specific name [matching the data that needed to be entered], which is different in LMS and does not always fit one-hundred percent. LMS is also a bit weird, so you sometimes have to fill the fields in a particular order to prevent error messages from appearing, and if you do not push the enter button, LMS hangs. And you cannot continue." **(Clerk 1, PSN, t_3)**	➔ **Attempting to achieve TI after the go-live** Directly after the go-live, employees struggled with the LMS GUI, because it differed from OLMS and they had to understand how and where to enter data. The expression "weird" specifically suggests that initially there was no TI.	[Clerk looking back to initial days:] "It's just extremely confusing. It's endlessly annoying that you need to repeatedly return to the previous screen, return, return, return, instead of saying, 'I can jump from here to there, I can jump there, or I can quickly look at the mortgage to see if they [the data] are somehow in there.' No, you have to back out of it [the screen]. You can open multiple windows, of course. I am not there yet, because that would drive me completely crazy, as I will completely lose track. Where am I now? Because you don't recognize this at a glance." (**Clerk 8, SSG, t_4**)	➔ **Attempting to achieve TI after the go-live** PSS and SSG employees struggled with the LMS GUI (navigation order), because it differed from OLMS, and they had to understand how to navigate the screens. In this phase, PSS and SSG clerks had no transparent interaction with the system.
	"Well yes, because we pretty much have to enter [loan data] manually, I would say that we now know how a loan is opened in the system [LMS] and what has to be entered where – now, you have it memorized." **(Clerk 1, PSN, t_4)**	➔ **Achieving TI** PSN clerks had memorized all the relevant aspects of the surface structure for their tasks in that phase. They could therefore use the system effectively in this regard.	"We have certainly become more experienced. If I have to look up the same thing five times, I actually know where it is and I no longer need to think about it so much. I've now developed a certain routine of always clicking in the same order." **(Clerk 2, PSS, t_4)**	➔ **Achieving TI** PSS clerks have developed a routine of navigating through the system. Transparent interaction with the system has improved.
Representational Fidelity (RF)	"And our tasks are actually constructed like this: Enter this [value into the system], enter this and that. This of course makes it difficult to understand why you have to do it. It has been documented (in the CF task documentation) so that once everyone has read it, they should theoretically be able to handle it. Although this is also the disadvantage of describing the smallest detail: Enter this and that, but no explanation of why you need to do it this way." **(Clerk 3, PSN, t_4)**	➔ **Sufficient level of RF achieved, but an understanding of the deep structure does not seem necessary** PSN clerks do not have to understand the deep system structures to do their tasks. It is sufficient for them to understand the surface structures to use the system effectively. In this sense, they achieved representational fidelity once they understood the surface structures (i.e. they achieved TI).	"Well I'm still not one-hundred percent clear how the system ticks. With OLMS I understood that if I do this, it has that effect, and if I don't do this, it has that effect. If I did something wrong, I could fix it and, in LMS, I don't know how to do that." (Clerk 2, PSS, t4)	➔ **No sufficient level of RF achieved** PSS clerks had not obtained representational fidelity after six months, although their tasks required this. To some extent, the consequences of certain actions in the system logic remained particularly unclear.

Table 4. Different Outcomes for Effective Use (EU) (Continued)

EU	Clerks Processing New Loans		Clerks Processing Existing Loans	
	Quote	**Interpretation**	**Quote**	**Interpretation**
Informed Action (IA)	"Initially, it was certainly a bit bumpy. We first had to get to know the system.... We then just tried it out with the new task descriptions and the training materials. But I think we managed this quite well and it worked. Questions did, of course, arise during work and we clarified them step-by-step. In the first three weeks we were certainly slower, but now we're back to normal." **(Clerk 1, PSN, t_3)**	➔ **Sufficient level of IA achieved soon after the go-live** PSN clerks had to improve their understanding of the system, but after approximately four weeks, they were back to using the system effectively and could perform informed actions.	"Well, not quite as dramatic … as in February, but the feeling of insecurity is still there. It is inevitable, but also clear to me, and I do not panic. But it's a slap in the face if you have safely worked for years with a certain routine and all at once you make mistakes that you really do not want to make. For example, because I forgot to enter a 9 somewhere [in LMS], money was deducted [calculation in the system logic] from the customer's account twice, or something." **(Clerk 2, PSS, t_4)**	➔ **No sufficient level of IA achieved** After six months, PSS clerks had not fully achieved the effective use they had with the old system. They can do their work, but still feel that their actions are only informed to a limited extent. The lack of informed action here is again closely linked to a lack of understanding of the LMS system logic that does not seem to fit the domain logic of calculating certain loan data.

> I was stuck with a case again, and I asked: "Man, can we look at it together?" It was during the time when we tried helping ourselves and, indeed, when we sat there together and looked … well… what checks does [a colleague] perform? How does [s/he] approach the case? And, yes, you can learn a lot from that. (Clerk 2, SSG, t_3)

When learning actions are added to task performances, the added steps increase the task complexity. As component complexity (Wood 1986) grows, learning actions affect other task properties: the user must effectively combine the newly added learning actions with existing work-related actions, raising the coordinative complexity (Wood 1986). Depending on the initial interdependencies between the work actions, clerks must also choose between leveraging what they have learned from their training and the training documents, exploring the system, and communicating with their peers or external colleagues. We also found that artifacts providing guidance for employees during their work performances (e.g., task documentation) changed frequently, demonstrating that no definitive recommendations existed on how to use LMS effectively. The result was an increase in dynamic complexity (Wood 1986), and after the go-live, clerks felt that their work had become more complex and was not as simple as they had previously perceived it to be:

> You sit here thinking that nothing works at all … at the beginning, we had the feeling that it wasn't that simple, that it doesn't work like that. … The first one, two, four weeks … we really had a problem. And we were not simply stupid. …[Management] always said: "Come on, you need to reduce [the backlog of cases], do something!" "Yes," we said, "We would like to, but it's not that simple." (Clerk 1, PSS, t_3)

We found that although these changes in component, coordinative, and dynamic complexity constrained all users by raising the level of overall task complexity, their causal efficacy was insufficient to explain the over-proportional escalation of effort required of PSS/SSG clerks to achieve effective use. Soon after PSN had returned to effective use, a manager stated about PSS:

> We face a catastrophe! We have lost all our tasks. [In PSS] we work with screenshots now … many work activities need four to five times as long as before. [LMS's logic] makes processing more complex. … We just don't understand [LMS]. … That's really, really bad at the moment. This affects all the employees, it affects customers … since we just cannot [work]. (Senior Manager, CF, t_3)

This response contrasts sharply with the experiences of PSN clerks. Looking back, a PSN clerk reflected on experiences made around the same time the struggles at PSS/SSG materialized:

> Now I'd say I work as well as before. … You simply enter the loans one by one and, at some point, you know it by heart. Most of us have internalized this, which allows us to process it quickly. … You now know where to enter what. For some things you understand what effects this has in the background, [but] you don't have this [knowledge] about everything. You say, "Well, I enter this value, and then it's just like this." We always work according to

> this [task description], so you don't need to know why you do it … or need … the exact technical knowl=edge [of] what happens in LMS. (Clerk 3, PSN, t_4)

Our data show that the different user groups applied similar learning types to acquire the relevant knowledge. However, the learning actions that PSS and SSG required to eventually learn how to effectively complete the task of processing existing loans in LMS were of a different quality. The groups had to understand system logic, which implies comprehending the semantics of the business domain entities implemented in the system:

> What I found difficult … was reading the bookings. What's debit, what's credit? What about the over-payment, where does this come from? I found understanding these bookings difficult. … We had no one to explain it to us. So we dealt with it on our own; explained it to ourselves .... You are held up by thinking about it. We held one another up with such things. You look and think: "Why does it [bookings in LMS] not work like that?" And so you are held up for half an hour processing just one loan. (Clerk 3, PSS, t_3)

Classical task complexity measures cannot explain the complexity stemming from this different quality of learning actions and the higher associated effort for PSS/SSG to achieve effective use. This insight motivated our analysis of the co-dependency between system and task.

Emergent Properties of System-Enabled Tasks

Our observations show that the escalation in effort required at PSS and SSG is linked to a persistent lack of effective use, particularly with respect to difficulties in learning representational fidelity (Table 4). Our previous analysis suggests that the PSS/SSG clerks' underlying need to understand the system logic does not stem from system properties or task properties alone, but also from the different properties of the emergent system-enabled tasks between PSN and PSS/SSG. We propose two such properties that help us identify relevant differences between the system-enabled tasks of PSN and PSS/SSG. One difference emerges from the relationship between task and system structures, the other from the relationship between surface and deep structures.

System dependency refers to the relationship between task and system and is based on representation theory, which maintains that system structures represent real-world entities.[6] For example, the business entity "loan" in the domain BANK is an entity that LMS must represent. In the real world, when any instance of a "loan" experiences a change in its states (e.g., when the customer repays an amount, which affects the loan balance), the LMS must reflect that state-change. In terms of system dependency, we argue for the importance of differences in the number of domain entities relevant to a task act or its information cues actually represented in the system.

System dependency matters from a task doer's perspective. Doing a task with a system that represents few domain entities requires relatively little cognitive effort in terms of mapping real-world entities to their system representations. In contrast, greater mapping effort is required to do tasks where many domain entities relevant in task execution are represented in the system. We conclude that system-enabled tasks can be characterized by the quantity of the representation between a task and a system. We define this property as *system dependency (SysD)*—that is, *the extent to which domain entities relevant for an act or information cue are represented in the system used to support the respective act. System dependency is expressed as a fraction of domain entities relevant in that act or its information cues, which are represented in the system, over the entirety of domain entities in that domain.* The definition as a fraction enables comparisons of different acts within a system-enabled task and comparisons of different system-enabled tasks.

Semantic dependency describes the quality of the relationship between surface and deep structures. Here, we posit that system-enabled tasks exist that require more business and system logic than others. That is, we differentiate between system-enabled tasks requiring little judgment about semantics and those needing more intense semantic engagement. We therefore define *semantic dependency (SemD) as the degree to which the user requires semantic understanding when making judgments to advance the completion of a system-enabled task. Semantic dependency is expressed as the number of domain entities represented in the system that are relevant for that task weighed by the depth of the understanding required to complete the task*.[7] The weight assigned to the respective entity is needed to account for the salience of semantics in the system-enabled task.

[6]For simplification we use the term *entity* for any real world phenomena that could be represented in a system.

[7]Our analyses of this weight rely on the terms clerks used to describe the intensity of their cognitive engagement with the domain entities relevant in the acts that make up their respective tasks. Where possible, we corroborated and extended this based on our observations. For instance, terms like *understand* or *interpret* suggest a different quality of cognitive effort than *check*, *scan*, or *update*.

The two concepts of system dependency and semantic dependency allow us to conceptualize the emergence of the co-dependency between the task and system. We found empirical evidence of these concepts by decomposing the system-enabled tasks of PSN, PSS, and SSG and analyzing their acts and information cues, along with their associated links to LMS.

We analyzed the system-enabled task of processing existing loans as carried out by PSS and SSG clerks. The goal of the underlying task is to make changes to loans in stock and to process and book the relevant payment transactions. A look at how this task was done by PSS clerks shows that several of the task acts depended on LMS (Table 5). Where we found such system dependency, we analyzed whether LMS represents business domain entities required for the act or the information cue. For instance, we identified one entity (columns "acts" and "information cues") referring to the "loan contract" (e.g., customer information such as name and address; loan conditions such as amount and interest rate).

Table 5 shows that we identified seven acts (5, 6, 7, 8, 9, 11, 12) with relevant domain entities represented in LMS. Beyond the loan contract, we also found evidence that business domain entities represented in LMS include loan status (e.g., current information on the loan, such as the amount due), loan calculations (e.g., calculations defining the repayment plan), and booking and payment transactions to transfer money from the loan account to another payment account.

When we analyzed and decomposed the PSS task—initially without our new concepts—we realized that the dependency between the task and the system involved relatively detailed understanding of the mapping between domain logic and system logic. A PSS clerk explains:

> If [the salesperson] says we need a loan redemption, s/he needs to tell me which loan, where to take the money from [which account in LMS], and on which day I need to do so. This information must be provided, because without it, I cannot work. When this [activity] has been completed, I look up the [account] number in LMS. I then run the necessary calculations, for which I can use the OLMS. (Clerk 1, PSS, t_1)

To illustrate our analysis, we identified acts 6, 7, and 8 as containing semantics because terms like *understand* or *interpret* suggest a different quality of understanding and cognitive effort than *check*, *scan*, or *update* (the latter were often used by PSN clerks to describe their acts). This difference stems from the semantics of business domain entities. This difference in terms is symptomatic of a higher semantic dependency.

In act 7, for instance, we found that clerks were required to understand the semantics of the business domain entity "loan" to determine the loan status and make the necessary calculations.

As the quote above suggests, clerks need to understand how transactions are performed, how interest is calculated, and how a repayment plan works. We performed the same analysis with respect to SSG to triangulate our findings and found results similar to those of PSS.

The system-enabled task of opening new loans (as primarily done by PSN clerks) involves the acts and information cues required to open a new loan in LMS, to notify the end customer, and to disburse the money to the customer.

Table 6 reveals that several of the task acts depended on LMS (acts 5, 6, 7, 10, 11). In act 5, for example, the clerk checks the loan data records in LMS against those in the original scan of the paper-based loan contract version. The "loan contract" is an essential business domain entity for completing the act, thus needing to be fully represented in LMS.

Acts 6 and 7, where the clerk enters loan data records into LMS, also depend on the system. Act 7 requires loan contract data from LMS as a source of the information cue required for the clerk's decision on whether the data records need correction.

Next, we analyzed whether the acts containing relevant representations in LMS entailed an understanding of semantics. We looked for judgments requiring a user to understand the semantics of domain entities. For instance, the judgment required in act 5 does not involve any semantics:

> I receive a loan contract, I open it … and I have my editing program. Sometimes it sounds a bit simple, but if the contract is perfectly fine [no missing values], I just transcribe it, and … another colleague approves [it]. (Clerk 1, PSN, t_1)

The clerk simply needs to compare the entries on the physical contract with the data that LMS displays. We analyzed all other acts containing representations and found no links to semantics.

Our analyses led to a twofold realization. Compared with PSN, the fraction of relevant domain entities represented in LMS is greater for clerks at PSS who engage with the system-enabled task of processing existing loans *and* the representations require greater understanding of the underlying semantics (i.e., deep structures). In other words, the system-enabled task of processing existing loans at PSS entails a greater extent of dependency on LMS *and* is more dependent

Table 5. Emergent Properties of the System-Enabled Task at PSS

ID	Observations	Acts *(incl. represented entities)*	Information Cues *(incl. represented entities)*	Representation	Semantics
1	The clerk receives a processing order through the workflow tool, talks directly to a sales person, or receives a list of errors.	Receive new order	"Standard order list" or error list	*Not evident*	
2	The clerk acquires the account ID from the order, or salesperson, to identify the loan and its status.	Open dedicated order, or create order	Account number or error message	*Not evident*	
3	The clerk remembers the act sequence for a loan change or double-checks in the CF task description.	Remember process steps, or look up	Act sequence and steps	*Not evident*	
4	The clerk checks whether the required documents are complete to determine a change to the loan.	Scan documents for completeness	Documents complete	*Not evident*	
5	The clerk navigates through the system to find account ID and loan data.	Look up account ID in LMS	*Loan contract* and *loan status*	*SysD+*	*Not evident*
6	The clerk needs to understand what the values in the data fields displayed by LMS mean and how the LMS system logic calculates them.	Retrieve and understand the *loan contract* data and *loan status*	Understand where to make changes in the *loan contract* and *loan status*	*SysD+*	*SemD+*
7	The clerk needs to perform calculations. She needs to understand how the system performs these calculations (system logic) and if the results are correct (domain logic).	Perform *loan calculations* required for making changes to the *loan status*	Mathematical calculation logic to perform *loan calculations* required for making changes to the *loan status* in the system	*SysD+*	*SemD+*
8	The clerk must have a clear understanding of the business concept "loan" and its representation in the system.	Interpretation of the *loan calculations* based on the old loan status, *new loan status*, and on the *loan contract* data	New *loan status*, old *loan status*, and *loan contract* data to interpret correctness of *loan calculations*	*SysD+*	*SemD+*
9	The clerk receives the calculation results and undertakes the required transactions with LMS and another system.	Perform *bookings/transactions* and make changes to *loan status* in the system	*Booking/transaction* logic to make actual changes to the *loan status* in the system.	*SysD+*	*SemD+*
10	The clerk documents the task's status in the workflow tool.	Document status	—	*Not evident*	
11	The clerk double checks the changes made in LMS before releasing the order for approval.	Check correctness and release for approval	Inconsistencies in *loan status* in the system	*SysD+*	*Not evident*
12	The clerk remembers act sequence, or double-checks in CF task documentation, and another clerk approves it.	Approval and release of money transfer done by another clerk	New order for approval with link to loan status in the system	*SysD+*	*Not evident*

SysD+: Acts have entities that are represented in the system and increase system dependency of the system-enabled task.
SemD+: Entities that are represented in the system require semantics and increase semantic dependency of the system-enabled task.
Not evident: Representation or semantics were not empirically evident in the act.

Table 6. Emergent Properties of the System-Enabled Task at PSN

ID	Observations	Acts *(incl. represented entities)*	Information Cues *(incl. represented entities)*	Representation	Semantics
1	The clerk receives a processing order through [a workflow tool]. Documents are attached and opened, such as a scanned loan contract.	Open dedicated order for processing	Update of the "standard order list"	*Not evident*	—
2	The clerk scans the processing order to see what she must do.	Scan dedicated order for processing	Order with data entry, or only check	*Not evident*	—
3	The clerk remembers the act sequence or double-checks in the CF task description.	Remember task steps, or look up	Act sequence and steps	*Not evident*	—
4	The clerk checks if all the required documents are attached for processing.	Scan documents for completeness	Documents complete	*Not evident*	—
5	The clerk scans the documents and loan data and matches them with LMS entries.	Scan documents with *loan contract* data and match with LMS entries	Deviation between LMS data and loan contract documents with *loan contract* data	*SysD+*	*Not evident*
6	The clerk copies data values from the contract into LMS.	Scan documents with *loan contract* data and enter values from documents into LMS	Deviation between LMS data and loan contract documents with *loan contract* data	*SysD+*	*Not evident*
7	The clerk checks if those contract values already in LMS are correct by comparing them with the contract.	Correct/Update wrong data entries based on documents with *loan contract* data	Deviation between LMS data and loan contract documents with *loan contract* data	*SysD+*	*Not evident*
8	The clerk writes a standard letter to the customer confirming the loan opening.	Create standard customer letter	—	*Not evident*	—
9	The clerk documents the process status in the workflow tool.	Document status	—	*Not evident*	—
10	The clerk double-checks the data entered into LMS before releasing the order for approval.	Check *loan contract* data entries and release for approval	Deviation between LMS data and loan contract documents with *loan contract* data	*SysD+*	*Not evident*
11	The clerk remembers the act sequence or double-checks in the CF task documentation. Another clerk approves the loan.	Approval of and loan opening done by another clerk	New order for approval with link to *loan contract* data in the system	*SysD+*	*Not evident*

SysD+: Acts have entities that are represented in the system and increase system dependency of the system-enabled task.
SemD+*:* Entities that are represented in the system require semantics and increase semantic dependency of the system-enabled task.
Not evident: Representation or semantics were not empirically evident in the act.

on system and domain logic. Thus, beyond task and system properties in isolation, the crucial difference between PSS and PSN results from the different extent of system dependency and different degree of semantic dependency in their respective system-enabled tasks.

This realization allows us to explain why PSS/SSG clerks had more problems with system logic than PSN clerks. Once a system-enabled task has a greater extent of system dependency *and* a higher degree of semantic dependency, users have to understand both the domain logic and system logic as well as how the two map to each other. For example, in PSS clerks' act 7, the clerk looks at certain loan parameters displayed in LMS. Simply understanding the loan calculation logic is insufficient. The clerk has to map her understanding of the business domain entity "loan calculation" (domain logic) to her understanding of how LMS represents this concept (system logic). This cognitive mapping process is part of users' learning when confronted with a new system.

Influence of Dependencies in Use

It is this learning that we propose is influenced by a different extent of system dependency and a different degree of semantic dependency. Users learn about the domain of their task, the relevant representations in the system, and their co-dependency to achieve effective use (Burton-Jones and Grange 2013)—a combination that links to our observations regarding the centrality of representational fidelity shared earlier (Table 4).

System dependency influences this learning by defining the extent to which users must cognitively map task structures and system structures. If the system represents only a few domain entities, little cognitive mapping is required. System dependency therefore creates complexity by imposing added cognitive demands (Campbell 1988) on the user compared to purely manual tasks.

Semantic dependency complements system dependency by capturing additional complexity for task structures represented in the system. Complexity increases once these structures involve semantics (i.e., deep structures), because understanding and mapping business and system logic impose higher cognitive demands than just understanding and mapping surface structures (e.g., following the processing order of a task on the system interface).

Once we discovered these emergent properties of system-enabled tasks in our data, we corroborated and expanded our understanding of their impact on users' learning through a detailed analysis of clerks' use of LMS over time and across the user groups. Figures 3 and 4 provide a stylized synthesis for a compact illustration by depicting how the system use of PSS and PSN clerks became effective over time. We show *use* as a structure represented by the outer boxes (A, B, and so on). Within each box, we describe the substructures of the system-enabled task (navigation order, system logic, processing order, domain logic) on the left side and the actions and cognitions a user needed to perform to achieve effective use on the right side. The opposing block arrows symbolize the learning effective use mechanism (from right to left) and the counteracting effect resulting from the complexity imposed by the emergent properties of the system-enabled task (left to right). The sequence of use boxes in Figures 3 and 4 allows us to contrast the differences in how effectiveness arises in PSS and PSN clerks' use and highlight the impact of our complexity-based mechanism where it is salient, illustrating how the added complexity impedes users' ability to achieve representational fidelity and effective use. For both figures, the progression of use boxes does not represent the sequential completion of the PSS or PSN tasks. Instead, the figures focus on how effectiveness of use arises over time and represent the different trajectories across the user groups at CF. Our discussion of the PSS and PSN tasks, which follows below, illustrates how and why the dependencies impede users' learning of effective use.

Beginning with PSS, the task of processing an existing loan is triggered when clerks receive an order to make changes to an existing loan contract. To decide how to process the loan contract, clerks must understand the loan calculations required for determining the loan status and be able to interpret the resulting values. For this task, Figure 3 reveals that clerks at PSS first encounter difficulty when trying to find the required information in LMS. They initially struggle because they are not familiar with the layout and structure of LMS. As step 5 in Table 5 suggests, this difficulty arises because clerks are confronted with system dependency in their system-enabled task. The new navigation order of LMS impedes their ability to interact with the system in a transparent manner (use structure A in Figure 3).

In response, clerks at PSS engage in learning actions designed to help them build a cognitive mapping between the processing order of their task and the navigation order of the system. From our data, we conclude that, compared to their colleagues working on new business, they require more learning at this stage because of the relatively higher extent of system dependency. However, because no semantic dependency is evident at this point, the mapping of surface structure enables clerks to achieve transparent interaction (use structure B). This result is also in line with our analysis of SSG clerks working on processing existing loans and fits the early success in achieving transparent interaction across CF.

The escalation of learning efforts occurs once clerks attempt to complete acts in their system-enabled task that contain relatively high degrees of semantic dependency (act 6 and onward). Here the ability to map surface structures to one another is insufficient to complete the task. Clerks' attempts to determine the loan status were unsuccessful because the indicators LMS calculates and displays for this purpose provided no meaningful input for taking informed action. For instance, the "principal amount," which is the amount the customer owes, is an important indicator of the loan status. The customer's incoming payments (minus the interest) are deducted from this amount to reflect the actual amount owed. In our case, the calculations that LMS performed (system logic) differed from the way this indicator was defined at CF, which the clerks were used to (domain logic). LMS did not automatically deduct incoming payments from the principal amount (as OLMS did), but accumulated them in a separate account where they were held for processing by clerks, creating problems for clerks attempting to process existing loans (use structure C).

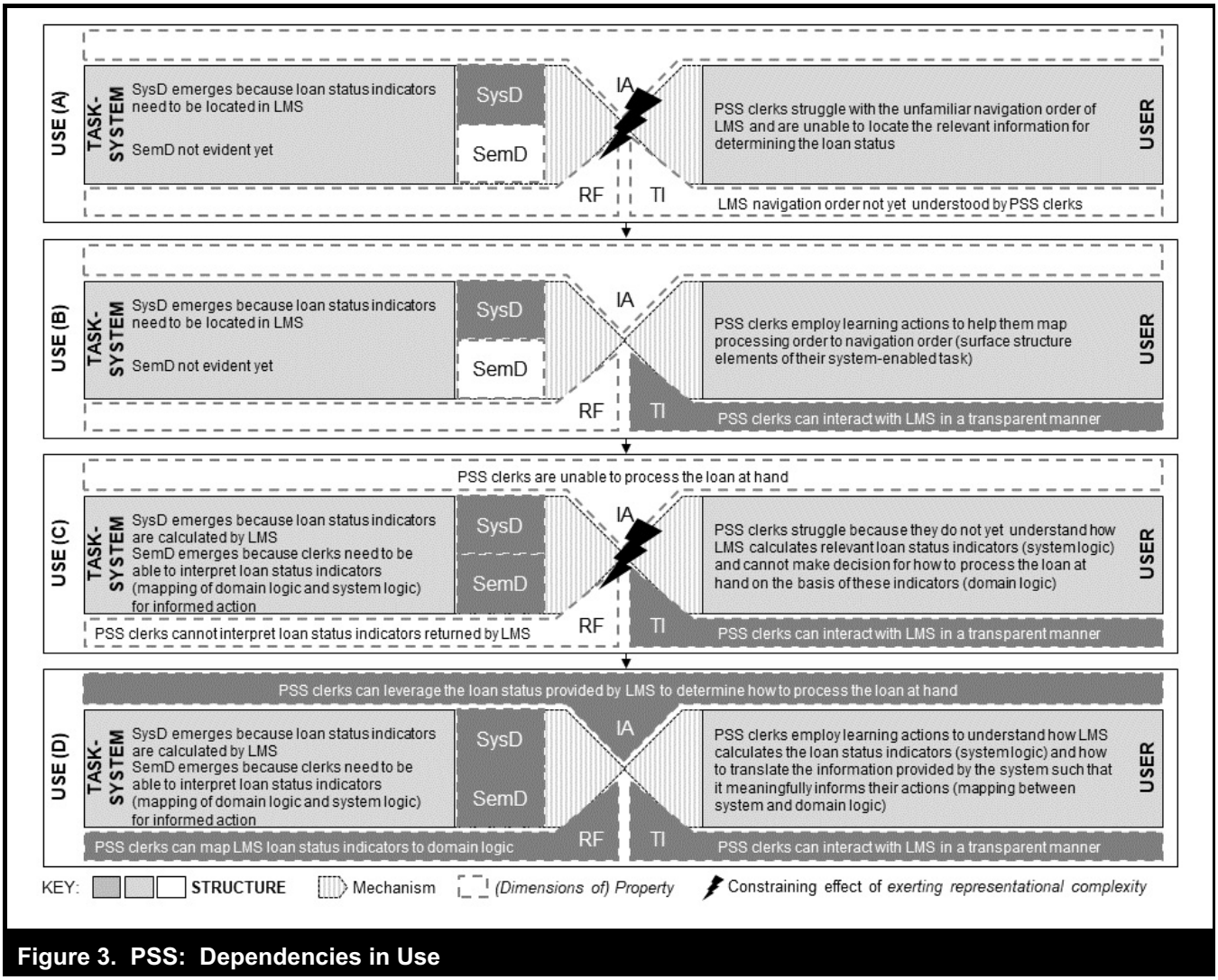

Figure 3. PSS: Dependencies in Use

Clerks did not understand the information LMS provided until this difference in how LMS calculated the principal amount was discovered and relevant learning materials were provided. However, these materials required clerks to employ learning actions beyond building a cognitive mapping between surface structures. Because acts 6 and onward are characterized by relatively high degrees of semantic dependency, clerks have to learn about the mapping between deep structures in the system-enabled task. In particular, clerks must understand what different values of the relevant loan status indicators mean, how the indicator values influence processing the transaction at hand (domain logic), and how LMS calculates the loan status indicators (system logic) relevant in this act. While clerks could draw on their existing cognitive understanding of the domain logic to help with understanding the indicators and their role per se, the way LMS calculated these indicators was new to them. Consequently, clerks had to map the domain logic to the system logic to understand whether the loan status retrieved from LMS faithfully represented the status of the loan contract such that a reliable decision could be made regarding the further processing of the transaction. In this process, learning effective use becomes more complex because the extent of system dependency requires clerks to learn how multiple domain entities are represented in LMS (loan contract, loan status, loan calculations). Complexity is further exacerbated by the degree of semantic dependency in clerks' system-enabled task, which requires them to understand domain logic and system logic and learn about the mapping between the two.

Clerks did not understand the information LMS provided until this difference in how LMS calculated the principal amount

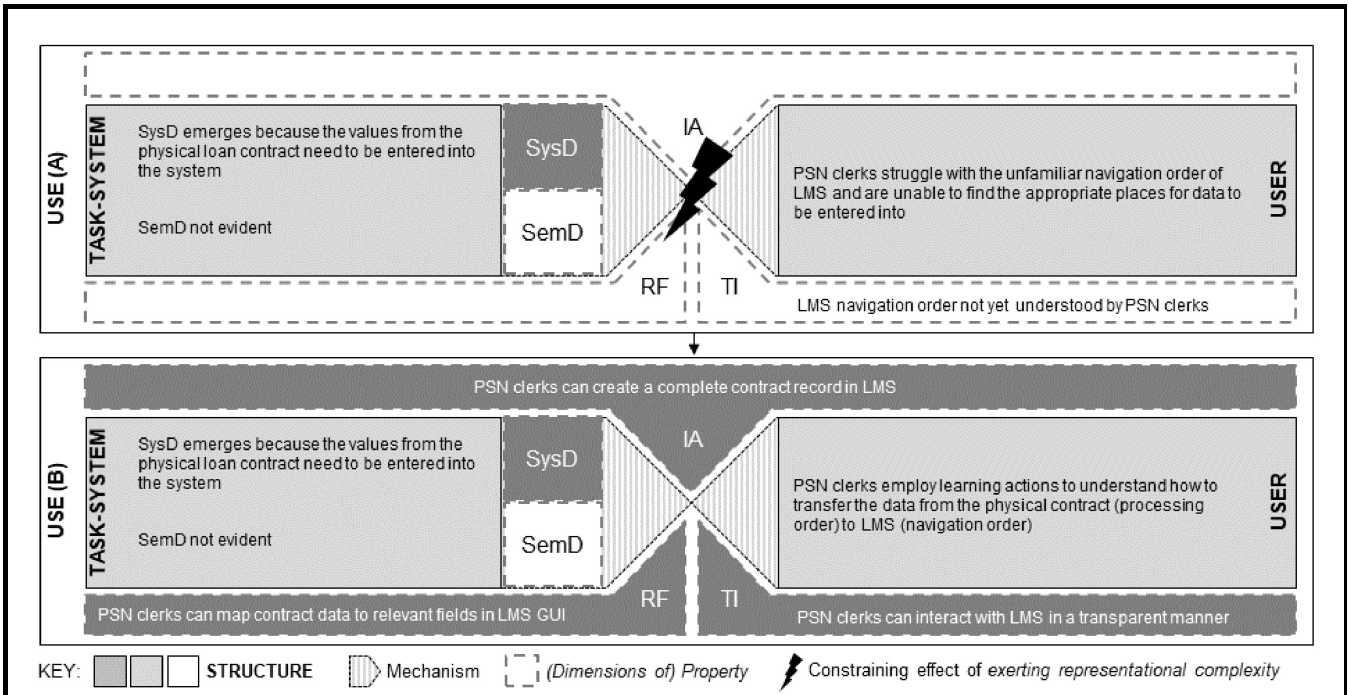

Figure 4. PSN: Dependencies in Use

was discovered and relevant learning materials were provided. However, these materials required clerks to employ learning actions beyond building a cognitive mapping between surface structures. Because acts 6 and onward are characterized by relatively high degrees of semantic dependency, clerks have to learn about the mapping between deep structures in the system-enabled task. In particular, clerks must understand what different values of the relevant loan status indicators mean, how the indicator values influence processing the transaction at hand (domain logic), and how LMS calculates the loan status indicators (system logic) relevant in this act. While clerks could draw on their existing cognitive understanding of the domain logic to help with understanding the indicators and their role per se, the way LMS calculated these indicators was new to them. Consequently, clerks had to map the domain logic to the system logic to understand whether the loan status retrieved from LMS faithfully represented the status of the loan contract such that a reliable decision could be made regarding the further processing of the transaction. In this process, learning effective use becomes more complex because the extent of system dependency requires clerks to learn how multiple domain entities are represented in LMS (loan contract, loan status, loan calculations). Complexity is further exacerbated by the degree of semantic dependency in clerks' system-enabled task, which requires them to understand domain logic and system logic and learn about the mapping between the two.

Our data suggest that once this mapping was successful (use structure D), PSS clerks were able to achieve representational fidelity. PSS clerks could understand and interpret the loan status indicators provided by LMS and could cognitively translate them to the information they required by their domain logic—a finding corroborated by our data from SSG. With restoration of their ability to interpret the information provided by the system, clerks were able to take informed action and process the existing loan.

Figure 4 depicts our analysis of the system-enabled task related to new loans as conducted by PSN clerks: they have to enter all the data from the physical loan contract into the system. This step involves entering data in various places in LMS and is successful when the loan is approved. For PSN clerks, the key challenge after the go-live was to figure out what goes where.

Initially (use structure A in Figure 4), clerks working on new loans struggled with effective use, primarily owing to a lack of transparent interaction. Consequently, learning actions were required. Table 6 shows that PSN clerks' system-enabled task exhibits a relatively lower extent of system dependency (i.e., the loan contract is the only business domain entity represented in the system). In contrast to the clerks working on processing existing loans, the clerks at PSN did not face any discernible degree of semantic dependency

in their system-enabled task. Clerks engaged only in learning actions required to map navigation order and processing order such that all data from the physical loan contract could be transferred to LMS.

Like their counterparts in PSS and SSG, clerks in PSN were able to achieve transparent interaction with the system relatively quickly and with the simple learning actions discussed thus far. However, our data suggest that once the clerks had figured out what goes where in the system and had a chance to build an initial routine around this mapping, they also achieved representational fidelity. In the absence of semantic dependency, the only representation relevant to them occurred between surface structures. PSN clerks were able to leverage their simple learning actions to improve representational fidelity to a degree where they were able to achieve the goal for using the system: to create a complete loan contract record. Once meaningful representation between surface structure elements of their system-enabled task had been established, PSN clerks leapfrogged into informed action (use structure B).

Comparison of the last steps of Figures 3 and 4 is particularly interesting. These steps represent a state where the clerks' system use has become effective. Figure 3 shows that effectiveness of use is achieved once a PSS clerk can navigate quickly in the GUI to find the loan status and correctly understands the loan status displayed on the screen. The clerk can now make correct changes to that loan—an undertaking that involves significant system dependency and semantic dependency. Figure 4 shows that PSN clerks' effective use to enter values correctly results from fast navigation in the GUI and correct field mapping between the loan contract and the GUI. This result is achieved without significant degrees of semantic dependency being evident. We therefore conclude that system dependency and semantic dependency jointly produce an effect that influences the quantity and quality of users' learning.

Exerting Representational Complexity

We propose that the interplay of system dependency and semantic dependency with users' learning actions produces a mechanism we call *exerting representational complexity*, which counteracts the mechanism learning effective use. That is, the effort to learn how to use a system for a task effectively depends not on the system or task properties alone and in isolation, but on the emergent properties of their co-dependency in a system-enabled task.

To better understand the sources of the added effort required when users encounter representational complexity, we further scrutinized the actions of clerks across all three departments when learning effective use. We found four classes of learning actions: users learn to (1) map the processing order to the navigation order, (2) understand the system logic behind the information displayed on the screen, and (3) understand the domain logic inherent in the task so that they can learn how to (4) map the domain logic to the system logic. Users employ specific learning actions from these classes of learning actions to build cognitions that inform concrete action such that performing the action effectively and efficiently advances the goal-directed completion of the task. In a specific context (e.g., PSS and SSG clerks' learning effective use of LMS), users employ specific learning actions that are instances of these classes and it is the actual effort incurred by these actions that determines users' effort for learning effective use overall.

By investigating how our classification relates to the original learning actions proposed by Burton-Jones and Grange (2013), we can show how the counteracting effects of exerting representational complexity we observed empirically relate to the established theory. A relatively greater extent of system dependency will increase the number of learning actions needed to build the required cognitive understanding of all mapping relationships between task and system in a system-enabled task. An increase in the extent of system dependency means that more domain entities relevant for an act or information cue are represented in the system used to support the respective act. Correspondingly, more learning actions that map system and task to one another are needed, which affects the learning actions related to learning fidelity because the amount of learning increases (Burton-Jones and Grange 2013).

The effect of semantic dependency is complementary. A higher degree of semantic dependency means that users require semantic understanding when making judgments to advance the completion of a task. In system-enabled tasks characterized by system dependency, users will have to learn not only semantics (i.e., domain logic and system logic), but also the mapping between the two. As our case highlights, the learning actions PSS/SSG clerks need for this additional learning differ from those required for system-enabled tasks that exhibit no notable degree of semantic dependency. Our data suggest that the cognitive efforts increase, explaining the step-change in effort needed to turn around the project at CF.

In our case study, our theorized mechanism is corroborated by empirical observations. In the weeks around the go-live, we observed very similar trajectories of learning transparent interaction with comparable levels of effort and success across clerks of all three departments. At this point, learning

effective use was not yet exposed to the counteracting effects of exerting representational complexity and the differences in dependencies did not yet matter. This result is in line with the original concept proposed by Burton-Jones and Grange, because the task-related aspects of system use are not yet salient in this stage of learning effective use. Users can learn the navigation order independently of the task.

Once users turn their attention to learning how to achieve representational fidelity, all three aspects of system use—user, system, and task—become salient. In this learning, Burton-Jones and Grange suggest that users learn representations and the domain. In line with their foundations in representation theory (Wand and Weber 1995), this means that learning moves to a deep structure level. In contrast, our work shows that exerting representational complexity determines *what* is represented and *how* users learn to achieve fidelity. When a system-enabled task is characterized by relatively less system dependency and semantic dependency (as in PSN), fewer domain entities are represented in the system (e.g., only the loan contract for PSN) and these representations are restricted to surface structures (i.e., syntactical elements only). Correspondingly, users employ only learning actions that involve mapping the processing order to the navigation order. However, Burton-Jones and Grange's theory suggests that learning to achieve fidelity occurs at the level of deep structures. In terms of our task and system conceptualization, current theory would require the user to learn how the domain logic is represented in the system logic to achieve representational fidelity. But in our case, given the absence of semantic dependency at PSN, the representation of the domain logic in the system logic was not evident. Thus, PSN users achieved effective use far more quickly than the present effective use conceptualization suggests. Users were already familiar with the navigation order of the system when learning transparent interaction and were also familiar with the processing order of their task from their work experience. Accomplishing the mapping necessary to achieve representational fidelity thus required relatively low effort only.

When the system-enabled task is characterized by relatively more system dependency and semantic dependency (as in PSS and SSG), users need to employ learning actions from all four classes and for a relatively higher number of representations. Only when our two dependencies are high do Burton-Jones and Grange's original "learning representations" and "learning domain" become learning actions that involve semantics. Consequently, "learning fidelity" is no longer achieved at the surface structure level only, but also at the deep structure level. This result explains why not only more of the same kind of learning actions were required, but why much more effort expended on qualitatively different learning actions (i.e., learn domain and system logic and map the two) was needed to learn effective use at PSS and SSG.

We suggest that once users turn their attention to "learning to leverage representations" (Burton-Jones and Grange 2013), the differences that arise in the context of learning fidelity have ripple effects. Our observations at CF indicate that learning to leverage representations in PSN was simpler because of the lower impact of exerting representational complexity. In contrast, PSS and SSG clerks were confronted more with representations at both the surface and deep structure levels and needed to accomplish the corresponding mapping to learn informed action.

In sum, the mechanism of *exerting representational complexity explains the need for additional learning effort to achieve effective use of a system, as determined by the respective system-enabled task's system dependency and semantic dependency*. Table 7 summarizes how our work affects the learning actions in the original effective use conceptualization.

Discussion

Our work explains the progression of events at CF concerning the breakdown and recovery of effective use, especially regarding the differences observed between PSN and PSS/SSG. The explanation we offer produces a twofold theoretical contribution.

First, we conceptualize system-enabled tasks and their emergent properties: *system dependency* and *semantic dependency*. These dependencies explain the nature of the co-dependency between tasks and systems in use, especially in the context of users' efforts to learn effective use. Combined, the emergent properties allowed us to uncover an as yet unknown form of complexity that explains users' added efforts in learning effective use.

Second, drawing on these emergent properties, we reexamined how users learn to carry out system-enabled tasks effectively. In particular, the emergent properties contribute to the generative mechanism *exerting representational complexity*, which describes the interaction between the user and the system-enabled task. This mechanism explains how and why users' learning efforts differ for the same system when the differences in how tasks and systems relate to one another are taken into account. Exerting representational complexity influences individuals' efforts for learning how to use a system effectively by counteracting the *learning effective use*

Table 7. The Impact of Exerting Representational Complexity on Learning Effective Use

Effective Use			*Learning Actions*	System Dependency (SysD)	Semantic Dependency (SemD)
Transparent Interaction			*Learning physical structures*	[While physical structures were not pertinent in our case, we draw on the original effective use conceptualization (Burton-Jones and Grange 2013) to propose that their impact is similar to that of surface structures.]	
			Learning surface structures	SysD is implicated in learning related to effective use, but SysD has no salience in learning related to transparent interaction, because users learn navigation order independent of tasks.	SemD has no salience in learning related to transparent interaction, because system logic is not involved in users' learning of navigation order.
	Representational Fidelity		*Learning representations*	**A greater extent of SysD increases the effort required to learn representations** because users need to learn representations for a greater number of domain entities.	**A higher degree of SemD increases the effort required to learn representations** because users need to know more about the system logic of their system-enabled task.
			Learning domain	**A greater extent of SysD increases the effort required to learn the domain** because more domain entities are relevant in the system-enabled task.	**A higher degree of SemD increases the effort required to learn the domain** because users need to know more about the domain logic of their system-enabled task.
			Learning to achieve fidelity	**A greater extent of SysD increases the effort required to learn to achieve fidelity** because more entities need to be mapped between task and system structures.	**A higher degree of SemD increases the effort required to learn to achieve fidelity** because users' need to engage more in mapping domain logic to system logic (as opposed to learning to map processing order to navigation order), which requires learning fidelity at a deep structure level.
		Informed Action	*Learning to leverage representations*	**A greater extent of SysD increases the effort required to learn to leverage representations** because users need to learn how domain entities are represented in the system for a greater number of domain entities.	**A higher degree of SemD increases the effort required to learn to leverage representations** because building adequate cognitive understanding that is able to inform effective and efficient action is more effortful.

mechanism. In effect, we improve the understanding of how a combination of these mechanisms interacts to generate effectiveness of system use.

Our contributions have a number of theoretical implications. First, our findings on representational complexity have strong implications for the task complexity literature, especially that in the tradition of Wood (1986) and Campbell (1988). While this literature recognizes complexity as a key issue, this stream of research is concerned with tasks in isolation and has only recently begun to highlight the problems of omitting system enablement (Hærem et al. 2015). Our work shows that merely counting acts, analyzing interdependencies, and examining the dynamics of a task do not sufficiently conceptualize the complexity that users face. Accounting for the system that enables users to do a task is instrumental, because the co-dependency between the task and the system (as captured in our dependencies) adds a new form of complexity that affects system use. Future work must account for the increasing degree of interweaving between tasks and systems. Only such an entangled perspective allows an adequate capturing of system-enabled tasks.

Second, drawing on representation theory (Wand and Weber 1995) we recognize the role of syntax and semantics in both the system *and* the task. Applying this reasoning to tasks allows us to recognize important task characteristics that previous concepts do not provide—especially regarding the co-dependency with systems. In particular, our work advances understanding of the interplay between the task and the system and how users can improve their cognitive understanding of how the two fit together in learning effective use. This understanding connects to earlier efforts in IS research that sought to grasp the fit between specific tasks and the technologies used to carry them out (e.g., Goodhue and Thompson 1995).

Third, our work improves the understanding of how users achieve effective use, as it sheds light on the micro-level mechanisms required to actualize technological affordances in individuals' practices (Leonardi 2011). Most importantly, however, representational complexity challenges the current understanding of the level at which users achieve representational fidelity. We found that users attempting to achieve effective use in system-enabled tasks with little system dependency and semantic dependency had no need to establish representation of their domain, and hence no need for representational fidelity, at the deep structure level. This finding contradicts the current understanding of representational fidelity (Burton-Jones and Grange 2013) and representation (Wand and Weber 1995). Our concept of representational complexity helps resolve this tension between our case observations and the current state of the literature by introducing a qualifying effect. That is, this concept helps specify the boundaries of representation theory in that researchers can use it to understand in what conditions of system dependency and semantic dependency the predictions of representation theory will not work as expected in the literature. As a result, we contribute to what Burton-Jones et al. (2017) refer to as the *intension* of representation theory by providing a concept that allows the idea of representation to remain meaningful, even when system-enabled tasks are characterized by low degrees of representational complexity.

Beyond these theoretical implications, our work also offers a methodological consequence. Specifically, our approach to capturing and analyzing data relevant to our complexity-based mechanism allowed us to leverage representation theory in empirical work. Our congruent task and system conceptualization helped us to analyze the acts and information cues in system-enabled tasks and revealed system dependencies and representations. Identifying these is a critical prerequisite to better understanding what exactly is represented where and how. This allowed us to better understand how users learn to comprehend this representation and how they build fidelity regarding this representation. Such an approach will be useful in future studies attempting to advance representation theory (Burton-Jones et al. 2017).

From a practical point of view, our findings allow managers to better grasp the nature of system-enabled tasks in which the co-dependency of tasks and systems is a major source of complexity. Ultimately, this complexity is a major driver of effort in any digital transformation project. Feedback we received as part of a knowledge transfer workshop indicates that a refined understanding of this co-dependency offers new ways of modeling and documenting system-enabled tasks, allowing for more accurate analyses of the pre-change status quo. As a result, our insights support a better understanding of why the support efforts of similar quality and quantity offered to all three departments allowed PSN to achieve effective use relatively quickly while PSS and SSG struggled far more. In particular, we explain that the emergent properties of PSS's and SSG's system-enabled tasks led to representational complexity in use. Initially unaware of this complexity, CF's managers attempted to apply a one-size-fits-all training and change management approach.

While successful at PSN, this approach failed at PSS and SSG and contributed to the escalation of problems there. Eventually, only an adaptation of the quantity (i.e., additional resources deployed, time invested) and quality (e.g., task force with specially trained clerks, changes to the technology) of support efforts turned the LMS implementation around for PSS and SSG. We show that the corresponding step-change in effort, which initially piqued our interest to engage with the case, can be explained by accounting for the sources and effects of representational complexity. While a detailed description of the specific efforts that CF employed is beyond the scope of this paper, the emergent properties of system-enabled tasks give practitioners insights into the challenges that designated users of a new or changed system encounter. In turn, we provide the basis for identifying the issues with which users might need the most help and the training and support tools best suited to deliver this help, such as formats, documentation, or change agents.

Together with recent research efforts that investigate the training formats best suited to specific facets of effective use (Gnewuch et al. 2016), our work enables managers to better anticipate efforts and timelines and provide more suitable support structures to help employees achieve effective use quickly. In light of the centrality of individuals' ability to transform their work in successful digital transformations (Mueller and Renken 2017) such enablement of individuals will likely contribute positively to future transformation success.

In terms of our objective to better understand how organizations quickly appropriate the benefits of new or changed IT, our results reveal an important nuance. While our theoretical contributions are instrumental in helping both scholars and practitioners understand why the situation at PSS and SSG escalated, and why achieving effective use was so much more complex for them, the literature already recognizes that achieving representational fidelity is often rather complicated. In a somewhat counterintuitive interpretation, the most revealing insight our findings provide is not how and why the situation at PSS and SSG escalated, but that the PSN clerks achieved effective use so quickly. Through the qualifying effect exerting representational complexity has on both learning

effective use and the very concept of representation, our findings direct attention to situations in which users can achieve effective use far more quickly than current literature suggests. From a practical point of view, managers of digitalization projects can now identify and highlight areas in which a new system will generate benefits swiftly, allowing managers to structure and manage their programs to achieve quick wins and generate lighthouse projects.

Furthermore, we find that our analyses are in line with what we know about the common dip in performance and productivity in enterprise system projects (e.g., Markus 2004; Shang and Seddon 2002). Our insights provide added support for the escalation of problems occurring some time after the go-live. In our case, only after all the groups had managed to achieve transparent interaction a couple of weeks after the go-live did the increased complexity at PSS and SSG become salient, although up to that point the clerks in these two departments were doing as well as their PSN colleagues in terms of learning effective use. This discovery calls for more phased training that will help users address issues related to transparent interaction before the go-live, while revisiting the challenge of achieving representational fidelity *in situ* after the go-live.

Conclusion

Our case study at BANK led to two new concepts—system dependency and semantic dependency—which characterize system-enabled tasks. These concepts were instrumental in recognizing the kind of impact the co-dependency between tasks and systems has on effective use via *exerting representational complexity*, a mechanism that generates an effect that counteracts *learning effective use*. The crux of the effect exerted by representational complexity is that the effort associated with cognitive demands and the corresponding actions required for learning a new system depend not only on the properties of the new system or those of the task at hand, but also strongly on the emergent properties of the system-enabled task.

We acknowledge that in addition to its contributions and implications, our study has limitations. First, in line with critical realism and case study research (Lee and Baskerville 2003; Wynn and Williams 2012), our study is generalizable only analytically. However, our case is typical of system-enabled organizational transformation in a corporate information systems context. We are confident that the concepts and mechanism we propose also apply to similar projects. Second, our case study allowed observation of just one specific system (i.e., LMS), which implies that the system properties remained constant. While this limitation helped us focus on the emergent properties of system-enabled tasks, future work should study the impacts of systems with different properties. In a similar vein, we acknowledge that systems like LMS are rarely used comprehensively (Yamauchi and Swanson 2010) and that different tasks might require different sets of features (e.g., Sun 2012), especially over time (Benlian 2015). Third, future work should further develop our two novel concepts, since we observed only relative differences in system dependency and semantic dependency. In effect, we were able to observe only the ways in which representational complexity is exerted as a relative concept between PSN and PSS/SSG. More comprehensive efforts introducing starker differences in tasks and systems can appraise the underlying dimensions better, enabling operationalization and empirical testing of our concepts.

These limitations notwithstanding, our work contributes to research beyond shallow system usage conceptualizations (Burton-Jones and Grange 2013; Burton-Jones and Straub 2006) by suggesting system dependency and semantic dependency, which capture how and why representational complexity arises in use. This knowledge allows us to advance the understanding of the impediments users must overcome in their efforts to use a system effectively.

The future will bring accelerated and more intense digital transformations. These transformations will be fueled by information systems that are more innovative and powerful than are current enterprise systems (Brynjolfsson and McAfee 2014). They are expected to allow for more dynamic links between tasks and systems and will require even more socio-cognitive sensemaking of users (Nambisan et al. 2017). Increased managerial attention to representational complexity will be essential, because the successful transformation of tasks with a new technology depends increasingly on individuals' learning capabilities and on providing adequate organizational support. Our work shows that simply providing additional resources to manage this process is inadequate. Rather, tailoring change and training measures to employees' system-enabled tasks and corresponding needs will help achieve effective use of new technologies, allowing companies to reap their benefits earlier and successfully transform themselves for the digital age.

Acknowledgments

We are thankful for BANK's support of our work and especially thank the many individuals that have taken time to participate in our work. We are grateful for the comments provided on earlier versions of this paper at the 2014 International Conference on Information Systems and during presentations at Bentley University, Maynooth University, the University of Sydney, the University of

Queensland, the Australian National University, Queensland University of Technology, and the University of Groningen. In particular, we are grateful for Albert Boonstra's many comments and suggestions. Above all, we are indebted to our editors and reviewers whose critical reading of our work and genuine interest in what we wanted to say has helped this paper become so much more than we initially imagined. Their dedication served as motivation and will continue to inspire us far beyond this paper.

This work was partially supported by the Banking Enterprise Systems Center of Competence of the Institute for Enterprise Systems, University of Mannheim, Germany, and the Research Alliance ForDigital funded by the Ministry of Science, Research and the Arts (MWK) in Baden-Württemberg, Germany.

References

Archer, M. S. 1995. *Realist Social Theory: The Morphogenetic Approach*, Cambridge, UK: Cambridge University Press.

Avgerou, C. 2013. "Social Mechanisms for Causal Explanation in Social Theory Based IS Research," *Journal of the Association for Information Systems* (14:8), pp. 399-419.

Beese, J., Aier, S., Haki, K., and Aleatrati Khosroshahi, P. 2016. "Drivers and Effects of Information Systems Architecture Complexity: A Mixed-Methods Study," in *Proceedings of the 24th European Conference on Information Systems*, Istanbul, Turkey.

Benlian, A. 2015. "IT Feature Use over Time and Its Impact on Individual Task Performance," *Journal of the Association for Information Systems* (16:3), pp. 144-173.

Bhaskar, R. 2008. *A Realist Theory of Science*, New York: Routledge.

Brooks, F. P. 1987. "Essence and Accidents of Software Engineering," *IEEE Computer* (20:4), pp. 10-19.

Brynjolfsson, E., and McAfee, A. 2014. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*, New York: W. W. Norton & Company.

Burton-Jones, A., and Grange, C. 2013. "From Use to Effective Use: A Representation Theory Perspective," *Information Systems Research* (24:3), pp. 632-658.

Burton-Jones, A., Recker, J., Indulska, M., Green, P., and Weber, R. 2017. "Assessing Representation Theory with a Framework for Pursuing Success and Failure," *MIS Quarterly* (41:4), pp. 1307-1333.

Burton-Jones, A., and Straub, D. W. 2006. "Reconceptualizing System Usage: An Approach and Empirical Test," *Information Systems Research* (17:3), pp. 228-246.

Burton-Jones, A., and Volkoff, O. 2017. "How Can We Develop Contextualized Theories of Effective Use? A Demonstration in the Context of Community-Care Electronic Health Records," *Information Systems Research* (28:3), pp. 468-489.

Campbell, D. J. 1988. "Task Complexity: A Review and Analysis," *Academy of Management Journal* (13:1), pp. 40-52.

Corbin, J., and Strauss, A. 1994. "Grounded Theory Methodology: An Overview," Chapter 17 in *Handbook of Qualitative Research*, N. K. Denzin and Y. S. Lincoln (eds.), Thousand Oaks, CA: SAGE, pp. 273-285.

Danermark, B., Ekström, M., Jakobsen, L., and Karlsson, J. C. 2002. *Explaining Society: Critical Realism in the Social Sciences*, New York: Routledge.

DeSanctis, G., and Poole, M. S. 1994. "Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory," *Organization Science* (5:2), pp. 121-147.

Devadoss, P., and Pan, S. 2007. "Enterprise Systems Use: Towards a Structurational Analysis of Enterprise Systems Induced Organizational Transformation," *Communications of the Association for Information Systems* (19:17), pp. 351-385.

Elder-Vass, D. 2005. "Emergence and the Realist Account of Cause," *Journal of Critical Realism* (4:2), pp. 315-338.

Gnewuch, U., Haake, P., Mueller, B., and Maedche, A. 2016. "The Effect of Learning on the Effective Use of Enterprise Systems," in *Proceedings of the 36th International Conference on Information Systems*, Dublin, Ireland.

Goodhue, D., and Thompson, R. 1995. "Task–Technology Fit and Individual Performance," *MIS Quarterly* (19:2), pp. 213-236.

Hackman, J. R. 1969. "Toward Understanding the Role of Tasks in Behavioral Research," *Acta Psychologica* (31), pp. 97-128.

Hærem, T., Pentland, B. T., and Miller, K. D. 2015. "Task Complexity: Extending a Core Concept," *Academy of Management Review* (40:3), pp. 446-460.

Klein, K. J., Dansereau, F., and Hall, R. J. 1994. "Levels Issues in Theory Development, Data Collection, and Analysis," *Academy of Management Review* (19:2), Academy of Management, pp. 195-229.

Kolb, A. Y., and Kolb, D. A. 2009. "Experiential Learning Theory: A Dynamic, Holistic Approach to Management Learning, Education and Development," *The SAGE Handbook of Management Learning, Education and Development*, S. J. Armstrong and C. V. Fukami (eds.), Thousand Oaks, CA: SAGE, pp. 42-68.

Lee, A. S., and Baskerville, R. L. 2003. "Generalizing Generalizability in Information Systems Research," *Information Systems Research* (14:3), pp. 221-243.

Leiser, S., and Hirschheim, R. 2007. "Fighting against Windmills: Strategic Information Systems and Organizational Deep Structures," *MIS Quarterly* (31:2), pp. 327-354.

Leonardi, P. M. 2011. "When Flexible Routines Meet Flexible Technologies: Affordance, Constraint, and the Imbrication of Human and Material Agencies," *MIS Quarterly* (35:1), pp. 147-168.

Liang, H., Peng, Z., Xue, Y., and Guo, X. 2015. "Employees' Exploration of Complex Systems: An Integrative View," *Journal of Management Information Systems* (32:1), pp. 322-357.

Markus, M. L. 2004. "Technochange Management: Using IT to Drive Organizational Change," *Journal of Information Technology* (19:1), pp. 4-20.

Markus, M. L., and Tanis, C. 2000. "The Enterprise System Experience—From Adoption to Success," in *In Framing the Domains of IT Management: Projecting the Future —Through the Past*, R. W. Zmud (ed.), Cincinatti, OH: Pinnaflex Educational Resources, pp. 173-207.

Mingers, J., and Standing, C. 2017. "Why Things Happen—Developing the Critical Realist View of Causal Mechanisms," *Information and Organization* (27:3), pp. 171-189.

Morton, P. 2006. "Using Critical Realism to Explain Strategic Information Systems Planning," *Journal of Information Technology Theory and Application* (8:1), pp. 1-20.

Mueller, B., and Renken, U. 2017. "Helping Employees to Be Digital Transformers—The Olympus.Connect Case," in *Proceedings of the 38th International Conference on Information Systems*, Seoul, South Korea.

Nambisan, S., Lyytinen, K., Majchrzak, A., and Song, M. 2017. "Digital Innovation Management: Reinventing Innovation Management Research in a Digital World," *MIS Quarterly* (41:1), pp. 223-238.

Sayer, A. 1992. *Method in Social Science* (2nd ed.), New York: Routledge.

Schultze, U., and Avital, M. 2011. "Designing Interviews to Generate Rich Data for Information Systems Research," *Information and Organization* (21:1), pp. 1-16.

Shang, S., and Seddon, P. B. 2002. "Assessing and Managing the Benefits of Enterprise Systems: The Business Manager's Perspective," *Information Systems Journal* (12:4), pp. 271-299.

Sharma, R., and Yetton, P. 2007. "The Contingent Effects of Training, Technical Complexity, and Task Interdependence on Successful Information Systems Implementation," *MIS Quarterly* (31:2), pp. 219-238.

Sun, H. 2012. "Understanding User Revisions When Using Information System Features: Adaptive System Use and Triggers," *MIS Quarterly* (36:2), pp. 453-478.

Volkoff, O., and Strong, D. 2013. "Critical Realism and Affordances: Theorizing IT-Associated Organizational Change Processes," *MIS Quarterly* (37:3), pp. 819-834.

Volkoff, O., Strong, D., and Elmes, M. 2007. "Technological Embeddedness And Organizational Change," *Organization Science* (18:5), pp. 832-848.

Wand, Y., and Weber, R. 1995. "On the Deep Structures of Information Systems," *Information Systems Journal* (5:3), pp. 202-223.

Wood, R. E. 1986. "Task Complexity: Definition of the Construct," *Organizational Behavior and Human Decision Processes* (37:1), pp. 60-82.

Wynn, D., and Williams, C. 2012. "Principles for Conducting Critical Realist Case Study Research in Information Systems," *MIS Quarterly* (36:3), pp. 787-810.

Yamauchi, Y., and Swanson, E. B. 2010. "Local Assimilation of an Enterprise System: Situated Learning by Means of Familiarity Pockets," *Information and Organization* (20:3-4), pp. 187-206.

About the Authors

Jens Lauterbach is an independent advisor for digital and organizational transformation projects. He holds a diploma in Business Information Systems from the University of Bamberg, Germany, a master's degree in business administration from the Western Illinois University, USA, and a Ph.D. from the University of Mannheim, Germany. Jens mainly works at the intersection of business and information technology and helps organizations to establish structures that lead to the effective implementation and use of digital technologies. He has served as reviewer and published in leading IS conferences.

Benjamin Mueller is an associate professor of Digital Innovation and Design at the University of Lausanne, Switzerland, and an associate researcher at the Karlsruhe Institute of Technology, Germany. He holds master's degrees in business and information systems from the EBS Business School, Germany, and Georgia State University, USA, and obtained his doctoral degree from EBS Business School. His work focuses on how advanced information and communication technologies transform organizations, paying particular attention to mechanisms through which individuals augment their work with technology and the resultant organizational benefits. Benjamin's work is published in journals such as *Journal of Management Information Systems*, *Journal of the Association for Information Systems,* and *Journal of Business Research.*

Felix Kahrau is an independent business consultant and head of information security consulting at a medium-sized cyber security service provider. He earned his Ph.D. in Business Administration from the University of Mannheim. His research interests include digital transformation and cyber security with particular emphases on the interplay of technology, organizational structures, and work practices and the specific challenges and needs of small and medium enterprises.

Alexander Maedche is a professor of Information Systems at Karlsruhe Institute of Technology, Germany. His work is positioned at the intersection of IS and human–computer interaction. He is specifically interested in creating impactful scientific knowledge for designing interactive intelligent systems that enable humans in organizations and society to perform activities more efficiently, effectively, and meaningfully. He publishes in leading IS and computer science conferences and journals, such as *Journal of the Association of Information Systems, Computers & Human Behavior, International Journal of Human-Computer Studies,* and *IEEE Transactions on Software Engineering*. Alexander is senior editor for *Journal of the Association for Information System*, department editor for *Business & Information Systems Engineering* and has served as program co-chair and track chair for major IS conferences.