Journal of the Association for Information Systems (2020) **21**(3), 664-694
doi: 10.17705/1jais.00616

RESEARCH ARTICLE

ISSN 1536-9323

Monitoring the Complexity of IT Architectures: Design Principles and an IT Artifact

Thomas Widjaja[1], Robert Wayne Gregory[2]
[1]University of Passau, Germany, thomas.widjaja@uni-passau.de
[2]University of Virginia, USA, rg7cv@comm.virginia.edu

Abstract

Monitoring the complexity of a firm's IT architecture is imperative to ensure a stable and flexible platform foundation for competing in the era of digital business strategy. However, IT architects lack IT support for dealing with this important problem. We engaged with five companies in a significant design science research (DSR) program and drew on the heuristic theorizing framework both to solve this problem through evolving IT artifacts and to accumulate nascent design knowledge. We base the design knowledge development on a conceptual framework involving three essential concepts for understanding and solving this problem: structural complexity, dynamic complexity, and problem-solving complexity. Drawing on this foundation, we address the research question: How can IT support be provided for reducing the problem-solving complexity of monitoring the structural and dynamic complexity of IT architectures in the context of a digital business strategy? To answer this question, we present a set of design principles that we derived from our iterative process of IT artifact construction and evaluation activities with five companies. Our nascent design knowledge contributes to the research on IT architecture management in the context of digital business strategy. In addition, we also contribute to the understanding of how, through the use and illustration of the heuristic theorizing framework, design knowledge can be accumulated systematically on the basis of generalization from IT artifact construction and evaluation outcomes generated across multiple contexts and companies.

Keywords: IT Architecture Complexity, Monitoring Complex Systems, Digital Business Strategy, Design Science Research, Heuristic Theorizing

Robert Winter was the accepting senior editor. This research article was submitted on November 30, 2017, and underwent three revisions.

1 Introduction

In today's digital era, firms must adapt continuously to quickly changing customer demands and often pursue digital business strategies, which are defined as organizational strategies formulated and executed by leveraging digital resources to create differential value (Bharadwaj et al., 2013). However, the historically developed complexity of established firms' IT architectures represents a significant cause of inertia for such digital transformations (e.g., Boh & Yellin, 2006; Boyle, Keywood, & Roberts, 2012; Guillemette & Paré, 2012). Monitoring this architectural complexity is critical for the success of digital platform initiatives that involve the construction of a stable core that enables efficient yet flexible solutions at the periphery (de Reuver, Sørensen, & Basole, 2018; Gregory et al., 2015). The problem domain focus of our design science work is monitoring the complexity of IT architectures in firms; the solution domain focus is IT support for IT architects in this respect. At a more general level, the focus of this study is the

accumulation of design knowledge about monitoring complex systems.

Monitoring architectural systems complexity is particularly relevant in today's era of digital business strategy, in which building a digital platform has become a primary concern for the top leadership of firms. Research related to building digital platforms has begun to distinguish between heavyweight and lightweight IT (Bygstad, 2017), infrastructural stability and change (Tilson, Lyytinen, & Sørensen, 2010), and a platform core and peripheral ecosystem (Wareham, Fox, & Cano Giner, 2014). A related distinction in the literature on the complexity of IT architectures (and other instances of systems of systems) is between structural and dynamic complexity (Henneman & Rouse, 1986; Schneberger & McLean, 2003; Xia & Lee, 2005). Structural complexity concerns the relatively stable form and function of the IT architecture and, thus, needs to be monitored to ensure the stability of the heavyweight IT and platform core. Dynamic complexity, in contrast, refers to the uncertain, unpredictable, and often ambiguous nature and rate of change of the IT architecture, and should be monitored to enable change in the lightweight IT within the surrounding platform ecosystem.

Monitoring the structural and dynamic complexity of the firm's IT architecture is important because key digitized products, services, and processes are symbolically represented in the firm's IT architecture, and changes in one have a direct impact on the other. However, monitoring architectural complexity has also become an extremely difficult problem to solve. We suggest that an important issue that has been overlooked in the prior literature on IT architectures (Beese et al., 2016; Richardson, Jackson, & Dickson, 1990; Ross et al., 2006; Schilling et al., 2017; Tilson et al., 2010; Tiwana & Konsynski, 2010) is problem-solving complexity, which concerns human reasoning, attentional resources, skills, and the overall ability to cope with structural and dynamic complexity in the search for a satisficing problem solution (Endsley, 1995; Henneman & Rouse, 1986; Lerch & Harter, 2001; Simon, 1996). In fact, one reason scholars have called for reducing the structural and dynamic complexity of IT architectures is its direct effect on problem-solving complexity (Schneberger & McLean, 2003). This line of reasoning, however, with its focus on the reduction of complexity, overlooks the value offered by IT artifacts (i.e., tools) in providing cognitive support for the monitoring of systems complexity (Lerch and Harter 2001), thereby reducing problem-solving complexity.

We address the following research question: How can IT support be provided for reducing the problem-solving complexity of monitoring the structural and dynamic complexity of IT architectures in the context of a digital business strategy? Addressing this question is both novel and important because the answer will contribute to our understanding of how to reduce the problem-solving complexity IT architects face in contemporary digital business strategy execution and transformation initiatives. Pursuing a digital business strategy and creating differential value "requires effective sensemaking and the ability to cope with complexity and uncertainty" (Woodard et al., 2013, p. 558).

In addition to this strong motivation for further work in the problem domain of this paper, we also identified a significant gap in the solution domain. While previous work has attempted to solve the problem of cognitive IT support (Lerch and Harter 2001), the focus has been on a different problem class, and the experimental study design did not involve IT artifact construction. Our work addresses the identified gaps in the problem and solution spaces by focusing on the evolution and accumulation of design knowledge, which, in our case, has matured through iterative projection and concurrent evaluation across multiple contexts and companies.

We engaged with five companies for IT artifact construction and evaluation activities over an eight-year period. Our work is guided by heuristic theorizing (Gregory & Muntermann, 2014), a framework for theorizing in the problem-driven DSR tradition (Iivari, 2015; Sein et al., 2011) that draws on the sciences of the artificial (Simon, 1996) and Hevner et al.'s (2004) description of "design as a search process." In our DSR program, we solved instantiations of a general problem class we refer to as the "monitoring the complexity of IT architectures" (MCITA) problem class. In particular, this involved constructing and concurrently evaluating the five expository artifacts of our emergent set of four design principles. To build the MCITA design principles during the heuristic theorizing process, we fundamentally revisited and reformulated our understanding of the problem at hand, which resulted in three heuristic theorizing cycles focusing on nested problem understandings: the problems of (1) making "optimal" IT standardization decisions, (2) assessing the "desirable" degree of IT heterogeneity, and (3) monitoring the complexity of IT architectures (MCITA).

We contribute to the research on IT architectures (Beese et al., 2016; Richardson et al., 1990; Ross et al., 2006; Schilling et al., 2017; Tilson et al., 2010; Tiwana & Konsynski, 2010) with a set of design principles that, in the context of digital business strategy (Bharadwaj et al., 2013; Drnevich & Croson, 2013; Keen & Williams, 2013; Woodard et al., 2013), address the class of problems we refer to as MCITA. We add this class and solution to the general body of design knowledge on monitoring complex systems (Becz et al., 2010; Domerçant & Mavris, 2011;

Henneman & Rouse, 1986; Maier, 1998; Natu et al., 2016; Psiuk & Zielinski, 2015). The distinctions between structural, dynamic, and problem-solving complexities are an important extension of the literature since it is clear that reducing problem-solving complexity is desirable, whereas structural and dynamic complexity may have to be embraced, for example, even to allow for a sufficient variety of IT functionalities in the context of a digital business strategy to respond to changes in customer demands through new product development.

2 Conceptual Background

2.1 IT Architecture in the Context of a Digital Business Strategy

An IT architecture is defined as the "organizing logic for applications, data, and infrastructure technologies" (Ross, 2003, p. 32).[1] Prior research on the contribution of IT in organizations often suggests that the structural and dynamic complexity of IT architectures should be reduced (e.g., Ahlemann et al., 2012; Guillemette & Paré, 2012; Tamm et al., 2011). According to prior IT architecture management research, a key challenge is to create an overall design and architectural blueprint for structuring the firm's collectivity of IT systems. Studies pursuing this line of thought have suggested various means to tame IT architectural complexity. For example, sorting IT components into different architectural layers has been recommended, resulting in the concept of a multilayered architecture (Richardson et al., 1990; Winter & Fischer, 2007). Prior research has also proposed drawing on the analogy of city planning (Schmidt & Buxmann, 2011) to improve the architectural planning, mapping, and designing of the blueprint that is the guide and standard for any type of IT implementation in the firm (Boh & Yellin, 2006). Overall, a key recurring theme in IT architecture management research is the need to limit IT complexity to achieve organizational benefits such as IT efficiency and IT agility (Guillemette & Paré, 2012; Tamm et al., 2011). We argue that in view of the increase in environmental dynamism and complexity associated with the rise of digital business strategies (Bharadwaj et al., 2013; El Sawy et al., 2010; Tanriverdi, Rai, & Venkatraman, 2010; Woodard et al., 2013), the sole focus on limiting complexity should be revisited.

The contribution of IT architectures in organizations has evolved fundamentally with the rise of the digital business strategy, defined as a business strategy formulated and executed by leveraging IT to create differential value (Bharadwaj et al., 2013; Drnevich & Croson, 2013; Woodard et al., 2013). In this new context, a firm's IT architecture provides the platform for its digital business strategy (de Reuver et al., 2018) by, for instance, defining technical interfaces for customers, partners, and suppliers and by setting standards that determine degrees of freedom for digital business moves (Keen & Williams, 2013). As an increasingly heterogeneous and distributed set of actors draws on these IT architectures, the architectures also continuously evolve (Tanriverdi et al., 2010; Woodard et al., 2013; Yoo et al., 2012). Furthermore, the shift toward more distributed control and greater autonomy of internal and external actors suggests that IT architectures in firms pursuing digital business strategies resemble digital infrastructures (Hanseth & Lyytinen, 2010; Tilson et al., 2010), which are instances of complex systems of systems (Amaral & Uzzi, 2007). An IT system consists of IT components and the relationships among those IT components (Hall & Fagen, 1969), and a "system of IT systems" consists of interconnected IT systems (Simon, 1962). We thus define IT architecture in the context of digital business strategy as a distributed, evolutionary, and emergent system of IT systems (Maier, 1998; Sommerville et al., 2012).

2.2 Structural, Dynamic, and Problem-Solving Complexity

In the preceding section, we highlighted that IT architectures share key characteristics with digital infrastructures (Henfridsson & Bygstad, 2013; Tilson et al., 2010). Although their degree of openness is certainly not comparable to, for example, the internet (Hanseth & Lyytinen, 2010), the IT architectures of firms pursuing a digital business strategy do evolve at a rapid rate of change through a heterogeneous set of participating internal and external actors. At the same time, however, the IT architectures of firms must also provide a stable foundation for organizational integration and control (Berente et al., 2016). The result is that firms' IT architectures exhibit forms of emergent behavior that are known from complex systems (Amaral & Uzzi, 2007), highlighting

[1] The literature distinguishes between the concepts IT architecture, enterprise architecture, and solution architecture. In this paper, we focus on IT architecture, which is a subsystem of an enterprise architecture; the latter is defined as "organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the company's operating model" (Ross et al., 2006, p. 47). Whereas both enterprise architecture and IT architecture consider the organization as a whole, a solution architecture focuses on a single project or subsystem and the "fundamental decisions in the design of a specific solution" (Greefhorst & Proper, 2011, p. 25).

complexity as an overarching characteristic of the IT architectures.

Our definition of IT architecture in Section 2.1 is based on the systems-of-systems perspective (DeLaurentis & Callaway, 2004). The information systems literature on systems of systems distinguishes between structural complexity and dynamic complexity (Schneberger & McLean, 2003; Xia & Lee, 2005). However, the literature on monitoring the structural and dynamic complexity of systems emphasizes the concept of problem-solving complexity (Henneman & Rouse, 1986). We propose the use of an IT artifact (i.e., a monitoring tool) to reduce problem-solving complexity (i.e., the complexity in the cognitive realm) and enable companies to embrace structural and dynamic complexity (i.e., complexity in the material realm) resulting from heterogeneous and distributed actors continually enlarging, reducing, or modifying the IT architectures in the context of digital business strategies (see Figure 1).

Structural complexity. The relatively stable form and function of an IT architecture is captured by the concept of structural complexity, which is typically conceptualized and measured in terms of the number and variety of system components and relations involved (Henneman & Rouse, 1986; Ribbers & Schoo, 2002; Schneberger & McLean, 2003; Xia & Lee, 2005). This understanding of structural complexity is rooted in studies of complex systems. Herbert A. Simon defined a complex system as one composed of a large number of subsystems that interact in a "nonsimple way" (Simon, 1962). This view of structural complexity highlights that complexity, in large part, stems from the nature of the interactions between different parts of a system.

Modular systems theory focuses precisely on the nature of the interactions between the parts of a complex system and uses the concept of modularity to describe the degree to which a system's subsystems can be separated, changed, or recombined through standardized interfaces (Ethiraj & Levinthal, 2004; Sanchez & Mahoney, 1996; Schilling, 2000). Modular systems design is based on the idea of creating highly integrated, mutually responsive, and tightly coupled subsystems that preserve a stable form and function, while simultaneously enabling flexibility for their use and recombination based on loose couplings and standardized interfaces between different modules (Baldwin & Clark, 2000). Modularity has been identified in the literature as a key facet of an IT architecture's form and state (Tiwana & Konsynski, 2010) and is important for understanding structural complexity. It is also important to the concept of dynamic complexity.

Dynamic complexity. The concept of dynamic complexity captures the continuous evolution, elaboration, and change of an IT architecture's form and function. Dynamic complexity concerns the uncertain, unpredictable, and often ambiguous nature and rate of change in the number and variety of components and the relationships among them over time (Henneman & Rouse, 1986; Ribbers & Schoo, 2002; Schneberger & McLean, 2003; Xia & Lee, 2005). This conceptualization and measurement of dynamic complexity highlight the interrelationship with the concept of structural complexity. In particular, dynamic complexity focuses on changes from one state of structural complexity in Time 1 to a new state of structural complexity in Time 2.

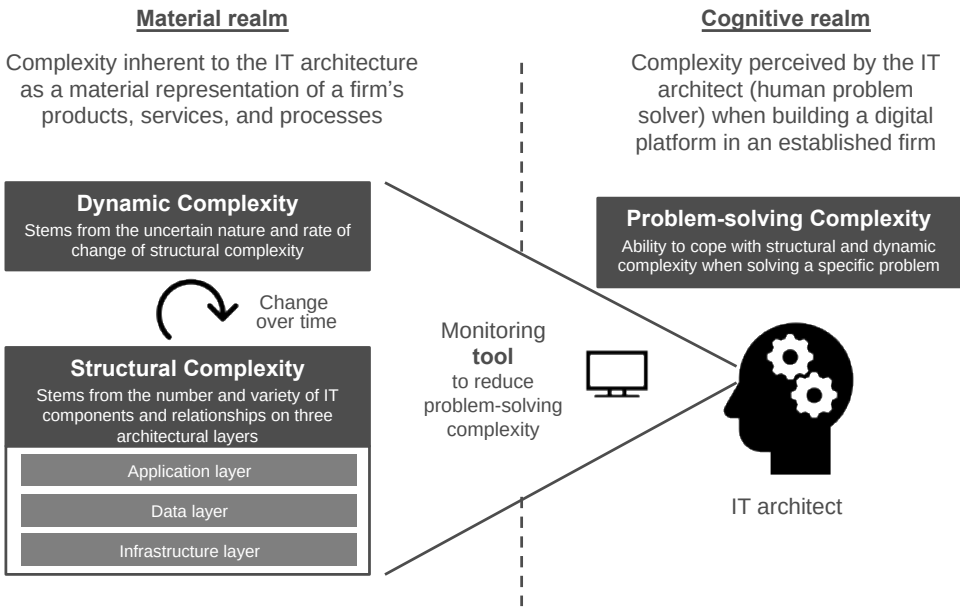

Figure 1. Conceptual Model

Problem-solving complexity. The literature on monitoring complex systems distinguishes between structural and dynamic complexity on the one hand and problem-solving complexity on the other hand (Henneman & Rouse, 1986). Problem-solving complexity concerns human reasoning, attentional resources, skills, and the overall ability to cope with structural and dynamic complexity in the search for a satisficing problem solution (Endsley, 1995; Henneman & Rouse, 1986; Lerch & Harter, 2001; Simon, 1996).

As illustrated by a neurophysiologist's perspective on a sheep's brain compared to a butcher's (Ashby, 1960), the needs for monitoring and capacities for understanding structural and dynamic complexity differ based on the perspective as well as the ability to decompose a system into modular subsystems. Thus, although a system itself may be inherently complex in terms of its compositional structure and evolution over time, this does not necessarily mean that solving a particular problem for that system is perceived to be difficult. As Schneberger and McLean (2003) noted, we are to a considerable extent interested in understanding structural and dynamic complexity because the "incredible capabilities and opportunities computing offers us" may be "destined to become so complex as to overwhelm human ability to cope with it" (p. 216). This means that the problem-solving complexity of a given problem can be reduced either by reducing the structural and dynamic complexity or by tool support.

2.3 Problem Requirements of Monitoring Complex Systems

To derive specific problem requirements that guide our design and tool construction work, we reviewed a variety of instances of the general problem class of monitoring complex systems. In addition to IT architectures, other instances we identified in the literature include large-scale networked IT systems such as communication infrastructures, hybrid clouds, self-adaptive software, and peer-to-peer networks (Henneman & Rouse, 1986; Murray & Liu, 1997; Murray & Yili, 1994; Natu et al., 2016; Psiuk & Zielinski, 2015; Vierhauser, Rabiser, & Grünbacher, 2016; Zinser & Henneman, 1989); military systems such as air defense networks and military aircraft fleets (Becz et al., 2010; Domerçant & Mavris, 2011; Maier, 1998; Tamaskar, Neema, & DeLaurentis, 2014); and national transportation systems (DeLaurentis & Callaway, 2004). Appendix Table A1 provides an overview of selected design and tool construction work. What we learned from this review of design knowledge is that the distinction between tracking and comprehending is useful for our understanding of the MCITA problem class that this paper addresses.

Tracking (Type 1 monitoring). At the elementary level of identifying, screening, describing, detecting, and tracing the complexity of IT architectures, IT support for Type 1 monitoring must address the requirement of tracking key variables associated with structural complexity (including the number and variety of system components and relationships among them). To monitor the dynamic complexity, these variables related to structural complexity must be traced and tracked over time.

Comprehending (Type 2 monitoring). To reduce problem-solving complexity and provide a useful cognitive aid, IT support for monitoring complex systems must go beyond tracking (Type 1 monitoring) to also support the comprehension (Type 2 monitoring) of structural and dynamic complexity. Type 2 monitoring builds on information yielded through Type 1 monitoring and focuses on higher-order complexity monitoring processes, including diagnosing, simulating, understanding, and giving meaning to the complexity of IT architectures as an input to decision-making.

3 Research Design

We conducted design science research, defined here as a problem-solving process involving a heuristic search to identify a relevant problem class and generate prescriptive knowledge (e.g., a set of design principles) for the design of artifacts (Gregor & Hevner, 2013; Hevner et al., 2004; Peffers et al., 2007) that addresses the metarequirements of the identified problem class. Hevner et al. (2004) describe artifacts as "innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished" (p. 83). Artifacts exist on different levels of abstraction, including the level of artifact instantiations, design principles that provide prescriptive guidance in the construction of artifacts, and the most abstract level of design theories (Gregor & Hevner, 2013; Gregor & Jones, 2007).

In this paper, we focus on the process of evolution and accumulation of design knowledge at the midrange level of design principles. Our process involved identifying the problem class and generating abstracted design principles through iterative projection and concurrent evaluation of instantiated IT artifacts (software prototypes) across multiple contexts and companies. In the typology of Iivari (2015) for DSR strategies, our research project corresponds to "DSR Strategy 2" and is based on close researcher-practitioner relationships and teams with mutual involvement, engagement, and exchange of DSR teams (see also Sein et al., 2011). We engaged five large companies in our DSR program, resulting in eight years of intense practitioner interaction over the entire program lifetime. Table 1 provides details about the participating companies and the specific design activities conducted in collaboration with them.

Table 1. Companies Engaged and Problem-Solving Activities

Company and Time period	Problem Context	Problem Solving Activities and Collected Evidence
Heuristic Theorizing Cycle 1		
Company 1 04/2008- 01/2009	Trading company (>15,000 employees) that confronted a major redesign and transformation of its IT systems environment. Cooperation with the chief IT architect and a consulting company	• Development of Tool A (with a team of four developers) • Two interviews with a consultant contracted by this firm to support the IT transformation (~ 60 min.) • Workshop with chief IT architect and consultant to apply the prototype and gather input parameters (~ 120 min.) • Informal feedback from the CIO on the organizational need and value of Tool A • Telephone interview with IT architect to assess the usefulness of Tool A (~ 30 min.) • Secondary data: Access to architectural domain model as well as cost and interface data for selected IT components on the application layer
Heuristic Theorizing Cycle 2		
Company 2 08/2009- 02/2013	Large-scale enterprise (>40,000 employees) that implemented a new sourcing strategy during IT transformation. Cooperation with the CIO and an IT architect	• Two meetings with the principal IT architect to understand the scope and define the problem (~ 90 min.) • Feedback from presenting the application of Tool B (developed with Company 3) to the CIO and an IT architect as well as two further results presentations to user groups within the company (~ 60 min.) • Secondary data: Access to data about 147 software components and the architectural domain model
Company 3 02/2010- 08/2015	Government organization (>100,000 employees) that was redesigning its corporate digital infrastructure. Cooperation with the CIO, different line managers, and the chief IT architect	• Development of Tool B (with a team of three developers) • Feedback from three workshops with IT architects and line managers to improve our joint understanding of the problem to solve (~ 60 min.) • Feedback from one presentation about applications of Tool B to the CIO and the chief IT architect (~ 60 min.) • Secondary data: Team member of project "strategic management of heterogeneous IT landscapes" with the task to analyze cost and benefits of heterogeneous IT systems; granted access to complete project documentation; access and analysis of detailed profiles of >206 IT components on the application layer; access and analysis of data of 752 IT components on the infrastructure layer
Heuristic Theorizing Cycle 3		
Company 4 06/2012- 10/2016	International bank (>50,000 employees) currently implementing company-wide complexity management to guide IT transformation. Cooperation with two leading IT architects and the department head of the IT architecture management	• Development of Tool C (with a team of seven developers and involving three iterations of software development) • Seven meetings with the development team of Tool C (~ 90 min.) • Two interviews (~ 90 min.) and notes from two workshops (120 min.) with IT architects during the development of Tool C • One interview with a lead user after presentation of Tool C (~ 60 min.) • Development of Tool E (with a team of two developers and involving two iterations of software development) • Final workshop with two leading IT architects and department head (~ 60 min.) • Two interviews after presenting Tool E to the two leading architects (~ 60 min.) • Secondary data: access to internal presentations, architectural domain model, historical information on > 4250 IT components, documentation of the complexity management initiative
Company 5 01/2013- 08/2015	International bank (>90,000 employees) currently implementing a new digitized platform. Cooperation with the senior IT transformation manager, users of the tool, and two specialized IT architects	• Development of Tool D (with a team of eight developers and involving two iterations of software development) • Six meetings with the development team of Tool D (~ 90 min.) • Two interviews with a senior IT transformation manager (~ 60 min.) • Notes from two meetings with the IT architects during the development of Tool D (~ 60 min.) • Three workshops to evaluate Tool D with IT architects and users (~ 60 min.) • Secondary data: access to internal presentations, architectural domain model, historical information on IT components on all layers of the IT architecture

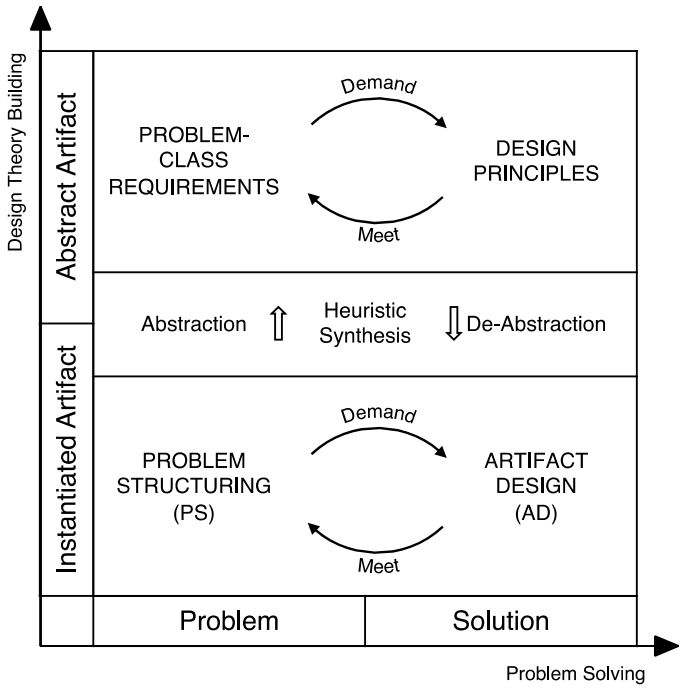

Figure 2. Heuristic Theorizing Framework

We identified Gregory and Muntermann's (2014) framework for heuristic theorizing as a useful lens to structure and explain our complex process of IT artifact evolution and design knowledge accumulation. By providing a simple structuring (see Figure 2) of design science research activities across problem and solution spaces, on the one hand, and different levels of abstraction, on the other hand, this framework helped us create a detailed yet accessible narrative that provides transparency into our journey of design knowledge evolution and accumulation (see Section 5). During our design science research journey, we conducted three heuristic theorizing cycles.

4 Findings

4.1 Problem Space

In the problem space, our search process resulted in a nested structure of problem classes (see Figure 3). We focused on the standardization problem in the first phase of our research (heuristic theorizing Cycle 1). The second heuristic theorizing cycle concerned the solution of the heterogeneity problem (which encompasses the standardization problem), and the third and final heuristic theorizing cycle focused on the problem of reducing the problem-solving complexity of monitoring the structural and dynamic complexity of IT architectures (MCITA) (see theoretical background section). More specifically, based on the assumption of bounded rationality (Simon, 1991), our study focuses on the design of cognitive IT support (Lerch & Harter, 2001) to reduce problem-solving complexity and aid decision makers in monitoring the structural and dynamic complexity of IT architectures. To reduce problem-solving complexity and provide a cognitive aid, the design of IT support for MCITA should address the two problem requirements (Walls, Widmeyer, & El Sawy, 1992) of tracking and comprehending structural and dynamic complexity (see Section 2.3). Following the ensemble view of IT (Gregor & Jones, 2007; Orlikowski & Iacono, 2001; Sein et al., 2011), addressing the specific problem requirements for IT design is not an end in itself, but any such effort is embedded in a broader social and organizational context. The context that we focus on in this design study is the organizational contribution of monitoring the complexity of IT architectures to accommodate the competing concerns of stability versus change in the context of digital business strategy (Keen & Williams, 2013; Tilson et al., 2010).

4.2 Solution Space

In the solution space, our search involved the construction of five prototypical IT artifacts that instantiated the accumulated stock of design knowledge. In addition to iterating back and forth between the heuristic search in the problem and solution spaces, we also iterated back and forth between two levels: (1) IT artifact construction and problem solving in cooperation with companies and (2) abstracted design knowledge accumulation. This corresponds to the distinction between heuristic search and heuristic synthesis in Gregory and Muntermann's (2014) heuristic theorizing framework. Figure 4

provides a high-level overview of the accumulation of design principles over time that resulted from this iterative process.

Table 2 presents the four design principles as outcomes of the heuristic theorizing process. It draws on the following structuring of ideas. First, we introduce a concise statement of the design principle. Second, we present the underlying rationale of the design principle. Third, we explain why and how this principle helps address the two problem requirements of the MCITA problem class. The description of our entire DSR journey in Section 5 provides transparency to the reader regarding how and why this particular set of design principles emerged from our heuristic theorizing work. While we describe our entire journey, we place the greatest emphasis on our third cycle (e.g., more detailed empirical evidence of evaluation outcomes), which is where our final results emerged and stabilized.

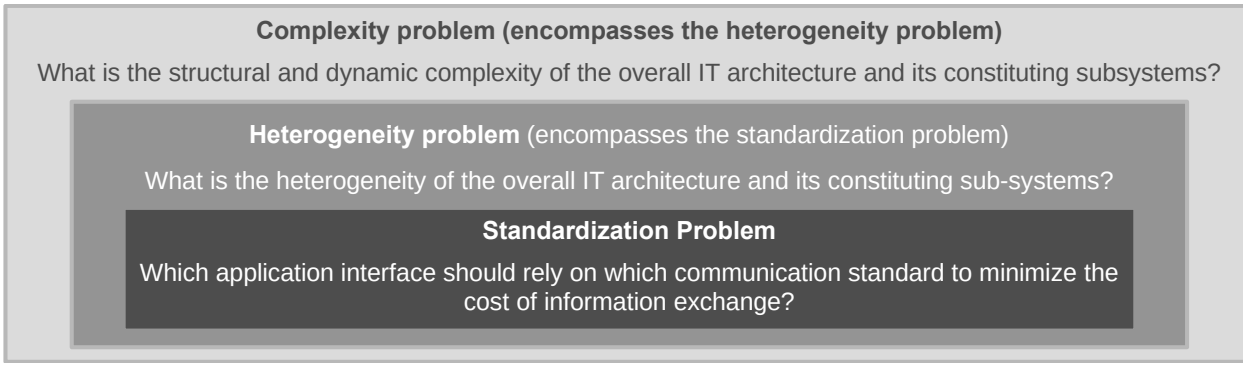

Figure 3. Nested Structure of the Problem Class

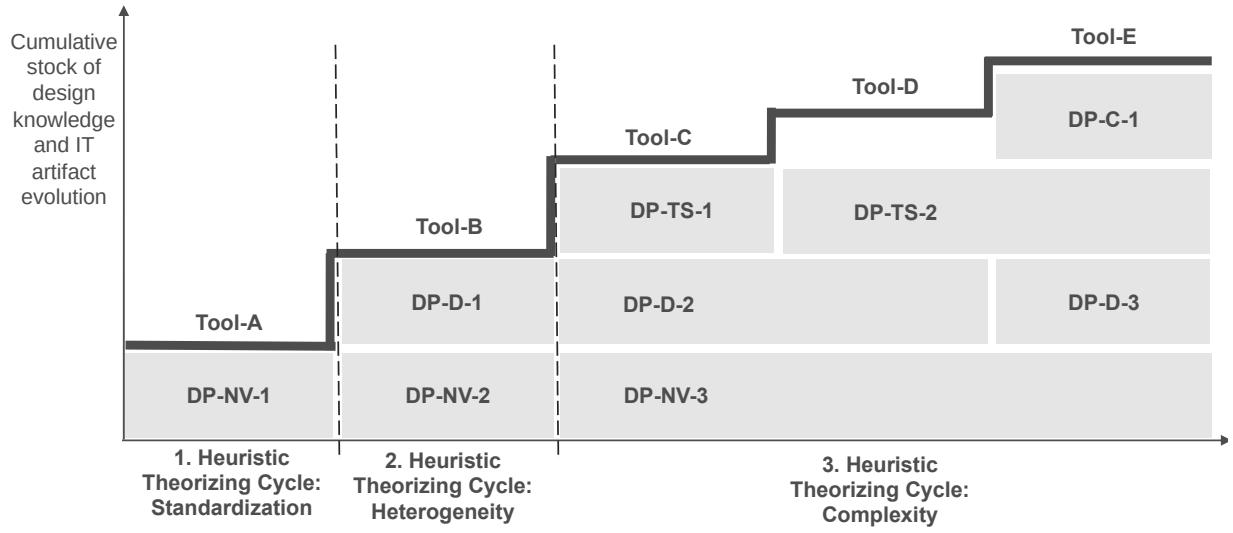

Figure 4. Evolution of the Five IT Artifacts and Accumulation of Design Knowledge

Table 2. Final Set of Design Principles for Constructing a MCITA Tool

Design principle	Rationale derived from prior theory matched with problem-solving experiences	Problem requirements
Number and variety (DP-NV): The artifact should provide information on the number and variety of IT components and relations.	• In a digital business strategy, the IT architecture serves as a platform for distributed, recombinant, and generative innovation (de Reuver et al., 2018; Woodard et al., 2013; Yoo et al., 2012). • Following the law of requisite variety (Ashby, 1956), a certain degree of variety within the IT architecture is needed to allow a sufficiently large variety of actions that can be executed on top of the IT architecture as responses to disturbances.	• Type 1 monitoring: Number and variety are constituent facets of structural complexity. • Type 2 monitoring: Adaptable measures and possibility of comparing values for number and variety across different architectural domains.
Decomposability (DP-D): The artifact should facilitate an understanding of interactions among elements of the IT architecture within and across subsystems at different levels of abstraction.	• Complex systems are organized hierarchically and consist of multiple layers spanning diverse levels of abstraction (Simon, 1962). • Simon describes complex systems such as IT architectures as nearly decomposable systems in which interactions among the subsystems are weak yet not negligible. • The efficient evolution of IT architectures is conditioned by the appropriateness of the degree of decomposability. • Modularity embodies the notion of creating highly integrated, mutually responsive, and tightly coupled subsystems that preserve a stable form and function while simultaneously enabling flexibility in their use and recombination based on the loose couplings and standardized interfaces among different modular subsystems.	• Type 2 monitoring: To solve a given problem the user is able to focus on a specific level of the IT architecture's hierarchy. The problem may appear unsolvable at one level of abstraction, but solvable after zooming in (divide and conquer) or zooming out (obtain a holistic view). DP-D facilitates comprehension by allowing assessment of the appropriateness of the degree of decomposability.
Trace and simulate (DP-TS): The artifact should allow the user to trace and simulate structural complexity over time.	• Monitoring structural complexity over time (retrospectively and prospectively) allows for identifying changes made to an IT architecture. • The theory of dynamically adjusting routines (Berente et al., 2016) suggests that organizations must ensure the proper balance between stability and change (Tilson et al., 2010) in the materiality of an organization (i.e., IT architecture). • The prospective element of DP-TS helps to anticipate possible future changes and may be useful in dealing with the high levels of environmental uncertainty associated with competition involving digital business strategy (El Sawy et al., 2010).	• Type 1 monitoring: DP-TS allows tracking of dynamic complexity (snapshots of structural complexity in time). • Type 2 monitoring: Visualization capabilities (i.e., plotting measures of structural complexity over time) facilitate comprehension.
Configurability (DP-C): The artifact should allow the user to select different perspectives on structural complexity, trajectories of structural complexity (dynamic complexity), and hierarchies of partitions to facilitate an understanding of the specific problem instance at hand.	• The specific perspective of the user of the MCITA tool involves three key elements, each of which relates to one of the three design principles DP-NV, DP-TS, and DP-D. • As Ashby's example of a sheep's brain viewed differently depending on perspective illustrates, it is critical to assume the appropriate perspective for solving any complexity problem. • As digital business moves are carried out by a heterogeneous set of actors, the configurability of the MCITA tool is of strategic importance. Actors' diverse views, such as on technological limitations, architectural standards, and business opportunities offered by an IT architecture, must be coordinated and aligned in the context of digital business strategy (Woodard et al., 2013).	• Type 2 monitoring: The user is able to: (a) ascribe meaning to abstract elements of the IT architecture on its different layers (Richardson et al., 1990; Ross et al., 2006); (b) specify the type of dynamic complexity with regards to the time window, the time intervals of interest, and the different time series; and (c) define, drill down, and roll up, and compare a selected hierarchy of partitions according to specific criteria such as ownership, product categories, markets, and so on.

5 Design Science Research Journey

In this section, we trace the evolution of the IT artifact construction and evaluation over eight years (2008-2016) and across five companies and contexts with the goal of illuminating our journey of accumulating generalizable design knowledge.[2] We describe three cycles of heuristic theorizing and the concurrent evaluation results that triggered transitions from one cycle to the other. In describing each cycle, we focus on the heuristic search within the problem space and solution space, as well as the key moments of heuristic synthesis that allowed us to discover connections between nascent chunks of design knowledge across the two spaces. In two critical moments of our overall search process, the evaluation results triggered decisions to revise and reformulate the problem at hand in fundamental ways, and, in turn, directed further searching in the solution space, resulting in three heuristic theorizing cycles (see Figure 5).

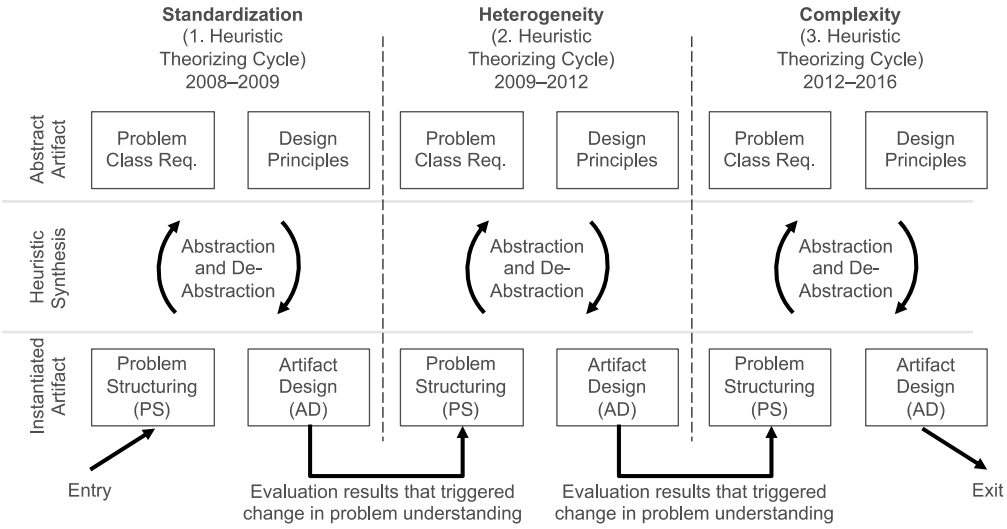

Figure 5. Overview of the Three Heuristic Theorizing Cycles

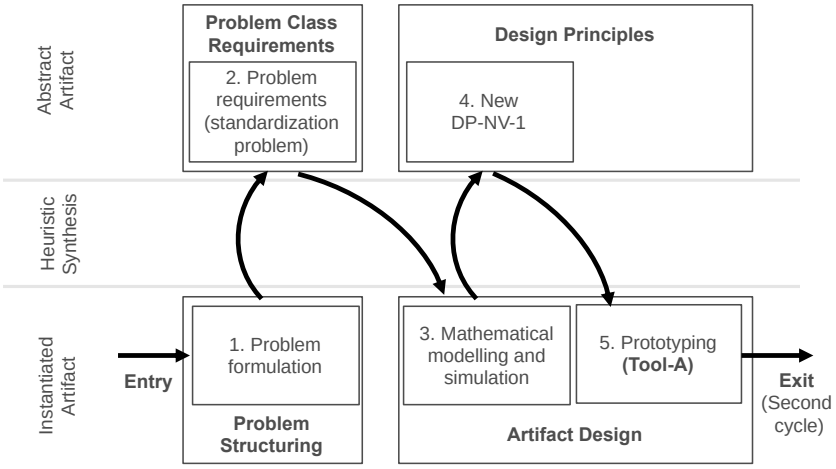

Figure 6. Overview of the First Heuristic Theorizing Cycle

[2] Intermediate results and further details of this process are documented in (Schütz, 2017; Schütz, Widjaja, & Gregory, 2013a; Schütz, Widjaja, & Kaiser, 2013b; Widjaja & Buxmann, 2009; Widjaja & Gregory, 2012; Widjaja et al., 2012).

5.1 First Heuristic Theorizing Cycle (2008-2009): Optimizing IT Standardization Decisions

The focus of the first heuristic theorizing cycle (from 2008 to 2009) and the entry into our DSR journey was the problem of making optimal IT standardization decisions of interfaces between applications encountered in Company 1 and the subsequent problem structuring (see Step 1 in Figure 6). During the research process, we learned that this problem of making optimal IT standardization decisions is encompassed in the heterogeneity problem which is in turn is encompassed by the complexity problem (see Figure 3). Therefore, the design principles in this first heuristic theorizing cycle address a subset of problem requirements of the final MCITA problem class.

As part of the subsequent heuristic synthesis and abstract problem requirements definition (Step 2), we drew on the literature about IT standardization (e.g., Weitzel, Beimborn, & König, 2006) and conceived the problem at Company 1 as an instance of the IT standardization problem (see Figure 3): Which application interfaces should rely on which communication standard to minimize the cost of information exchange? This resulted in the idea for a tool that would provide the IT architect support for optimizing the variety of communication standards between applications. To address this key requirement of the formulated IT standardization problem, our heuristic search in the solution space included mathematical modeling and simulation (artifact design in Step 3). We developed a linear optimization model that incorporates problem context-specific parameters to assist IT architects in analyzing the cost-benefit tradeoffs involved in making optimal IT standardization decisions. We used this linear optimization model as the foundation for extensive simulation studies (i.e., purposeful manipulation of the problem parameters), yielding an understanding of the optimal solution structure.

Heuristic synthesis (Step 4) yielded the conclusion that IT standardization drives IT efficiency by eliminating extra costs for information exchange between applications that use different communication standards. This resulted in the first version of the "design principle number and variety" (abbreviated as DP-NV-1; see Table 3), which we instantiated through collaboration with Company 1 through prototyping (artifact design in Step 5) a first tool (Tool A). Tool A allows an IT architect to specify the parameters of an IT standardization problem for application landscapes and interpret the optimization results (see Figure 7).

5.2 Concurrent Evaluation and Transition from Cycle 1 to Cycle 2

Company 1's CIO confirmed the usefulness of our tool but chose not to implement the "optimal" suggested configuration. Instead, the CIO opted for the solution proposed as second-best by the linear optimization model, justifying his choice by highlighting that its "second-best" configuration yielded a much lower overall heterogeneity of vendors and that, according to his experience, large IT application landscapes with a low degree of vendor heterogeneity are much less complex and more efficient to manage.

Based on these experiences, three key ideas triggered our identification of a new problem class. First, the standardization of interfaces between IT applications in the interest of reducing information exchange costs (classical IT standardization view) typically also affects the degree of heterogeneity of the overall IT application landscape because applications are usually tightly coupled to a particular communication standard (e.g., an application comes with different standard data formats for information exchange). Thus, the implementation of IT standardization decisions with a focus on relationships between applications often results in a less heterogeneous landscape of the applications themselves.

Second, in reflecting upon the CIO's comments about the importance of considering the IT vendor constellation in making IT standardization decisions, we realized that minimizing the costs of information exchange between applications is only one facet of the problem. We conceived various other advantages and disadvantages related to making IT standardization decisions and identified the need to explore this further.

Table 3. Design Knowledge after Heuristic Theorizing Cycle 1

	Set of design principles: **To support IT standardization decision making, the artifact should ...**	**IT artifact instantiation (Tool A)**
DP-NV-1	... provide information on the variety of relationships between software components (i.e., application layer).	• Graphical representation of the topology of relations between software components. • Features that allow structured input of parameters of the standardization problem (information cost, standardization cost). • Different types of parameter validations (e.g., no negative costs). • Solver for the linear optimization problem and sensitivity analysis.

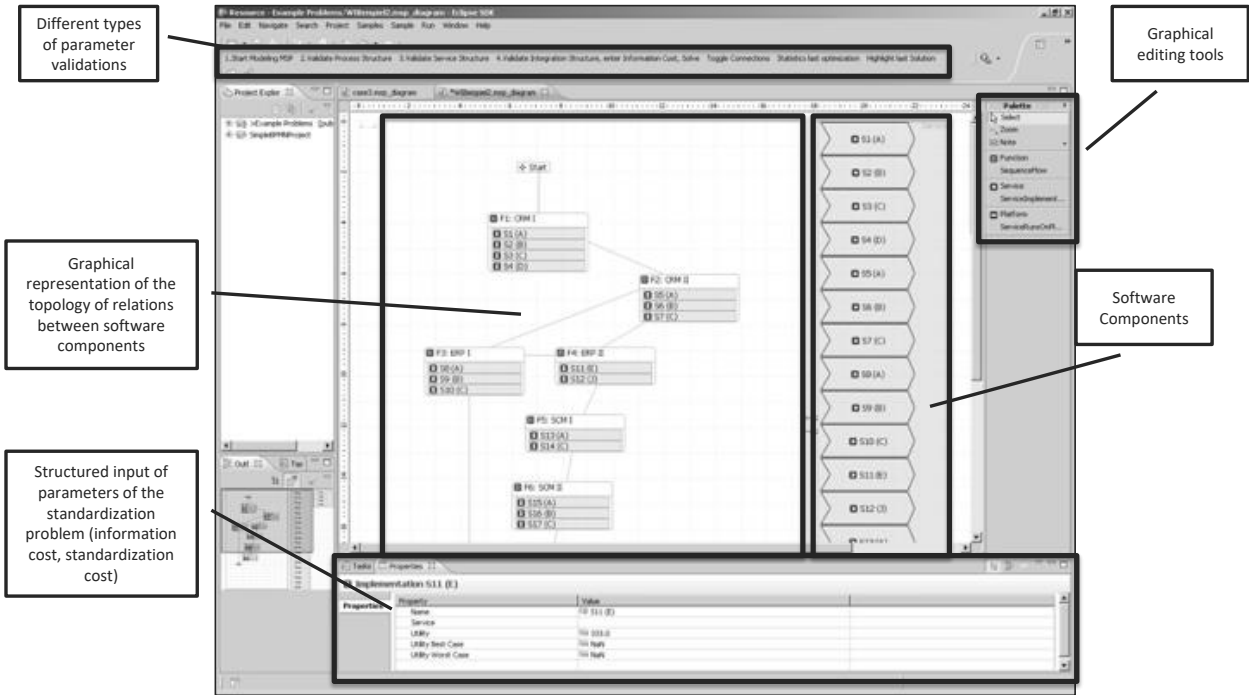

Figure 7. Screenshot of Tool A (Specification of the Network Topology of Software Components and Input of Key Parameters for Standardization Decisions)

Third, we also reflected upon the multilayered architecture of a firm's IT, including application, data, and infrastructure layers, and realized that optimization decisions for an IT application landscape are not isolated, standalone decisions. In fact, a decision affecting the application layer in most cases also affects other interdependent layers of the IT architecture. We also observed that, in practice, multiple IT vendors typically provide IT system components and services across multiple interdependent layers. As a result, we shifted our focus from application landscapes to multilayered enterprise IT architectures.

5.3 Second Heuristic Theorizing Cycle (2009-2012): Monitoring IT Heterogeneity

Triggered by the emergent outcomes of the concurrent evaluation explained above, we again engaged in problem reformulation and class identification (see Figure 8: Step 6), marking the entry point to the second heuristic theorizing cycle. In an attempt to generalize beyond the context of Company 1 and identify the abstract problem requirements, we began working with Company 2 (problem reformulation in Figure 8: Step 6; see Table 1 for the purpose and scope of these interactions). These interactions yielded the key insight to structure the problem space: the idea of viewing the IT standardization problem defined in the previous cycle as a nested subproblem of determining the desirable degree of IT heterogeneity of a multilayered IT architecture, which requires tracking and comprehending the variety of applications, data, and infrastructure components and their interrelations (see Figure 3). Our reasoning was as follows. First, the information exchange costs between applications, the primary focus of making IT standardization decisions for interfaces on the application layer, is only one of many different cost categories (e.g., maintenance, licensing, and employee training) that need to be monitored by the IT architect across different layers. This broader set of aspects can be monitored by expanding the focus from the standardization of application interfaces to IT heterogeneity. Second, this expanded problem understanding permitted a broader managerial focus that goes beyond numbers and IT efficiency, the typical focus of IT standardization, and also considers strategic effects such as IT flexibility. Synthesizing these insights into new general problem requirements (Step 7), our newly defined focus was to find a desirable degree of IT heterogeneity of a multilayered IT architecture (in which the IT application landscape, which was the focus of our first heuristic theorizing cycle, represents only one layer in addition to the other data and infrastructure layers).

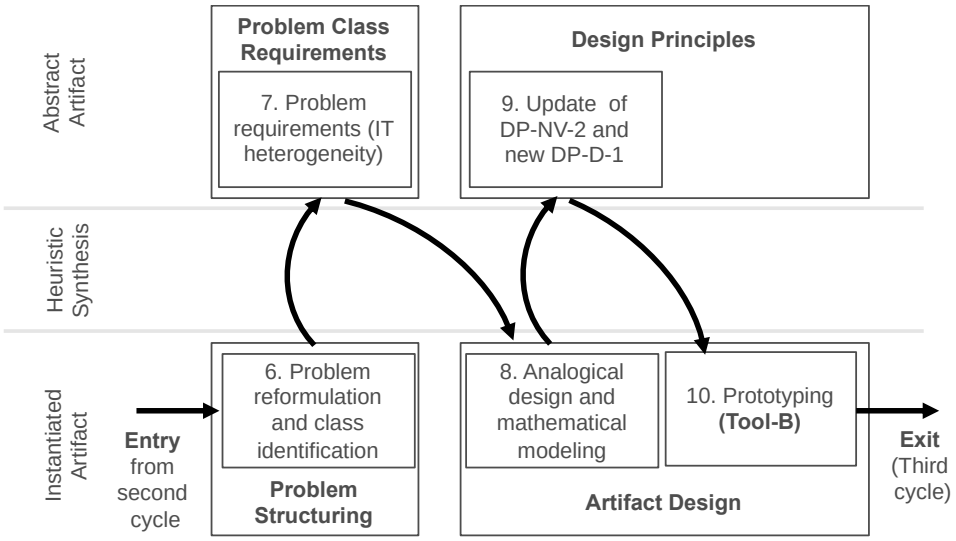

Figure 8. Overview of the Second Heuristic Theorizing Cycle

Based on our perception of again having a sufficiently well-defined problem at hand, we transitioned to artifact design. Drawing on analogical design and mathematical modeling (Step 8) yielded the idea to apply the theory of biological ecosystems (Peet, 1974). Accordingly, systems variety can be conceptualized to include two facets: evenness, or the parity of the prevalence of a species, and richness, or the number of species. We transferred the concept of entropy measure as used in the fields of biology and economics (Jacquemin & Berry, 1979) to measure IT heterogeneity (see Appendix B for details of this approach to measure IT heterogeneity).

Synthesizing (Step 9) the cumulative learning outcomes described above and building upon the insights from our first heuristic theorizing cycle, we arrived at the conclusion that reducing the IT heterogeneity of an IT architecture is associated with potential cost and knowledge synergies (e.g., implementation and training costs are significantly reduced for homogeneous software systems) that must be considered within and across different layers (i.e., application, data, and infrastructure). Based on the above, we refined design principle DP-NV-2 (see Table 4). In working with Company 2, we also realized the need for the IT architect to be able to drill down and up, or zoom in and out, to assess the degree of IT heterogeneity across different departments and subdepartments or other types of partitions of the overall system. Thus, the IT architect should be able to identify sources of IT heterogeneity and take appropriate corrective action in the identified part of the organization. This resulted in the first version of "design principle decomposability" (DP-D-1) (see Table 4). Through prototyping (artifact design in Step 10) and drawing on standard spreadsheet technology, we instantiated the accumulated stock of design knowledge and constructed a new tool (Tool B).[3] The essence of the artifact is the underlying mathematical model represented as a set of formulas (see Appendix B for more details).

5.4 Concurrent Evaluation and Transition from Cycle 2 to Cycle 3

We engaged Company 2 for the concurrent evaluation of Tool B and the embedded stock of accumulated design knowledge. We learned that it is relevant for an IT architect to know whether the IT heterogeneity in a given department and subpart of the organization stems from various subparts of that department that are themselves plagued by IT heterogeneity—suggesting the need to zoom further in—or whether that IT heterogeneity stems from various subparts of that department that are themselves characterized by IT homogeneity. With the goal of replication and increasing the generalizability of our nascent design knowledge, we expanded the concurrent evaluation activities to Company 3. With the help of data we obtained from Company 3's IT department, we were able to show that our Tool B enables IT architects to identify those parts of the IT architecture that are a significant source of IT heterogeneity (e.g., technologies and programming languages in use), a constituent element of structural complexity.

[3] As the graphical user interface of Tool B is based on standard spreadsheet technology, it is therefore not useful for demonstrating the instantiated design knowledge and we omit presenting a screenshot.

Table 4. Design Knowledge after Heuristic Theorizing Cycle 2

	Set of design principles **To support IT standardization decision making, the artifact should ...**	**IT artifact instantiation (Tool B)**
DP-NV-2	... provide information on the heterogeneity of relations and components of a multilayered IT architecture.	• The formulas of the mathematical model as part of the spreadsheet prototype allow for calculating the entropy measure and the Herfindahl Hirschman Index to quantify the IT heterogeneity (for the basic idea of the entropy measure, see Appendix B).
DP-D-1	... facilitate zooming in and zooming out to different levels of abstraction of a multilayered IT architecture.	• Tool B allows different subsystems of the IT architecture to be defined. The decision maker is able to specify the architectural domains in which he or she is interested.

In summary, the IT architects from Company 2 and Company 3 both confirmed the usefulness of our conceptualization and measurement of IT heterogeneity (see Appendix B). However, similar to our previous transition from the first to second heuristic theorizing cycle, we obtained feedback that a broader view of the problem was required. This prompted, once again, a revision of the problem class (Figure 9: Step 11) and thus a transition into the third and final cycle of heuristic theorizing. In doing so, we conceived the idea of drawing on a complex system perspective, viewing IT standardization and IT heterogeneity decisions as nested subproblems of monitoring the structural and dynamic complexity of an IT architecture (see Figure 3). This idea occurred to us mainly because we learned that IT architects focus on different types of IT heterogeneity: both the heterogeneity of IT *components* (e.g., applications) and the heterogeneity of the relationships between those IT components (e.g., application interfaces). Comparing this insight with the literature, we repeatedly came across the notion of complexity (Schneberger & McLean, 2003). Complexity can be understood as a system state that results from the number of its constituent components (Klir, 2001; Flood & Carson, 1993) and relationships (Flood & Carson, 1993). Furthermore, a complex system can be characterized by the heterogeneity (i.e., variety) of components and relationships that form part of the overall system (Simon, 1962), highlighting the interconnections between systems heterogeneity and systems complexity.

5.5 Third Heuristic Theorizing Cycle (2012-2016): Monitoring IT Complexity

The concurrent evaluation activities with Company 2 and Company 3 explained above and the identification of monitoring IT architectural complexity as the relevant, general problem class triggered problem reformulation (Step 11), which was the entry into the third and final heuristic theorizing cycle (see Figure 9). From the beginning of mid-2012, we focused on the problem of monitoring the complexity of IT architectures (see Figure 3 for the problem description).

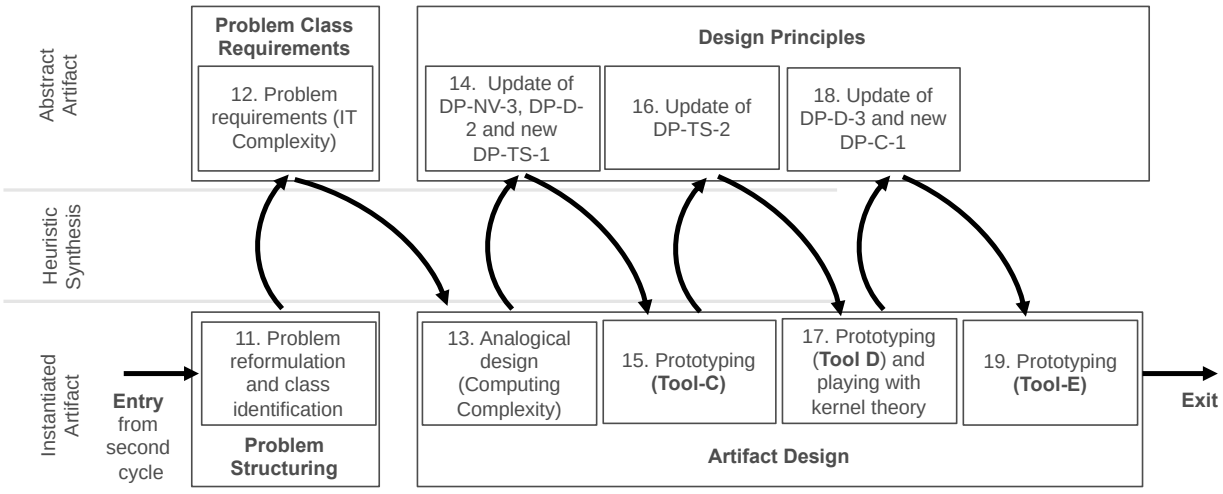

Figure 9. Overview of the Third Heuristic Theorizing Cycle

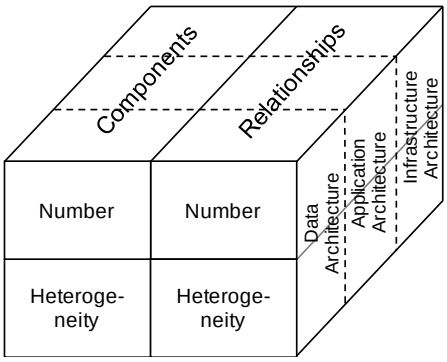

Figure 10. Conceptualization of the Structural Complexity of an IT Architecture

In this cycle, we began working with Company 4 and Company 5, two large banks with legacy, historically grown, and complex IT architectures that formed the foundation for the execution of their digital business strategies. This was a defining moment for our work, as we identified digital business strategy and the associated challenge of transforming a legacy IT architecture into a digital platform as a highly relevant context for monitoring architectural IT complexity.

In Step 12, we synthesized the shift of our problem understanding from heterogeneity to complexity and identified the new problem requirements associated with the following question: What is the structural and dynamic complexity of the overall IT architecture and its constituting subsystems? This resulted in the nested problem structure described in Figure 3—that is, that the standardization problem class is part of the heterogeneity problem class, which in turn is part of the complexity problem class. In an effort to define the boundaries of our nascent set of design principles, we decided to maintain our focus on IT architecture (a subset of an enterprise architecture).

Through analogical design (Step 13), we identified connections between architectural IT complexity and the concept of computing complexity (Schneberger & McLean, 2003). Drawing on Schneberger and McLean (2003) was the inception point in our DSR journey to draw on a system-theoretic perspective on man-made IT systems (ISO/IEC/IEEE 2011; Simon, 1962). A system (i.e., IT architecture) is defined as a set of components $c \in C$ (e.g., applications, data, and infrastructure) and the relationships between them $R \in (C \times C)$ (e.g., interfaces between applications) (Hall & Fagen, 1969). The computing complexity is influenced by the number and heterogeneity of IT components and relationships (Schneberger & McLean, 2003). Combining these ideas with our evolved understanding of multilayered IT architectures, our conceptualization of the structural complexity of IT architecture emerged: structural IT architectural complexity stems from the number and heterogeneity of components and relationships on the three layers of an IT architecture (see Figure 10).

In a subsequent effort of heuristic synthesis (Step 14), we adapted our nascent design knowledge (i.e., DP-NV and DP-D) to the revised general problem requirements. Feedback provided through workshops with IT architects from Company 4 and a participating consultancy firm yielded the refinement of our first design principle as follows: DP-NV-3 concerned the additional monitoring of the "number" of components and relations (instead of the focus on variety in the first version of this design principle). Accordingly, an artifact for monitoring the IT architecture complexity should provide information on the number and variety of IT components of a multilayered architecture (see Table 5). In addition to this structural perspective on IT architectural complexity, the first version of "design principle trace and simulate" (DP-TS-1) emerged. We realized the need to track the history and evolution of the structural complexity measures over time. The first version of this design principle, therefore, focused on incorporating and visualizing historical data on the complexity measures.

Furthermore, DP-D-2 was extended to allow flexible zooming in and out to different subsystems of the IT architecture (in comparison to the previous "static" zooming in on the spreadsheet prototype). Accordingly, based on the basic definition of an IT system (see Section 2.1), partitioning of the system is possible, as follows: One partition is defined as the set of nonempty subsets of C (i.e., the set of components) such that every element is in exactly one of these subsets. Each subset may be considered a system itself and therefore can also be partitioned, resulting, overall, in a nested hierarchy of partitions. Hierarchic thinking plays a crucial role in understanding any complex system (Simon, 1962). To illustrate, if the partitioning of applications is done according to the dimension "organizational structure," a subset corresponds to the set of all applications owned by one department. This results in a hierarchy of partitions for the dimension "organizational structure." That is, the organization can be partitioned into departments, which can be partitioned into subdepartments.

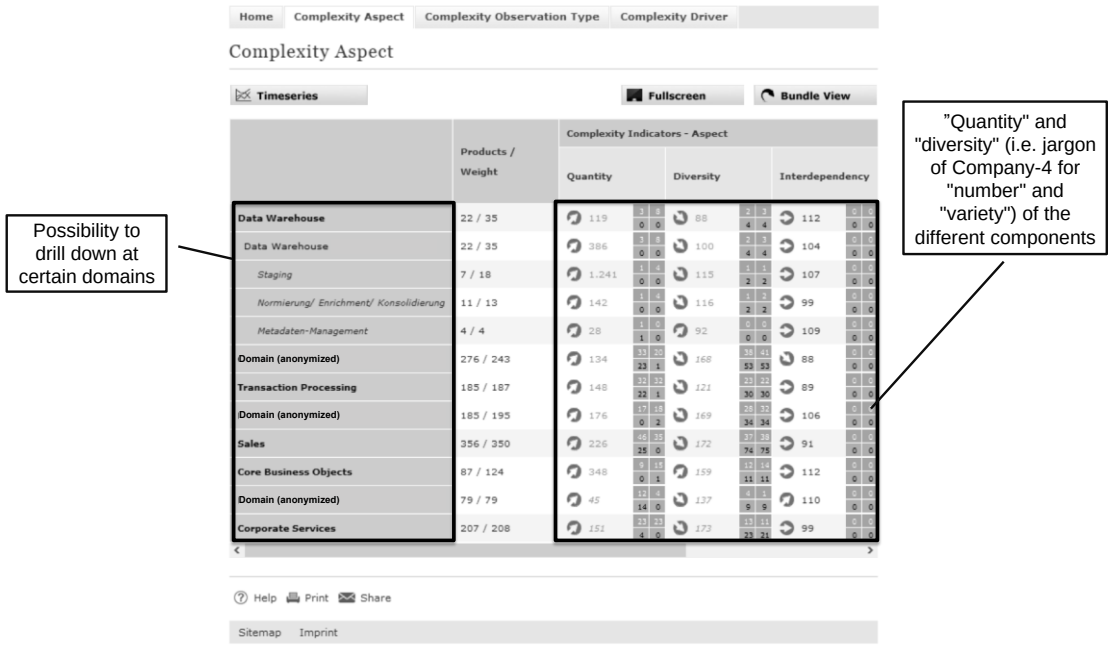

Figure 11. Screenshot of Tool C (Dashboard with Information on Number and Variety)

Typically, different dimensions to partition an IT architecture exist. Examples include geographic (world, country, and city), functional (overall system and business functionality), and the organizational dimension (overall organization, department, and subdepartment) discussed earlier.

We engaged in prototyping activities (artifact design in Step 15) to instantiate our current state of nascent design knowledge and evaluated the resulting Tool C with Company 4. Figure 11 shows a screenshot of the dashboard in Tool C, where the user is able to drill down to certain domains and the prototype presents the "quantity" and "diversity" (Company 4's terms for "number" and "variety") of the different components and relations. Furthermore, Tool C allows for synchronizing data concerning the current state of the IT architecture with different repositories and includes a "data warehouse" for information related to IT complexity. Based on these data, it is possible to obtain an overview of the historical data as a "time series" (see Figure 12).

Concurrent evaluation feedback obtained through interactions with Company 4 illustrated the usefulness and utility of Tool C and the underlying conceptualization of structural IT architectural complexity (see Figure 10). The instantiated prototype helped structure and improve the understanding of a large range of IT architecture complexity measures that Company 4 had developed. We were also able to identify areas in which Company 4's IT architecture complexity had not been sufficiently monitored in the past.

As the principal IT architect at Company 4 explained,

> *Of course, we first had to think about what IT complexity really is, what kind of complexity we want to look at and where to find the objects in the company to pinpoint the complexity. ... And here we focus on the structural complexity of a system or cluster, by which I mean multiple applications in a functional domain, which results from the number and diversity of components and the number and diversity of relationships between them. ... The next step was to think: what do we really want to look at now? Yeah, well, we're architects, so let's look at the "layers of the architecture," that is, data, applications, and infrastructure.*

In addition to providing feedback that our conceptualization of the IT architectural complexity aligned well with the view in practice, Company 4 also began to use our tool for new IT investment decisions and IT project prioritization activities that were part of the company's IT transformation planning. Specifically, our prototype, fed with company-specific data, helped IT decision makers conduct a so-called "architecture check" during budget negotiations and systematically assess the effect of a proposed IT investment and change on the state of architectural IT complexity.

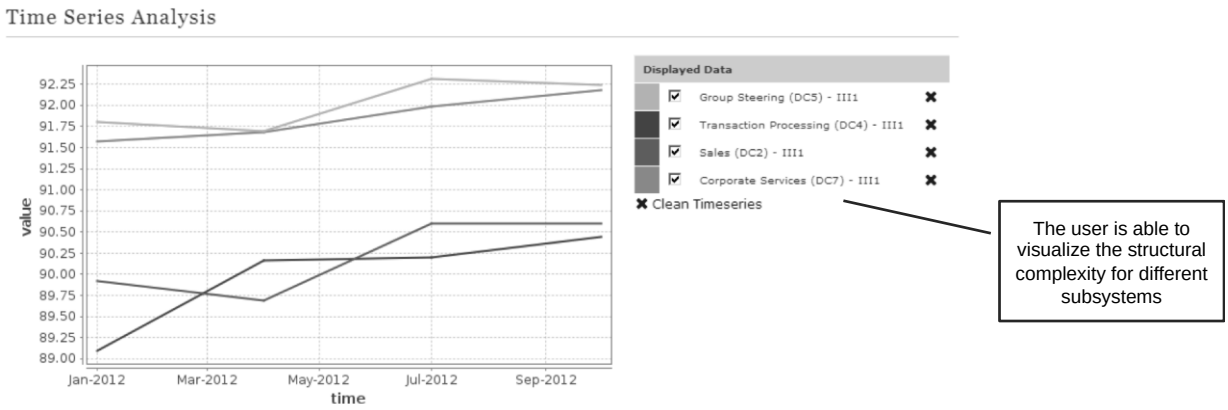

Figure 12. Screenshot of Tool C (Evolution of Complexity Indicators over Time)

The principal architect at Company 4 stated:

> *A colleague has been fighting for quite some time now to get his "zoo" a bit smaller—that is, to standardize—but he has not made it over the years because of the budget situation. Now he was able to see from Tool C that standard compliance has decreased even further over the last years. ... So, he said, "OK, if I have the information now that it [the standard non-compliance] is even higher, then maybe I have a better argument for getting that budget."*

The department head of architecture management of Company 4 articulated how DP-D-2 helps to derive action from the information provided by Tool C, stating:

> *If I only have key metrics at the level of the entire bank, then it does not help me. I have to be able to break it down. Otherwise, I cannot say where to start consolidating.*

The principal architect at Company 4 added more details and stated:

> *At the level of the entire application landscape, the aggregated key metric is nothing more than an indicator. You may take decisions based on: "Here I have to do something." But what this "something" is, you will only find out if you really hone in. Because if you realize that standard compliance in a certain domain is not good, then you need to examine the root causes. The domain may contain 50 application systems, but which of these 50 systems is ultimately nonstandard, you cannot tell from the aggregate number. That's why you need the drill-down. This way, you are able to track down the responsible manager and then they have to investigate this further.*

During workshops with Company 4, the principal architect emphasized the usefulness of the time series (DP-TS-1) for interpreting the measures of IT complexity, stating:

> *I think that the rate of change in architecture is meaningful only when reflected in complexity indicators. ... In my opinion, the rate of change of the architecture alone is not relevant information. More useful is the change of the indicators over time, a time series of indicators.*

The feedback obtained about Tool C triggered a new phase of heuristic synthesis (Step 16). Specifically, feedback from Company 4 suggested the importance of tracking the history as well as simulating the further evolution of IT architecture complexity at hand—leading to a refinement of the design principle "track and simulate" to DP-TS-2. Our rationale for the adjustment to DP-TS is twofold. First, tracing historical changes of key facets of IT complexity over time may be useful (this was already part of the previous version of DP-TS). Second, examining the potential effects of prospective future changes through simulation may be useful because simulation enables the IT decision maker to test different options for action and investigate the effects of particular techniques (e.g., IT standardization) on the overall IT architecture. Thus, overall, the tool should allow the user to observe the effectiveness of past and planned applications of techniques (e.g., IT standardization) to deal with IT complexity.

The satisfaction of Company 4 with the artifact and our growing confidence regarding the usefulness of our nascent design theory in the given context prompted us to reflect upon the potential generalizability of our design principles. Up to this stage of our research program, we had developed DP-NV-3, DP-D-2, and DP-TS-2 (see Figure 4). To explore the potential generalizability of this set of design principles and

carry out further replication in a new context, we undertook further prototyping activities to develop Tool D in collaboration with a new company (Step 17). In 2013, we engaged Company 5, which was at the time in the middle of the largest IT transformation program in its corporate history as part of its broader digital strategy and transformation agenda. The feedback confirmed the revised version of DP-TS, and the added simulation abilities of Tool D (see Figure 13) were highly appreciated by the practitioners. An IT architect at Company 5 stated:

> *Simulation offers me the opportunity to illustrate my planned projects and express the underlying idea. I can express my idea, I can document it, and I can present it: "We have planned the following projects and tried out the following alternatives in which the complexity changes as follows." For me, that is the essential added value of simulation.*

From our engagement with Company 5, we learned that monitoring the architectural IT complexity in the context of digital business strategy and transformation involved developing the digital platform capability to enable the flexible recombination of sets of integrated IT components (i.e., modules) according to differentiated business needs. Therefore, during the implementation of Tool D, we also worked with the theory of modular systems (Ethiraj & Levinthal, 2004; Simon, 1996) as kernel theory (Step 17 in Figure 9). This helped us conceive the idea for an extended conceptualization of IT complexity. Specifically, modularity enables an efficient, flexible recombination of sets of integrated IT components (so-called modules).

Particularly in the context of digital platform design, the notion of modularity is relevant for understanding how to deal with IT complexity. Prior studies—for example, that of Ethiraj and Levinthal (2004)—state that the efficient and flexible recombination of modules requires an "appropriate" degree of modularity. According to this, stretching modularization too far may result in the IT architecture leaning strongly toward flexibility but at the expense of efficiency; that is, the efforts of integrating and testing evolved subsystems are significantly increased. Conversely, giving modularity too little emphasis may result in an insufficient range of possibilities for flexible recombination, i.e., the efforts of separating, changing, or recombining subsystems become excessively high.

Synthesizing the insights (Step 18) from joint problem solving with Company 4 and Company 5 and prior kernel theory on modular systems yielded a further refinement of the design principle "decomposability" (DP-D3). Specifically, the lesson from Company 5 about "flexible recombination" pointed us toward the concept of modularity. As a result, we extended our prior conceptualization of IT complexity (see Figure 10) by including this concept. Accordingly, the artifact for monitoring IT complexity should also provide relevant information for assessing the appropriateness of modularity. In testing this idea in our engagement with Company 5, we reasoned that an appropriate level of modularity contributes to balancing IT efficiency with IT flexibility.

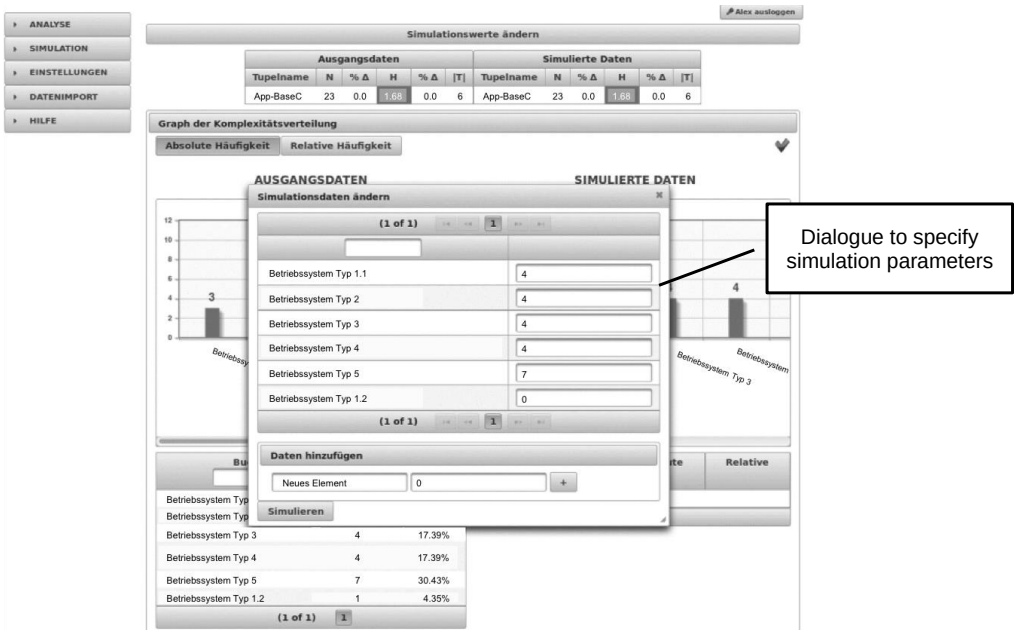

Figure 13. Screenshot of Tool D (Simulation Dialogue)

In addition to the validation and refinement of existing design principles, the concurrent evaluation of Tool D and especially the satisfaction with the revised conceptualization of IT architectural complexity prompted us to develop (also in Step 18) the "design principle configurability" (DP-C-1). The tool should allow the user to choose the perspective on IT architectural complexity needed to solve a specific decision problem: IT decision makers must be able to specify a "complexity configuration" and select (1) the type of IT components (e.g., application components, hardware components, etc.); (2) the type of relationship between IT components (e.g., interfaces, physical connections, etc.); (3) characteristics that differentiate IT components from one another (e.g., vendor heterogeneity, programming language used, etc.); and (4) characteristics that differentiate the relationships between IT components from one another (e.g., type of interface, such as synchronous or asynchronous, etc.).

For additional replication of our revised stock of cumulative design knowledge, we engaged once again with Company 4 (artifact design in Step 19). For this final prototyping and concurrent evaluation effort, we included all of our nascent design principles in the instantiated IT artifact (i.e., Tool E; see Figure 14), which was constructed through cooperation with Company 4. For our instantiation of the new version of design principle DP-D-3, we followed Ethiraj and Levinthal (2004, p. 162) and relied on four dimensions of appropriateness of modularity: "the 'appropriate' number of modules, the 'appropriate' mapping of design elements to the modules, the 'appropriate' interactions among the design elements within each module, and the 'appropriate' interfaces or interactions between modules."

Figure 14 shows how the measures proposed by Ethiraj and Levinthal (2004) were implemented in our MCITA tool. The user interface is structured into one primary and three secondary panes. By using the visualization of modules and IT components in the primary pane, the user can assess the appropriateness of mapping IT components to modules. The same visualization in the primary pane provides the information required for an assessment of the appropriateness of the number of modules. The upper two secondary panes on the right side show the number of internal and external relations among IT components, allowing the user to assess the appropriateness of relations within and between modules. The fourth pane in the lower right draws on information from the upper two panes and provides additional information on the proportion of relations within and between modules.

The feedback we obtained from Company 4 representatives was positive and provided support for our nascent set of design principles. Including measures on the "appropriateness" of modularity on all levels of abstraction was conceived as extremely helpful by the companies we worked with. The IT architect of Company 4 stated:

> *An incorrect assignment of components [to modules] can have catastrophic consequences for the flexibility of the overall system. If, for example, dependencies are distributed across all departments, this can make it much more difficult to communicate change. When responsibilities—or even budget—for a single application is divided among 15 different parties, you cannot achieve anything.*

The assessment of modularity seemed critical to avoiding this situation. As the architect explained:

> *You will never get a completely clean, disjoint 1:1 mapping. But cutting it in pieces, in a way that the overhead stays low or dependencies are generally concentrated within a single unit, is the goal. This is loose coupling: being as self-contained as possible, and the interaction outside my module should be as little as possible and as planned as possible.*

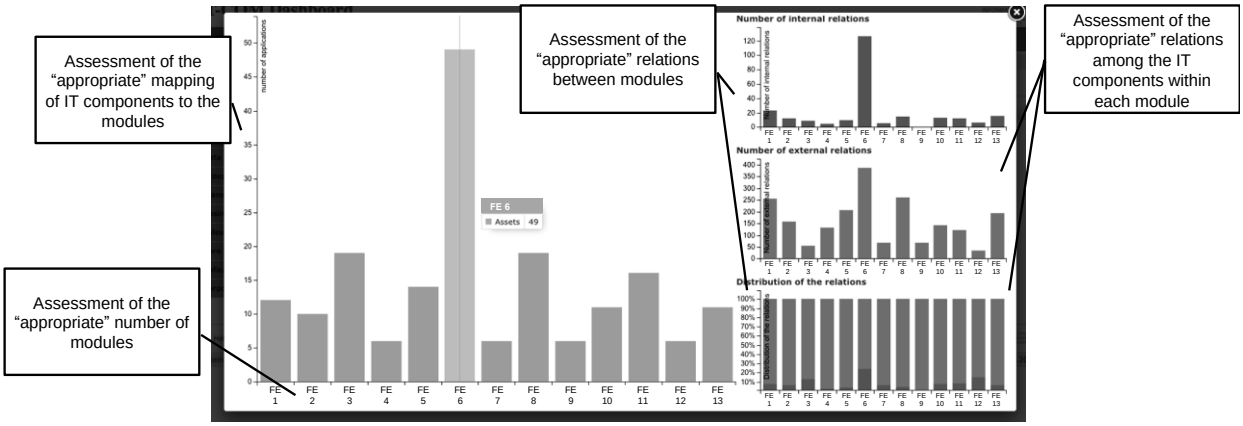

Figure 14. Screenshots of Tool E (Focus on the Appropriateness of Modularity)

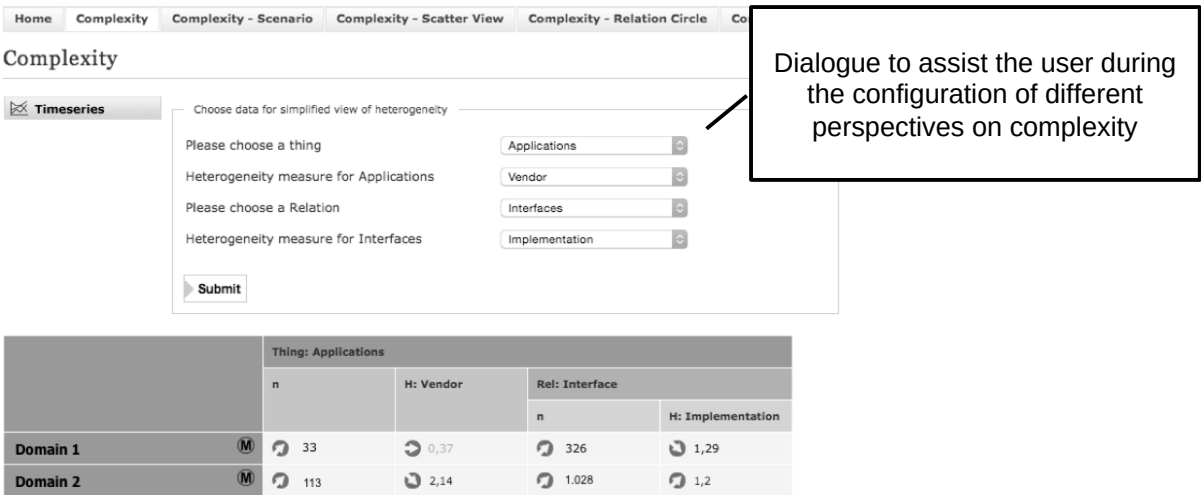

Figure 15. Screenshot Tool E (Selection of Complexity Type)

Table 5. Design Knowledge in the Solution Space after Heuristic Theorizing Cycle 3

	Set of design principles **To support IT standardization decision-making, the artifact should ...**	**IT artifact instantiation (Tool C, D, E)**
DP-NV-3	... provide information on the number and variety of IT components and relations.	• Tools C, D, and E contain dashboards with measures to quantify the number and variety of selected IT components (see Figure 11). • The prototypes use the entropy measure and the Herfindahl Hirschman index to quantify the variety. • The dashboards are enriched by visualizations of data quality (highlighting missing values) to facilitate the interpretation.
DP-D-3	… facilitate an understanding of interactions among elements of the IT architecture within and across subsystems on different levels of abstraction.	• Tools C, D, and E allow the user to drill down into different domains of the IT architecture. • Tool E integrates measures to assess the "appropriateness" of modularity at different levels of abstraction.
DP-TS-2	... allow the user to trace and simulate structural complexity over time.	• Tools C, D, and E contain visualizations of complexity measures over time (see Figure 12). • Tools C, D, and E can import historical data from repositories. • Tools D and E allow simulations of IT projects (e.g., consolidation projects) on the measures of IT architectural complexity.
DP-C-1	… allow the user to select different perspectives on structural complexity, trajectories of structural complexity (dynamic complexity), and hierarchies of partitions to facilitate an understanding of the specific problem instance at hand.	• Tool E enables the user to customize the perspective on complexity depending on the problem at hand (see Figure 15).

The newly implemented version of DP-C increased the generalizability of the application of Tool E. Figure 15 shows a dialogue in the prototype that exemplifies the assistance provided to the tool user during the configuration of different perspectives on structural complexity. This involves the selection of IT components (e.g., applications), a measure of the variety of IT components (e.g., vendors), the relations between the selected IT components (e.g., interfaces), as well as a measure of the variety of relations (e.g., implementation of the interfaces). Similar dialogues exist to configure the trajectories of structural complexity (dynamic complexity) and hierarchies of partitions. As the principal IT architect of Company 4

stated: "That's the generic thing about the tool. You can now use it as a toolbox for almost any question."

A department head of Company 4 expanded on this point:

> *The exciting thing here is that you can pick individual questions from the huge wallpaper of possibilities. I can only look at programming languages or at database systems and, and, and. ... Here, I can investigate an isolated problem without being overwhelmed by all details.*

6 Discussion and Implications

Our main contributions are an increased understanding of a novel problem class as well as a corresponding set of design principles. We developed these contributions by following an iterative approach with successive refinements and integrating actual projections into instantiated IT artifacts across multiple contexts and companies. The design knowledge we accumulated during this process contributes to our understanding of both the problem and solution spaces introduced earlier. In addition, from our rich description of the design process itself, we derive implications for the practice of conducting DSR.

6.1 Implications for Research on IT Architecture Management in the Context of Digital Business Strategy

This paper's contribution to design knowledge is twofold: in the problem space, we discovered and conceptualized the problem class "monitoring the complexity of IT architectures" (MCITA); in the solution space, we built nascent design theory, drawing on the kernel theory of complex systems (e.g., Amaral and Uzzi, 2007; Ashby, 1960; Buckley, 1967; Schneberger & McLean, 2003; Simon, 1962) as well as our experiences and observations from iterative IT artifact construction and concurrent evaluation across multiple contexts and companies. This knowledge contribution in the solution space takes the form of a set of design principles that offer prescriptive guidance for how to design IT support for addressing MCITA problems. We discuss the contributions in the two spaces and their respective implications in more detail in the following.

In the problem space, we discovered a nested problem structure in which the MCITA problem class encompasses the heterogeneity problem, which in turn encompasses the standardization problem (see Figure 3). The MCITA problem class we discovered by examining IT architecture management in the context of digital business strategy is not entirely novel; rather, it is an instance of the general class of problems of monitoring complex systems (Henneman & Rouse, 1986; Murray & Liu, 1997; Murray & Yili, 1994; Natu et al., 2016; Psiuk & Zielinski, 2015; Vierhauser et al., 2016; Zinser & Henneman, 1989). To the best of our knowledge, however, this study is the first systematic attempt to establish a link between this general problem class and the problem domain of IT architecture.

This contribution in the problem space has implications for research on IT architecture, as we offer a change in perspective from the view of "reducing IT architecture complexity" (Guillemette & Paré, 2012; Ross et al., 2006) toward "monitoring the complexity of IT architectures" (i.e., MCITA). The former view is based on the assumption of the separation of the IT function from business units, whereas the latter view is based on the alternative assumption, in line with the fusion view (El Sawy, 2003; El Sawy et al., 2010), that IT architectures are symbolic representations and material manifestations of the firm's systems of digital offerings, processes, and platforms (Keen & Williams, 2013; Woodard et al., 2013). Based on this latter assumption and the experiences gathered during our heuristic search, we offer a definition of an IT architecture as a distributed, evolutionary, and emergent system of IT systems that enables the digital business strategy of one or more partnering firms. Based on our insights derived from heuristic theorizing, we suggest that IT architectures in the context of digital business strategy exhibit characteristics that resemble digital infrastructures (Hanseth & Lyytinen, 2010; Henfridsson & Bygstad, 2013; Tilson et al., 2010) and that monitoring their complexity is integral for business strategy and should be viewed as an aspect of digital business strategy execution. A key implication of these insights is that reducing the structural and dynamic complexity of IT architectures, which is the focus of the literature, is often not feasible and is certainly not a sufficient means to ensure the strategic business value contribution of an IT architecture (Guillemette & Paré, 2012; Ross et al., 2006).

In the solution space, we developed a set of four design principles for the MCITA problem class. This accumulated design knowledge was actually projected (Baskerville & Pries-Heje, 2014) into multiple instantiated IT artifacts (i.e., software prototypes) across different contexts and companies. Each of the four design principles offers cumulative contributions and extensions of the existing literature (see Table 6). In particular, our IT artifact construction work illustrated that features of existing system types (i.e., spreadsheet, metadata repository, and simulation) may be enhanced by novel features (i.e., functions to navigate the hierarchy of systems, complexity measures, and tools to assess the appropriateness of modularity) to address the class of problems we refer to as MCITA.

Table 6. Summary of Contributions of our Design Principles to the Existing Literature

DP	Cumulative contributions and extensions of the existing literature
Number and variety (DP-NV)	• We transferred the concept of structural complexity (Henneman & Rouse, 1986; Ribbers & Schoo, 2002; Schneberger & McLean, 2003; Xia & Lee, 2005) to IT architecture management in the context of digital business strategy (Keen & Williams, 2013). • We transferred and illustrated the utility of the entropy measure and the Herfindahl Hirschman Index from the field of economics (Jacquemin & Berry, 1979) to the MCITA problem. • We extended the conceptualization and measurement of systems variety in the MCITA context by drawing on knowledge from the field of biology that distinguishes between richness and evenness (Peet, 1974).
Decomposability (DP-D)	• Design knowledge about IT architecture management (e.g., Aier et al., 2009; Winter & Aier, 2011) and monitoring complex systems (e.g., Maier, 1998) is extended by drawing on the notion of the "near decomposability" of a complex system (Simon, 1962). • The incorporated idea of hierarchic thinking for understanding a complex system extends the need, previously identified in the literature, for a comprehensive system-wide view on monitoring complex systems (e.g., Natu et al., 2016). • We transferred and illustrated the utility of Ethiraj and Levinthal's idea of "appropriateness of modularity" to facilitate comprehension of the (near) decomposability (Simon, 1962) of a complex system or IT architecture (Ethiraj & Levinthal, 2004). • The idea of "drill-down" from the literature on monitoring complex systems (i.e., Psiuk & Zielinski, 2015) is extended to the broader idea of decomposability.
Trace and simulate (DP-TS)	• The idea of monitoring the evolution, elaboration, and change of systems complexity over time is consistent with the literature on the conceptualization of dynamic complexity (Henneman & Rouse, 1986; Ribbers & Schoo, 2002; Schneberger & McLean, 2003; Xia & Lee, 2005). • We extended our understanding of how a balance between stability and change can be ensured in the evolution of IT architectures that resemble digital infrastructures (Tilson et al., 2010) and the materiality of organizations (Berente et al., 2016) by monitoring structural complexity over time. • The ability to trace and simulate changes in structural complexity over time helps in understanding how uncertainty and risks (e.g., system failure) can be managed, which has been discussed in the literature on monitoring complex systems (e.g., Domerçant & Mavris, 2011).
Configurability (DP-C)	• The view embodied in the design principle to select a specific perspective on a given instance of MCITA problems is in line with Ashby's observation that a complex system (e.g., a sheep's brain) is viewed differently depending on the perspective taken (e.g., that of a butcher versus a neurophysiologist) (Ashby, 1960). • The configurable conceptualization of IT architectural complexity (Figure 10) provides a universal language that allows a diverse set of heterogeneous stakeholders across an organization to participate in solving MCITA problems by contributing each of their unique perspectives depending on the given MCITA instance. • The idea of configurability extends our understanding of how to deal with the diversity of different system types, functions, and organizational units overlooking them, which has been identified as a key challenge in the literature on monitoring complex systems (e.g., Domerçant & Mavris, 2011).

This design knowledge contribution in the solution space offers implications for research on the complexity of IT architectures (e.g., Beese et al., 2016; Schneberger & McLean, 2003). In this research stream, the predominant focus has been on structural complexity and, to a lesser extent, on dynamic complexity (Ross et al., 2006; Xia & Lee, 2005). Our insights derived from heuristic theorizing complement this research and suggest the need to pay closer attention to problem-solving complexity and building IT support that serves as a cognitive aid (Lerch & Harter, 2001). In contrast to the concepts of structural and dynamic complexity, problem-solving complexity concerns human reasoning, attentional resources, skills, and the overall ability to cope with structural and dynamic complexity (Henneman & Rouse, 1986). Under the condition of embracing complexity, for example, the ability to cope with structural and dynamic complexity (i.e., problem-solving complexity) shifts to the foreground of attention by leveraging higher requisite variety in an IT architecture to address the increasing heterogeneity of requirements and demands through the identification of new value-generation opportunities (Ashby, 1956; Priem, Butler, & Li, 2013).

6.2 Implications for Generating Design Knowledge

The DSR paradigm rooted in Herbert A. Simon's seminal work on the science of design (Simon, 1996) has developed significantly (Hevner et al., 2004). Despite these advancements, further guidance is needed for DSR teams on the rigorous and systematic generation of projectable design knowledge. The development of specific methods and frameworks that

focus on the intertwined generation of design knowledge and IT artifact construction/evaluation—action design research (Sein et al., 2011) and heuristic theorizing (Gregory & Muntermann, 2014) in particular—offer a useful point of departure for developing a genre of DSR that emphasizes the accumulation of generalizable and tangible design knowledge grounded in real-world problem solving through iterative IT artifact construction and concurrent evaluation (Peffers, Tuunanen, & Niehaves, 2018). In this paper, we followed this emergent genre of DSR and offer as an empirical contribution "a rich description of the design process" (Ågerfalk, 2014, p. 595). From this description, we derive the following implications for generating design knowledge that may contribute to the ongoing methodological discourse on making DSR contributions (Baskerville et al., 2018).

Our work offers insights into how to achieve generalizability as projectability in DSR (Baskerville & Pries-Heje, 2014). An accumulated stock of design knowledge should be projectable. A projection is any relevant instance that supports the accumulated design knowledge (Baskerville & Pries-Heje, 2014). In our case, observations about problem solving involving the use of the instantiated artifact provided an impetus to develop a revised problem-class definition that was more general and projectable, as it encompassed the previous problem class while also extending beyond it (see Figure 3). In our DSR project, we increased the projectability of our design knowledge in two ways: (1) by adding actual projections of our current state of design knowledge to increase confidence that the prescriptive theory works; and (2) by increasing the number of possible projections to enhance the theory's potential to solve a broader range of concrete problems.

In the case of (1), we achieved greater projectability of our developed design knowledge by actively and deliberately switching between similar yet different problem-solving contexts (i.e., different companies experiencing different instances of the same problem class) and actually projecting our accumulating stock of design knowledge through new IT artifact instantiation and evaluation activities across these different contexts. Here, the evaluation feedback results may indicate that the solution might work in the old context but does not represent a satisficing solution in the new context. In that process of sampling for contexts to achieve greater projectability of our design knowledge, we focused on contexts that represented a balance between conformity and difference when compared to previously focused contexts in analogy with substantiation and extension strategies in grounded theory (Gregory et al., 2015). Our reasoning for seeking this balance was that our existing stock of design knowledge needed to be projectable to the new context (conformity) to increase its projectability, while the new context also needed to involve new problem-solving challenges (differences) to explore and perhaps extend its boundaries.

In the case of (2), we sought to increase the potential of our nascent design knowledge. This ocurred at two critical moments in our design knowledge development. In the transition between design cycles (see Sections 5.2 and 5.4), we learned from the evaluation of our instantiated artifact that it fully addressed the requirements of the defined problem class. At the same time, however, the feedback we received suggested unresolved adjacent or superordinate problems, which prompted us to redefine and extend the problem class.

In addition to gaining these insights regarding the two ways of achieving greater projectability of accumulated design knowledge over the course of a multi-context DSR project, we also reflected on the stopping rule of this process (i.e., either putting a halt to provoking opportunities to enhance projectability or pursuing the emerging opportunities to enhance projectability). In our DSR project, we decided as a team at some critical point to stop sampling for additional contexts for actually projecting our accumulated design knowledge into new versions of IT artifacts and carrying out concurrent evaluation activities across the sample of contexts. Our reasoning for "stopping" was our accumulated experience of having achieved sufficient levels of utility, in the sense of "value outside the development environment" (Gregor & Hevner, 2013, p. 15), based on the substantial positive feedback and evaluation results obtained across the various contexts and companies. Put a different way, we answered the question of whether we had identified a satisficing artifact design with a "tentative yes" (Gregory & Muntermann, 2014). The answer to this important question can be tentative only insofar as it relates to a concrete artifact design addressing a concrete problem in a specific set of contexts. Similar to doing grounded theory and reaching "saturation" (Glaser, 1978), the degree to which further iterations in IT artifact development and design knowledge accumulation yielded novel insights declined sharply, adding to our sense of having achieved an adequate level of design knowledge projectability under the given constraints. These constraints included our own limited capacities as designers ourselves in terms of design knowledge and access to relevant contexts that fit and thus could be used for further projecting activities (Baskerville, Kaul, & Storey, 2011).

6.3 Implications for Practice

Based on the emergent problem understanding, the developed solution components, and the results of the concurrent evaluations, we inferred three practical

implications for IT architects. First, monitoring IT architectural complexity has become a foundation for managing complexity of digital businesses, as IT architectures are symbolic representations of digitized products, services, and processes. Second, IT architects are able to contribute proactively to the digital business strategy of their firms by transforming their IT architectures into digital platforms. This involves reducing sociotechnical inertia by encapsulating IT architectural complexity. Third, we observed that the role of the IT function in the context of digital business strategy changes from a service provider and architect reducing IT architectural complexity toward a broker of platform services and architect monitoring IT architectural complexity.

7 Future Research and Limitations

In the course of our design science research study, we identified IT architectures as an important class of IT artifacts that are undergoing a metamorphosis in light of the rise of digital business strategy, which offers a wide array of possibilities for future research. In the past, IT architecture has often been viewed as a stable foundation upon which an enterprise can function, embodying the organizing logic of business processes, data, and IT capabilities reflecting the firm's key integration and standardization requirements (Ross, Weill, & Robertson, 2006). What we witness today in the digital business strategy context is competition between diverse logics and requirements (Tilson et al., 2010) associated with the fusion of IT within firms' environments (Woodard et al., 2013) as well as the fusion of business and IT strategy (El Sawy et al., 2010). The changing nature of IT architectures is a manifestation of this fundamental shift. IT architectures are continually enlarged, reduced, or modified (Tanriverdi et al., 2010) as they fuse with shared, open, heterogeneous, generative, and constantly evolving digital infrastructures (Hanseth & Lyytinen, 2010) and as businesses increasingly compete through their IT architectures—for example, through exposure of IT systems to a heterogeneous set of actors (i.e., customers, suppliers, business partners, and IT developers). In our research, these and related observations prompted us to draw on the complex systems theory (e.g., Maier, 1998; Simon, 1962; Sommerville et al., 2012), an idea that may also be relevant for future studies in the digital business strategy context.

A limitation of this study is that our theory development and evaluation activities in the last cycle focused on the banking industry because of the author team's particularly strong academic-industry relationships in that particular sector. For future work, we suggest drawing on our design, which we believe can be used to address both MCITA and MCITA-like problems (e.g., monitoring large-scale, networked IT, military, and national transportation systems complexities). In this vein, an interesting direction for future research could be to extend the subject being monitored from IT architecture to enterprise architecture, that is, to the high-level logic for business processes and IT capabilities (Ross et al., 2006, p. 48).

The present study focused on the perspective of individual firms and on their need to monitor the complexity of the IT architecture. We envision that our set of design principles will also provide value during the construction of IT artifacts for groups of firms as well as institutions overseeing groups of firms or entire markets. For example, regulators in the banking context could use our set of design principles to build a tool that is useful for assessing the complexity and systemic risk of the international banking sector.

An underlying assumption of the design principle development in this paper is bounded rationality (Simon, 1991) and the resulting need for IT support and a tool that enables human decision makers to monitor the complexity of IT architectures. As shown in related areas (e.g., monitoring the complexity of large-scale networked IT systems), the task of monitoring the complexity of IT architectures can, under certain conditions, be automatized and carried out by machines or "digital control systems" (Lee & Berente, 2012). It remains to be seen and explored in future studies whether, how, and under what conditions we can relax the assumption of bounded rationality and address (parts of) MCITA problems through such digital controls. The suggested focus of our theory to place problem-solving complexity as opposed to structural and dynamic complexity into the foreground may need to be revisited in the future as the level of automation increases and thus reduces problem-solving complexity.

Acknowledgments

We appreciate the contribution of the entire review team in helping us to improve the manuscript. We are thankful for the developmental review process and the helpful guidance of the senior editors and the associate editor. We thank the participating industry partners and the design teams for their extraordinary commitment, without which this research project would not have been possible. We would like to express special gratitude to Dr. Alexander Schütz for helping with data collection, artifact design, and providing feedback on prior versions of the paper. This study has received financial support from the Agencia Estatal de Investigación (AEI) of the Spanish Ministry of Science, Innovation and Universities—ECO2017-88576-R (MINECO/AEI/FEDER,UE) as well as industry partners.

References

Ågerfalk, P. J. (2014). Insufficient theoretical contribution: a conclusive rationale for rejection? *European Journal of Information Systems*, *23*(6), 593-599.

Ahlemann, F., Stettiner, E., Messerschmidt, M., & Legner, C. (eds.). (2012). *Strategic Enterprise architecture management: Challenges, best practices, and future developments*. Springer.

Aier, S., Kurpjuweit, S., Saat, J., & Winter, R. (2009). Enterprise architecture design as an engineering discipline. *AIS Transactions on Enterprise Systems*, *1*(1), 36-43.

Amaral, L. A. N., & Uzzi, B. (2007). Complex systems: A new paradigm for the integrative study of management, physical, and technological systems. *Management Science*, *53*(7), 1033-1035.

Ashby, W. R. (1956). *An introduction to cybernetics*. Chapman & Hall.

Ashby, W. R. (1960). *Design for a brain*. Wiley

Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity*. MIT Press.

Baskerville, R., Baiyere, A., Gergor, S., Hevner, A., & Rossi, M. (2018). Design science research contributions: Finding a balance between artifact and theory. *Journal of the Association for Information Systems*, *19*(5), 358-376.

Baskerville, R., Kaul, M., & Storey, V. (2011). Unpacking the duality of design science. *Proceedings of* the *Thirty-Second International Conference on Information Systems*.

Baskerville, R., & Pries-Heje, J. (2014). Design theory projectability. *Proceedings of Working Conference on Information Systems and Organizations.*

Becz, S., Pinto, A., Zeidner, L., Banaszuk, A., Khire, R., & Reeve, H. (2010). Design system for managing complexity in aerospace systems. *Proceedings of 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.

Beese, J., Aier, S., Haki, K., & Aleatrati Khosroshahi, P. (2016). Drivers and effects of information systems architecture complexity: A mixed-methods study. *Proceedings of the European Conference on Information Systems*.

Berente, N., Lyytinen, K., Yoo, Y., & King, J. L. (2016). Routines as shock absorbers during organizational transformation: Integration, control, and NASA's enterprise information System. *Organization Science*, *27*(3), 551-572.

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital business strategy: toward a next generation of insights. *MIS Quarterly*, *37*(2), 471-482.

Boh, W. F., & Yellin, D. (2006). Using enterprise architecture standards in managing information technology. *Journal of Management Information Systems*, *23*(3), 163-207.

Boyle, D., Keywood, M., & Roberts, J. (2012). *IT Complexity: The silent killer of business performance*. http://www.ey.com/Publication/vwLUAssets/IT_complexity__the_silent_killer_of_business_performance/$FILE/IT_complexity.pdf

Buckley, W. (1967). *Sociology and modern systems theory.* Prentice-Hall.

Bygstad, B. (2017). Generative innovation: A comparison of lightweight and heavyweight IT. *Journal of Information Technology*, *32*(2), 180-193.

de Reuver, M., Sørensen, C., & Basole, R. C. (2018). The digital platform: A research agenda. *Journal of Information Technology*, *33*(2), 124-135.

DeLaurentis, D., & Callaway, R. K. (2004). A system-of-systems perspective for public policy decisions. *Review of Policy Research*, *21*(6), 829-837.

Domerçant, J. C., & Mavris, D. N. (2011). Measuring the architectural complexity of military systems-of-systems. *IEEE Aerospace Conference Proceedings*.

Drnevich, P. L., & Croson, D. C. (2013). Information technology and business-level strategy: Toward an integrated theoretical perspective. *MIS Quarterly*, 37(2), 483-509.

El Sawy, O. A. (2003). The IS Core IX: The 3 faces of is identity: Connection, immersion, and fusion. *Communications of the AIS*, *12*(1), 588-598.

El Sawy, O. A., Malhotra, A., Park, Y., & Pavlou, P. A. (2010). Research commentary—Seeking the configurations of digital ecodynamics: It takes three to tango. *Information Systems Research*, *21*(4), 835-848.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. human factors. *The Journal of the Human Factors and Ergonomics Society*, 37(1),32-64.

Ethiraj, S. K., & Levinthal, D. (2004). Modularity and innovation in complex systems. *Management Science, 50*(2), 159-173.

Greefhorst, D., & Proper, E. (2011). *Architecture principles—The cornerstones of enterprise architecture.* Springer.

Gregor, S., & Hevner, A. (2013). Positioning and presenting design science research for maximum impact. *MIS Quarterly, 37*(2), 337-355.

Gregor, S., & Jones, D. (2007). The anatomy of a design theory. *Journal of the Association for Information Systems, 8*(5), 312-335.

Gregory, R. W., Keil, M., Muntermann, J., & Mähring, M. (2015). Paradoxes and the nature of ambidexterity in IT transformation programs. *Information Systems Research, 26*(1), 57-80.

Gregory, R. W., & Muntermann, J. (2014). Research note—Heuristic theorizing: Proactively generating design theories. *Information Systems Research, 25*(3), 639-653.

Guillemette, M. G., & Paré, G. (2012). Toward a new theory of the contribution of the IT function in organizations. *MIS Quarterly, 36*(2), 529-551.

Hall, A. D., & Fagen, R. E. (1969). Definition of system. In J.A. Litterer (Ed.), *Organizations: Systems, control and adaptation* (pp. 31-43). Wiley.

Hanseth, O., & Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: The case of building internet. *Journal of Information Technology, 25*(1), 1-19.

Henfridsson, O., & Bygstad, B. (2013). The generative mechanisms of digital infrastructure evolution. *MIS Quarterly, 37*(3), 907-931.

Henneman, R. L., & Rouse, W. B. (1986). On measuring the complexity of monitoring and controlling large-scale systems. *IEEE Transactions on Systems, Man, and Cybernetics, 16*(2), 193-207.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75-105.

Iivari, J. (2015). Distinguishing and contrasting two strategies for design science research. *European Journal of Information Systems, 24*(1), 107-115.

ISO/IEC/IEEE. (2011). Systems and software engineering: Architecture description. (2011:ISO/IEC/IEEE 42010). https://www.iso.org/standard/50508.html

Jacquemin, A. P., & Berry, C. H. (197 9). Entropy measure of diversification and corporate growth. *The Journal of Industrial Economics, 27*(4), 359-369.

Keen, P., & Williams, R. (2013). Value architectures for digital business: Beyond the business model. *MIS Quarterly, 37*(2), 643-647.

Lee, J., & Berente, N. (2012). Digital innovation and the division of innovative labor: Digital controls in the automotive industry. *Organization Science, 23*(5), 1428-1447.

Lerch, F. J., & Harter, D. E. (2001). Cognitive support for real-time dynamic decision making. *Information Systems Research, 12*(1), 63-82.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering, 1*(4), 267-284.

Murray, J., & Liu, Y. (1997). An experiment to assess task complexity in the supervision of networked systems. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*.

Murray, J., & Yili, L. (1994). A software engineering approach to assessing complexity in network supervision tasks. *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics: Humans, Information and Technology*.

Natu, M., Ghosh, R. K., Shyamsundar, R. K., & Ranjan, R. (2016). Holistic performance monitoring of hybrid clouds: Complexities and future directions. *IEEE Cloud Computing, 3*(1), 72-81.

Orlikowski, W., & Iacono, C. S. (2001). Research commentary: Desperately seeking the "IT" in IT research: A call to theorizing the IT artifact. *Information Systems Research, 12*(2), 121-134.

Peet, R. K. (1974). The measurement of species Diversity. *Annual Review of Ecology and Systematics, 5*(1), 285-307.

Peffers, K., Tuunanen, T., & Niehaves, B. (2018). Design science research genres: Introduction to the special issue on exemplars and criteria for applicable design science research. *European Journal of Information Systems, 27*(2), 129-139.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45-77.

Priem, R. L., Butler, J. E., & Li, S. (2013). Toward reimagining strategy research: retrospection and prospection on the 2011 AMR decade

award article. *Academy of Management Review*, *38*(4), 471-489.

Psiuk, M., & Zielinski, K. (2015). Goal-driven adaptive monitoring of SOA systems. *Journal of Systems and Software, 110*, 101-121.

Ribbers, P. M. A., & Schoo, K.-C. (2002). Program management and complexity of ERP Implementations. *Engineering Management Journal*, *14*(2), 45-52.

Richardson, G. L., Jackson, B. M., & Dickson, G. W. (1990). A principles-based enterprise architecture: Lessons from Texaco and Star Enterprise. *MIS Quarterly, 14*(4),385-403.

Ross, J. W. (2003). Creating a strategic IT architecture competency: Learning in stages. *MIS Quarterly Executive, 2*(1),31-43.

Ross, J. W., Weill, P., & Robertson, D. (2006). *Enterprise architecture as strategy: Creating a foundation for business execution*. Harvard Business Press.

Sanchez, R., & Mahoney, J. T. (1996). Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal, 17*(S2), 63-76.

Schilling, M. A. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *The Academy of Management Review, 25*(2),312-334.

Schilling, R., Beese, J., Haki, K., Aier, S., & Winter, R. (2017). Revisiting the impact of information systems architecture complexity: A complex adaptive systems perspective. *Proceedings of the International Conference on Information Systems*.

Schmidt, C., & Buxmann, P. (2011). Outcomes and success factors of enterprise IT architecture management: Empirical insight from the international financial services industry. *European Journal of Information Systems*, *20*(2),168-185.

Schneberger, S. L., & McLean, E. (2003). The complexity cross: Implications for practice. *Communications of the ACM, 46*(9), 216-225.

Schütz, A. (2017). *Komplexität von IT-Architekturen.* Wiesbaden: Springer Fachmedien.

Schütz, A., Widjaja, T., & Gregory, R. (2013a). Escape from Winchester Mansion: Toward a set of design principles to master complexity in IT architectures. *Proceedings of the International Conference on Information Systems*.

Schütz, A., Widjaja, T., & Kaiser, J. (2013b). Complexity in enterprise architectures: Conceptualization and introduction of a measure from a system theoretic perspective. *Proceedings of the European Conference on Information Systems*.

Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *MIS Quarterly, 35*(1), 37-56.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American philosophical society, 106*(6), 467-482.

Simon, H. A. (1991). Bounded rationality and organizational learning. *Organization Science*, *2*(1), 125-134.

Simon, H. A. (1996). *The Sciences of the Artificial*, (3rd ed.). MIT Press.

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., Mcdermid, J., & Paige, R. (2012). Large-scale complex IT systems. *Communications of the ACM, 55*(7), 71-77.

Tamaskar, S., Neema, K., & DeLaurentis, D. (2014). Framework for measuring complexity of aerospace systems. *Research in Engineering Design, 25*(2), 125-137.

Tamm, T., Seddon, P. B., Shanks, G., & Reynolds, P. (2011). How does enterprise architecture add value to organisations? *Communications of the Association for Information Systems, 28*(1), 141-168.

Tanriverdi, H., Rai, A., & Venkatraman, N. (2010). Research commentary: Reframing the dominant quests of information systems strategy research for complex adaptive business systems. *Information Systems Research, 21*(4), 822-834.

Tilson, D., Lyytinen, K., & Sørensen, C. (2010). Research commentary—Digital infrastructures: The missing IS research agenda. *Information Systems Research, 21*(4), 748-759.

Tiwana, A., & Konsynski, B. (2010). Complementarities between organizational IT architecture and governance structure. *Information Systems Research, 21*(2), 288-304.

Vierhauser, M., Rabiser, R., & Grünbacher, P. (2016). Requirements monitoring frameworks: a systematic review. *Information and Software Technology, 80*, 89-109.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an information system design theory for vigilant EIS. *Information Systems Research, 3*(1), 36-59.

Wareham, J., Fox, P. B., & Cano Giner, J. L. (2014). Technology ecosystem governance. *Organization Science, 25*(4), 1195-1215.

Weitzel, T., Beimborn, D., & König, W. (2006). A unified economic model of standard diffusion: the impact of standardization cost, network effects, and network topology. *MIS Quarterly, 30*(1), 489-514.

Widjaja, T., & Buxmann, P. (2009). Service-oriented architectures: Modeling the selection of services and platforms. *Proceedings of the 17th European Conference on Information Systems*.

Widjaja, T., & Gregory, R. W. (2012). Design Principles for heterogeneity decisions in enterprise architecture management. *Proceedings of the International Conference on Information Systems.*

Widjaja, T., Kaiser, J., Tepel, D., & Buxmann, P. (2012). Heterogeneity in IT landscapes and monopoly power of firms: A model to quantify heterogeneity. *Proceedings of International Conference on Information Systems*.

Winter, R., & Aier, S. (2011). How are enterprise architecture design principles used? *Proceedings of 15th Enterprise Distributed Object Computing Conference Workshops*.

Winter, R., & Fischer, R. (2007). Essential layers, artifacts, and dependencies of enterprise architecture. *Journal of Enterprise Architecture, May*, 1-12.

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

Xia, W., & Lee, G. (2005). Complexity of information systems development projects: Conceptualization and measurement development. *Journal of Management Information Systems, 22*(1), 45-83.

Yoo, Y., Boland, J. R. J., Lyytinen, K., & Majchrzak, A. (2012). Organizing for innovation in the digitized world. *Organization Science, 23*(5), 1398-1408.

Zinser, K., & Henneman, R. L. (1989). A model-based aid for monitoring and controlling a large-scale system. *IEEE Transactions on Systems, Man, and Cybernetics, 19*(4), 888-892.

Appendix A

Table A1. Selected Existing Design Knowledge about Monitoring Complex Systems

Source	Problem space	Solution space
Becz, S., Pinto, A., Zeidner, L., Khire, R., Reeve, H., & Banaszuk, A. (2010)	*Problem focus*: Managing complexity in aerospace systems to reduce cost and schedule overruns. *Problem understanding*: Technical complexity of the system is interrelated with complexity of system requirements, development teams and organizational partnerships, resulting in emergent behavior of the system as a whole.	*Artifact development*: A newly proposed design process for complex systems. *Artifact components*: Four key elements of the design process, including: (1) abstraction based design tools (that are able to handle different levels of abstraction); (2) quantitative complexity metrics; (3) advanced architecture synthesis methods (allowing for evaluation of different feasible architecture options); and (4) robust uncertainty management (to manage key risks).
Domerçant, J. C., & Mavris, D. N. (2011)	*Problem focus*: Acquisition of a complex military system of systems that provides a suppression of enemy air defenses capability. *Problem understanding*: Complexity is a key design issue for a system-of-systems architecture, defined as the structure of components, their relationships, and the principles and guidelines governing their design evolution over time.	*Artifact development*: Method to define and measure complexity of a military system of systems. *Artifact components*: Number of functionally and physically distinct system types; number of functions performed by each system; number of network interfaces used to transmit data/information; interface complexity multiplier; cyclomatic complexity (to understand interactions among systems and coordination of diverse system functions).
Henneman, R. L., & Rouse, W. B. (1986)	*Problem focus*: Monitoring a communication network to identify failures by human operators. *Problem understanding*: Complexity of large-scale systems is viewed as being a result of both the structure of the system and the human operator's understanding of the system.	*Artifact development*: CAIN (contextually augmented integrated network). *Artifact components*: Measures of structural complexity of the system, including the physical system and the human-system interface, and measures of strategic complexity that capture operator performance, including node failures and subject's paths through the network to resolve those failures.
Maier, M. W. (1998)	*Problem focus*: Design of large-scale systems of systems such as integrated air defense networks, the internet, and enterprise information networks. *Problem understanding:* System is defined as an assemblage of components that produces behavior or functionality not available from any component individually. System of systems is proposed to be an assemblage of components that individually may be regarded as systems, and that possess two additional properties, i.e., operational and managerial independence of the components.	*Artifact development*: Four abstract design principles are proposed to architect a system of systems. *Artifact components*: (1) stable intermediate forms (so systems are capable of operating and fulfilling useful purposes); (2) policy triage (choosing very carefully what systems to control, as overcontrol will fail for lack of authority and undercontrol will eliminate the system nature of the integrated system); (3) leverage at the interfaces (paying more attention to the interfaces than the components of the system due to high interdependence of components); (4) ensuring cooperation (taking a joint utility approach and ensuring that each participant's wellbeing is partially dependent on the wellbeing of other participants).
Natu, M., Ghosh, R. K., Shyamsundar, R. K., & Ranjan, R. (2016)	*Problem focus*: Performance monitoring of hybrid clouds. *Problem understanding*: Monitoring the continuous evolution of enterprise systems, especially in hybrid cloud computing environments, is a problem of monitoring complexity, including the databases, operating systems, and cloud-based storage and network devices underlying applications that serve business functions.	*Artifact development*: Conceptual ideas for an IT artifact and solution. *Artifact components*: Referred to as "insights" in the paper: (1) need for a comprehensive system-wide solution (holistic view of entire complexity of IT architecture); (2) need for solutions that adapt to system changes (adaptive monitoring, adaptive probing); (3) need for efficient ways to manage scale (invariant detection, multiresolution analysis, noise reduction); and (4) need to be proactive rather than reactive (through use of analytics).
Psiuk, M., & Zielinski, K. (2015)	*Problem focus*: Monitoring of service-oriented architecture (SOA). *Problem understanding*: To tackle the problem of increasing system complexity, adaptive monitoring must occur across SOA layers on the basis of monitoring goals defined by the user.	*Artifact development*: DAMON (Dynamic Adaptive MONitoring). *Artifact components*: Automated monitoring instrumentation (during the startup process of the service container; monitoring mechanisms (topology discovery, causality identification and measurement acquisition); and realization of the monitoring process (nominals identification, sentinels selection and adaptive drill-down).

Appendix B

Application of the Entropy Measure to Quantify IT Heterogeneity

The foundational idea of Tool B is the transfer measures of heterogeneity from different contexts to the domain of IT architecture.[4] The tool allows the incorporation of different measures of heterogeneity (e.g., the Herfindahl-Hirschman-Index and the Entropy Measure discussed in Jacquemin & Berry, 1979) and supports the analysis of different types of heterogeneity in IT architectures. In the following, we use the entropy measure to illustrate. A classical use of the entropy measure in the field of economics involves quantifying market concentrations. Here, f_i denotes the relative market share of firm i.

$$\text{entropy measure} = \sum_{i=1}^{n} f_i \, ln\left(\frac{1}{f_i}\right)$$

A high index value denotes a low concentration. The entropy measure takes a minimum value of 0 in a "monopoly" and reaches its maximum for values with an equal distribution. Applied to the MCITA problem, an example would involve quantifying systems variety with respect to partnering firms in a platform-based business, where f_i can be interpreted as the relative share of solution components provided by firm i. A high entropy measure index value denotes a high variety of solution components with respect to the firms that provide them. The entropy measure takes a minimum value of 0 in a monopoly (single-partner scenario) and reaches its maximum for values with an equal distribution (multi-partner scenario). The tool can be used to quantify heterogeneity for all kinds of elements of an IT architecture, such as heterogeneity of semantics in databases, software vendors in application architectures, suppliers of hardware, and performance of clients.

The entropy measure captures the both facets "evenness" and "richness" of the conceptualization of systems variety described in the second heuristic theorizing cycle (Section 5.3). Accordingly, applied to the example above, the entropy measure increases with a larger number of different partnering firms (richness) and with a higher parity of the prevalence of partnering firms (evenness). To understand better which of the two facets of variety is the main driver of the entropy measure, at least one of the two facets (richness or evenness) needs to be considered in addition to the overall entropy measure (if one of the two facets is known, the other facet can be easily derived). Drawing on the overall entropy measure in addition to at least one of the two facets (richness or evenness) offers the advantage of combining an aggregated and detailed view of systems variety. This is illustrated in the following example: The distributions 5%-5%-10%-80% (four partnering firms contributing to the ecosystem solutions portfolio with differing intensity) and 50%-50% (two partnering firms contributing with the same intensity) lead to nearly the same value of the entropy measure value (0.70 and 0.69, respectively). However, these two scenarios are very different. In the first scenario, the observed variety stems primarily from the richness of the ecosystem. In the second scenario, the observed variety stems primarily from evenness.

[4] The underlying model is based on (Widjaja et al., 2012).

About the Authors

Thomas Widjaja is professor of business information systems at the School of Business, Economics and Information Systems at the University of Passau, Germany. His research interests include digital services, data-driven business models, and IT architecture management. His work has appeared in journals such as *Information Systems Research*, *Journal of Strategic Information Systems*, *OR Spectrum*, and *Journal of the Association for Information Systems*.

Robert Wayne Gregory is associate professor of IT and entrepreneurship at the University of Virginia, McIntire School of Commerce. His research focuses on strategic IT management and digital business. He received the AIS Early Career Award in 2016. His research has been published in top-tier journals including *MIS Quarterly* and *Information Systems Research*. Robert serves on the editorial review board of the *Journal of the Association for Information Systems*.