ESCAPE FROM WINCHESTER MANSION – TOWARD A SET OF DESIGN PRINCIPLES TO MASTER COMPLEXITY IN IT ARCHITECTURES

Completed Research Paper

Alexander Schütz
Technical University Darmstadt
Chair of Information Systems
Hochschulstr. 1
64289 Darmstadt, Germany
schuetz@is.tu-darmstadt.de

Thomas Widjaja
Technical University Darmstadt
Chair of Information Systems
Hochschulstr. 1
64289 Darmstadt, Germany
widjaja@is.tu-darmstadt.de

Robert Wayne Gregory
IESE Business School
Information Systems Department
Avda Pearson 21
08034 Barcelona, Spain
RWGregory@iese.edu

Abstract

Although the management of complexity is a central task of CIOs and IT architects, in-depth examinations and the development of design theory in this area is, to the best of our knowledge, underrepresented in existing IS literature. Especially theory-based guidelines and information systems for the management of IT architecture complexity are missing. In a joint team of practitioners and researchers, we applied the action design research (ADR) method in order to tackle this class of problems, i.e., IT architecture complexity management. We derived a set of seven design principles (that guide the design of an information system that supports IT architects to manage IT architecture complexity), which we evaluated and enriched during multiple 'building, intervention and evaluation' cycles, according to ADR. In addition, we simultaneously implemented and evaluated a material artifact (i.e., a piece of software) for IT architecture complexity management.

Keywords: Complexity, IT Architecture, Design Principles, Action Design Research

Introduction

For about 38 years the mysterious Winchester Mansion was under construction. Due to continuous expansion (e.g., to 160 rooms, 2 Ballrooms, and 10.000 windows) and recurrent remodeling of rooms, a number of strange architectural facets emerged such as a staircase that descends seven steps only to rise eleven steps back up again. An earthquake in 1906 trapped Sarah Winchester – the widow of William Wirt Winchester – in her bedroom and it took the servants hours to locate and rescue her (Winchester Mystery House 2013). This unfortunate outcome was the result of emerged architectural complexity. Architectural complexity is also increasingly being debated in information technology (IT) as homegrown legacy system environments in organizations created an almost unmanageable complex net of system components and relations between them (Ross et al. 2006). Further drivers of IT complexity include emerging novelty in business requirements, introduction of new technologies, and company growth (organic or inorganic, e.g., through mergers) that trigger changes made to the organization's IT environment (Mattern et al. 2003; Schmidt and Buxmann 2011). Especially in the financial service sector, which provides information-intense products (Porter and Millar 1985), IT landscapes emerged, consisting of a huge number and heterogeneity of applications and underlying infrastructure components.

IT architecture complexity is an important issue, because the above described historically emerged IT landscapes often constitute the basis for further IT architectural decisions. Existing literature offers rich descriptions of managerial approaches and strategies that may be useful as a starting point to learn how to deal with this challenge of managing IT architectural complexity (e.g., Ross et al. 2006). Further studies in this area only address sub-aspects of IT architecture complexity, such as the standardization problem (e.g., Boh and Yellin 2006). Recent contributions discuss measurability of an IT architecture's complexity as a prerequisite for IT complexity management (Schütz et al. 2013; Widjaja et al. 2012), but in-depth examinations and the development of design theory in this area is, to the best of our knowledge, underrepresented in existing literature. Former research demonstrates that information systems can support management in decision-making (e.g. Markus et al. (2002)). In summary, research is needed to generate knowledge on how information systems that support IT architecture complexity management should be designed. Hence, we strive to answer the following research question (RQ): *Which design principles should guide the design of an information system that supports IT architects to manage IT architecture complexity?*

After discussing the topic of managing IT architecture complexity with a variety of different organizations experiencing this problem, we decided to focus on a collaborative research-practitioner relationship with a single bank who showed a particular interest for a joint research and problem solving project (referred to as 'the bank' subsequently). The IT architects of the bank became collaborative partners for the exchange of experiences, ideas, and problem solutions during the course of this research project and agreed with us that there was an urgent need to tackle the above explained gap. We decided to adopt an action design research (ADR) approach (Sein et al. 2011) due to the equal emphasis of problem solving in practice (which the bank was interested in) and the development of nascent design theory (i.e., design principles) for an unexplored class of problems, i.e., IT architecture complexity management, which is our primary purpose in this paper.

The reminder of the paper is structured as follows: In the section 'Theoretical Background' we give a brief overview of Enterprise-/IT architecture and IT complexity in IS literature. Next, we describe ADR as the adopted research method and clarify briefly how the approach was applied to the joint project with the IT organization of the bank. Thereby, we describe the building process of the IT artifact and the continuous evaluation and refinement of the artifact based on the use context. Afterwards we introduce a formulation of the discussed problem, which emerged through cooperation with the IT architects of the bank. We then present the results of the ADR project, which emerged during the design cycles by building upon a theory-ingrained artifact. In line with ADR, we also discuss and formalize our learning. Finally, we give a short summary of the results, highlight limitations of the approach and suggest avenues for further research.

Theoretical Background

In this paper we focus on IT architecture complexity which is considered to be a sub-problem of Enterprise Architecture (EA) complexity. To ensure a common understanding of the terms EA, IT architecture and IT complexity, we discuss and propose working definitions of these terms.

Enterprise- /IT Architecture

Established EA frameworks such as TOGAF (The Open Group Architecture Framework; The Open Group 2011) or EAP (Enterprise Architecture Planning; Spewak and Hill 1993) apply a hierarchical separation into four architecture dimensions: business process, data/information, application and infrastructure. The literature on EA proposes a similar separation. Richardson et al. (1990) define EA as follows: "[Enterprise Architecture] defines and interrelates data, hardware, software, and communications resources, as well as the supporting organization required to maintain the overall physical structure required by the architecture" (Richardson et al. 1990, p. 386). Thus, Richardson et al. (1990) definition does not include the business process domain as part of an EA. Ross (2003), in contrast, includes business processes: "In some firms the enterprise IT architecture acts as a tool for aligning IT and business strategy. This alignment focuses on the IT components that enable critical business processes. Thus, at the enterprise level, an IT architecture is the organizing logic for applications, data and infrastructure technologies, as captured in a set of policies and technical choices, intended to enable the firm's business strategy" (Ross 2003, p. 31).

In summary, based on the above discussion, that EA comprises more than IT (data, application and infrastructure), we focus on the analysis of IT architecture (see Figure 1 for our view on the relation between EA and IT architecture).

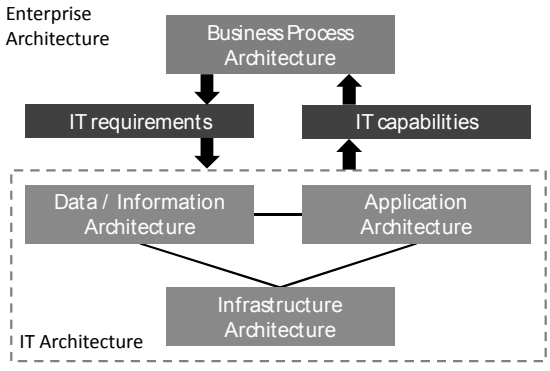

Figure 1. Enterprise-/IT Architecture

IT architecture has been defined as "the organizing logic for applications, data and infrastructure technologies [...]" (Ross 2003, p. 31). Based on the IEEE Standard 1471-2000 (IEEE Architecture Working Group 2000) we consider architecture as a system, embodied in its components and relations to each other. This is in line with the systems science literature, which characterizes a system as a set of system components ('things') and relations between the components (Klir 2001). Considering the IT architecture as a system allows us to further decompose this hierarchic system (Simon 1962) into the interrelated subsystems 'Data/Information Architecture', 'Application Architecture', and 'Infrastructure Architecture'. Because the components and relations of these subsystems have different characteristics, this decomposition enables us to separately analyze their respective degree of complexity, while also considering their interrelations at the subsystem level.

Recent literature has identified the management of an IT architecture's complexity as one fundamental task of the IT function in organizations. Guillemette and Paré (2012) coined the term 'Architecture Builder' for this function and describe "to build and manage an IT infrastructure that supports business

processes and reduces architectural complexity" (Guillemette and Paré 2012, p. 532) as main responsibility of this function.

Complexity in IT Architectures

In IS literature the term complexity is used in various contexts. Xia and Lee (2005) refer to complexity in connection with information system development projects and a considerable amount of literature has been published on task complexity (e.g., Francalanci and Piuri (1999), Jiang and Benbasat (2007), Roberts et al. (2004)). Some studies also focus particularly on software complexity (e.g., Berlinger (1980), Davis and LeBlanc (1988), Harrison (1992)), and data complexity (e.g., Benedikt et al. (2003), Vardi (1982)). A more technical focus on communication complexity of a protocol has been emphasized by Awerbuch and Schulman (1997). In recent years, there has been an increasing amount of literature on complexity of (adaptive) systems (e.g., Kim and Kaplan (2006), Morel and Ramanujam (1999), Nan (2011)).

Schütz et al. (2013) conducted a structured literature review to analyze how the term complexity is defined in IS literature. Most contributions focused on complexity in a particular context so that the proposed concepts only cover specific parts of an IT architecture. Opposed to this, Schütz et al. (2013) identified the contributions of Schneberger and McLean (2003) and Henningsson and Hanseth (2011) as holistic concepts. These contributions define complexity as the number and heterogeneity (or variety) of the components and relations (cf. Figure 2). Because this perception is in line with our definition of an IT architecture as a system, embodied in its components and relations to each other, and in accordance with Schneberger (1997), Schneberger and McLean (2003), Henningsson and Hanseth (2011) and Schütz et al. (2013), *we define complexity as the number and the heterogeneity of the components and relations of an IT architecture*.

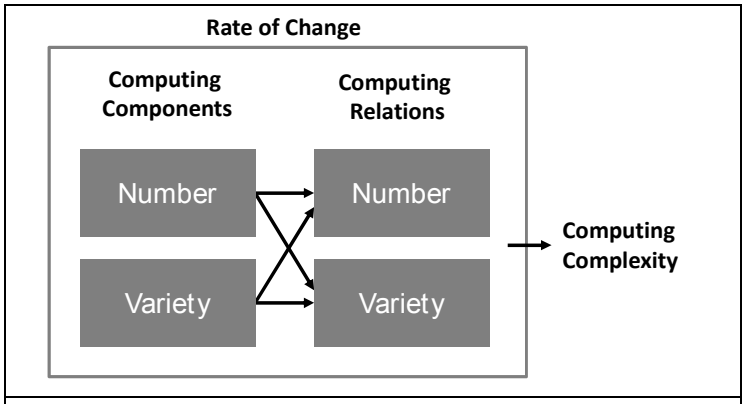

Figure 2. Conceptualization of Complexity based on Schneberger and McLean (2003)

To the best of our knowledge, concepts and information systems for IT architecture complexity management have, so far, not been proposed in IS literature. There exist EA management information systems such as planningIT (alphabet AG), ARIS IT Architect, ARIS ArchiMate Modeler (Software AG/IDS Scheer) or Rational Software Architect (IBM), which are used for the management of IT architectures. However these information systems are not based on a comprehensive conceptualization of complexity in IT architectures.

Method: Action Design Research

The research presented in this paper is part of a long term research program on complexity in IT architectures. Related publications resulted from this research program discuss design principles for heterogeneity decisions in EA management (Widjaja and Gregory 2012), heterogeneity in IT landscapes (Widjaja et al. 2012) and a conceptualization and a measure of complexity in EA (Schütz et al. 2013). This study builds upon the research insights generated so far in this program.

We decided to adopt an engaged scholarship approach in this research project because of the high practical relevance and lack of theory about IT complexity management in EA, which lends itself to an approach that emphasizes new theory development that is of highest practical relevance (Van de Ven 2007). Van de Ven (2007) discussed four different types of engaged scholarship, two of which are action research and design science. Recently, a method has been proposed that combines these two engaged scholarship types, i.e., the Action Design Research (ADR) method (Sein et al. 2011), which is in line with prior prominent design theory development exemplars (Lindgren et al. 2004; Markus et al. 2002). In brief, the key idea of ADR is to combine an IT-based organizational intervention and observation (i.e., the 'action' component) with a research perspective that has staked its ground in IS as the design science research (DSR) paradigm (Hevner et al. (2004) as well as Gregor and Hevner (2013) provide good introductions into the nature and characteristics of this paradigm, among others). ADR was proposed on rightful grounds because significant DSR programs involving both instantiated, material artifacts and abstract artifacts (Gregor and Jones 2007) oftentimes involve organizational interventions when embedding an instantiated artifact in the respective problem environment (e.g., Markus et al. (2002), Lindgren et al. (2004)). ADR fitted particularly well for our research purposes because our goal from the outset was to solve a practical problem in a real-world environment (Briggs et al. 2011), i.e., by developing and implementing an artifact for IT complexity management, while also reflecting and abstracting upon the experiences in that process to build Level 2 contributions (e.g., abstracted design principles) based on that Level 1 contribution (i.e., the instantiated artifact or instantiated design principles) (Gregor and Hevner 2013). For these purposes, we engaged in a collaborative ADR relationship with an industry partner from the European banking industry who currently experiences and deals with the problem of IT complexity management in banking IT architecture. In the following, we summarize the ADR method and how we applied this method in our research project.

ADR involves four stages, each with an associated set of research tasks, which relate to each other in an intertwined way. Table 1 summarizes the research tasks typically involved in each stage and briefly states the research outcomes of conducting each of these tasks in our specific research project. The first stage 'Problem Formulation' illustrates the nature of ADR as an engaged scholarship approach (Van de Ven 2007) in emphasizing the practically grounded formulation of a research opportunity. The second stage 'Building, Intervention and Evaluation' (BIE) includes subsequent 'BIE cycles' to create an initial design and further shaping of the IT artifact due to the organizational intervention and evaluation. The third stage 'Reflection and Learning' parallels the first two stages by reflecting on the design and redesign of the artifact and the evaluation of the suggested design principles, and "moves conceptually from building a solution for a particular instance to applying that learning [during the first two stages] to a broader class of problems" (Sein et al. 2011, p. 44). The fourth and last stage 'Formalization of Learning' is used to generalize the outcomes to an abstract level, applicable for a class of problems. Due to a permanent shaping of the IT artifact by the organizational context and the experience embedded during the artifact's development, use and refinement process, ADR focuses on artifacts as ensembles (Orlikowski and Iacono 2001).

One of the underlying assumptions of the ADR method is that evaluation of the nascent artifacts and design theory occurs concurrently in an ongoing process. In that sense, ADR adds a new perspective to the generally held belief that DSR requires a final evaluation at the end of the research process as suggested by March and Smith (1995) or Hevner et al. (2004). Following the ADR method, the rigor of the design theory building process is ensured by a concurrent evaluation during the BIE cycles.

With regard to the stated research question, i.e., the demand for design principles to guide the design of an information system that supports IT architects to manage IT architecture complexity, the focal object of the design principles derived in this research is the information system's design. Therefore, the design principles derived in the paper at hand guide what should be addressed when designing such an information system.

Table 1. ADR Stages, Tasks, and Outcomes of Application in this Research (based on Sein et al. (2011))

Stage (S)	Research Task (T)	Outcomes
Problem Formulation (S1)	Identify and conceptualize the research opportunity (S1.T1)	The need of the IT architects of the bank to measure and manage the IT architectures complexity in order to build a basis for an improved decision-making was the initial trigger of our presented research.
	Formulate initial research questions (S1.T2)	With reference to the stated class of problems we formulate the following RQ: Which design principles should guide the design of an information system that supports IT architects to manage IT architecture complexity?
	Cast the problem as an instance of a class of problems (S1.T3)	"IT architecture complexity management" is defined as a class of problems (cf. this section).
	Identify contributing theoretical bases and prior technology advances (S1.T4)	A theoretical model of complexity proposed by Schneberger and McLean (2003) was used in combination with a system theoretic view on IT architectures (cf. section 'Theoretical Background' and subsection 'Theory-Ingrained Initial Design Principles' in section 'Results').
	Secure long-term organizational commitment (S1.T5)	The researchers had the possibility to participate in all stages of the development process and beyond that (cf. this section).
	Set up roles and responsibilities (S1.T6)	The ADR team was structured into three disjunctive groups: researchers, IT architects, and the implementation team. Thereby, theoretical, practical, and technical perspectives were simultaneously considered (cf. this section).
Building, Intervention, and Evaluation (BIE) (S2)	Discover initial knowledge-creation target (S2.T1)	Addition of design principles for "IT architecture complexity management" to the knowledge base.
	Select or customize BIE form (S2.T2)	Because the expected result of the project was not specified in detail, the development of the artifact was permanently evaluated in applying an IT-Dominant BIE cycle. Therefore, IT-COM was initially evaluated by the ADR team and subsequently by a selected set of end users in the bank (cf. this section).
	Execute BIE cycle(s) (S2.T3)	Table 2 provides an overview over the design process and the four different cycles we went through.
	Assess need for additional cycles, repeat (S2.T4)	Using scrum as an agile software development framework allowed for continuous evaluation and additional cycles (cf. this section).
Reflection and Learning (S3)	Reflect upon the design and redesign during the project (S3.T1)	Continuous intervention and evaluation led to cyclic reflection, revision, and introduction of the design principles as well as changes and adaption of new requirements in the artifact IT-COM, which emerged during the BIE stage (cf. Table 2 and section 'Results').
	Evaluate adherence to principles (S3.T2)	The evaluation of the adherence to principles took place during the BIE cycles as shown in the section 'Results'. During the above mentioned continuous intervention and evaluation of emerged and subsequently refined design principles, we verified whether they appropriately captured problem solving reality, i.e., the evaluation of adherence to principles.
	Analyze intervention results according to stated goals (S3.T3)	During the above mentioned continuous intervention and evaluation we also compared newly generated information and insights with our stated goals and defined requirements of our problem class.
Formalization of Learning (S4)	Abstract the learning into concepts for a class of field problems (S4.T1)	We generalized the solution by deriving abstract design principles applicable to each representative of this class of problems as stated in S1.T3 (cf. subsection 'Theoretical Implications' in section 'Discussion of Results').
	Share outcomes and assessment with practitioners (S4.T2)	We discussed the abstracted design principles (c.f. Table 5) with the IT architects. The alpha-version of the artifact IT-COM was evaluated during the BIE-cycle and the beta-version was distributed among end users in the environment.
	Articulate outcomes as design principles (S4.T3)	We formulated seven design principles for an IT complexity management information system, depicting IT-COM, implemented for the cooperating bank, as an instance of a solution to the stated class of problems (cf. Table 4 and Table 5).
	Articulate learning in light of theories selected (S4.T4)	We discussed the learning with respect to the theories selected in subsection 'Theoretical implications' in section 'Discussion of Results'.
	Formalize results for dissemination (S4.T5)	The generalized design principles are applicable to the stated class of problems (cf. Figure 6 and Table 5 in subsection 'Theoretical implications' in section 'Discussion of Results'.

Literature in the EA domain offers an understanding concerning the nature of design principles and the way principles should be presented (cf. Fischer et al. (2010), Winter and Aier (2011)). Greefhorst and Proper (2011), Richardson et al. (1990), and Widjaja and Gregory (2012) suggest to specify principles in the form of three facets: (1) A statement communicating the fundamental rule, (2) a rationale explaining why this principle should be followed, and (3) the implications identifying the requirements for carrying out the principle. We apply the proposed structure, though we report the facets in different sections of the paper. Statement and rationale of the derived design principles are presented in Table 5 in the section 'Results'. Because the level of abstraction of the implications is relatively low, these are presented in Table 6 in the section 'Discussion of Results'. Due to the ADR context we use the term 'consequences'.

As briefly stated before, the basis for this ADR project formed a joint cooperation agreement with the IT architects of a European bank. Through this cooperation, the artifact IT-COM, as an information system for IT architecture complexity management, was developed and implemented, which formed the basis for the theory development presented in this paper. The ADR team was structured into three disjunctive groups: the ADR researchers (hereinafter also referred to as 'we' or 'researchers'), the IT architects (or 'practitioners') and the software implementation team. The IT architects were very engaged in collaborating with us on applying the ADR method because they saw great value in solving the stated problem jointly. As a result, we had the possibility to collaborate on all stages of the development process, from the planning through to handover of the final implementation results. Furthermore, we had access to the complete project documentation and attended all meetings involving both the IT architects and the implementation team. By engaging in this ADR project and relationship, we formulated (a) "IT architecture complexity management" as the examined class of problems (S1.T3) and (b) "IT architecture complexity management in the environment of the bank" as an instance of the class of problems (a) in the context of the bank.

Material artifact implementation in this research project started as greenfield development without any foundational basis and placing an emphasis on an innovative technological design as a starting point for design and action research. Therefore, we decided in our ADR team to apply an IT-dominant BIE (note that Sein et al. (2011) distinguish between IT-dominant and organization-dominant BIE which differ with regards to the primary source of innovation). Furthermore, the ADR team agreed on scrum (Schwaber and Beedle 2002) as an agile software development framework to facilitate a continuous intervention and evaluation already in the early stages of the development process. Given that the expected result of the project was not specified in detail from the beginning, the use of scrum turned out as an appropriate decision.

The entire duration of the implementation was fifteen weeks, divided into seven 'two-week sprints'. A sprint can be understood as a working cycle in scrum. It represents a kind of mini-project in which a set of requirements is developed (Schwaber and Beedle 2002). At the end of each sprint, a project evaluation meeting was conducted. The entire ADR team attended the meetings (including planning meeting, kickoff and final presentation: ten meetings, approx. 90 minutes each). In addition to these meetings we had four workshops (approx. 120 minutes) and two interviews (approx. 60 minutes) with the IT architects to recurrently evaluate the emerging state of the implementation and the adequacy of the suggested design principles. We subdivided the ADR project into four BIE cycles. Table 2 provides an overview over the design process and the four different cycles we went through, and depicts how the sprints were distributed among the BIE cycles. Table 2 shows furthermore which groups were involved in the respective BIE cycle.

Table 2. Overview of the BIE Cycles (BIEC) of the ADR Project

	BIEC 1: Suggestion of Initial Design Principles	**BIEC 2:** Requirements Analysis and Mockup Design (Sprint 1-3)	**BIEC 3:** Complexity Measures and Data Quality (Sprint 4)	**BIEC 4:** Time Series Analysis, Drill-Down (Sprint 5-7)
ADR Researchers	X	X	X	X
IT Architects	X	X	X	X
Implementation Team		X	X	X
End Users				X

In the first BIE cycle (BIEC1) we derived the initial set of design principles that were both practically and theoretically informed. Due to the fact that the implementation team essentially had to deal with technical issues and the setup of the implementation environment at the beginning of the project, we combined the sprints one to three in the second BIE cycle (BIEC2). Objectives of the third BIE cycle (BIEC3) (which matched the forth sprint) were the implementation and evaluation of the complexity measures and the influence of data quality on the measures and on IT-COM as a whole. During the sprints five to seven the functions 'time series analysis' (to observe and compare a certain complexity over the time) and 'drill-down' (to enable analysis on different levels of abstraction) emerged. This was part of the fourth BIE cycle (BIEC4).

During the development of the alpha-version of the artifact (BIEC1 to BIEC3) the BIE-cycles were characterized by an evaluation of early designs by the members of the ADR Team, especially the IT architects and the researchers. This ensured a practical as well as a theoretical perspective on the resulting artifact. In accordance with the procedure proposed by Sein et al. (2011), the IT architects focused on intervention. Furthermore, they critically reflected upon the emerging design principles with regards to the problem instance and the organizational environment of the bank, and shared the insights. During BIEC4, the IT architects presented the beta-version to a wider set of end users in the bank and we discussed the feedback received in the course of the workshops.

Results

The following subsections present the results of our ADR project, starting with a formulation of the problem and the theoretically informed initial design principles as the result of the first BIE cycle. Afterwards we explain how these design principles and the artifact were revised during the course of the project.

Formulating the Problem as Complexity in IT Architectures

The presented research is grounded in an intensive collaboration with the IT organization of the bank. In this context, we held meetings and conducted joint workshops on general EA topics with the bank's IT architects. It turned out that a lot of issues they deal with were related, or even the result of the ongoing growth of IT architectural complexity (e.g. architecture governance, IT development planning, procedures for evaluating an IT architecture and complexity measures themselves). In addition, we attended regular meetings of regional IT architects from various industries. It became apparent that IT architecture complexity is not only a key IT challenge in the banking sector, but also of IT organizations in many other industries. Therefore, IT architecture complexity management is not only an individual problem of one company or unique to a specific industry. Rather, this can be seen as a class of problems. Although there exist different information systems to document the current state of an IT architecture, there are no information systems available to support IT architects to manage the complexity of an IT architecture. This deficiency makes it difficult to assess subsections of an IT architecture, not to mention a development plan of a future company-wide IT architecture.

Inspired by this practical need, we realized the challenges to conceptualize complexity in IT architectures (Schütz et al. 2013), to find a way to measure this IT complexity and to support the development of an artifact for IT architecture complexity management. These challenges are reflected in the problem formulation mentioned above: (a) "IT architecture complexity management" as a class of problems and (b) "IT architecture complexity management in the environment of the bank" as an instance of the class of problems in the context of the bank.

One of our assumptions in the study is that complexity is not necessarily always bad and that it is not only about reducing complexity, but finding the right degree of complexity. Therefore we want to propose design principles that guide the design of an information system that supports IT architects to manage IT architecture complexity in order to reach an efficient application of IT.

Theory-Ingrained Initial Design Principles

The ADR approach starts with a theory-ingrained artifact which in our specific case forms the key component of the first BIE cycle (BIEC1). Therefore, we refer to the conceptualization of complexity as introduced by Schütz et al. (2013), complemented by the 'rate of change' proposed by Schneberger and McLean (2003). Consequently the conceptualization of complexity is composed of the number and heterogeneity of components, the number and heterogeneity of the relations between these components, as well as the rate of change of the considered environment. The initial set of design principles was based on literature and joint workshops in preparation for the implementation project. Again, we emphasize the underlying idea to consider an IT architecture as a system, embodied in its components and relations to each other (Klir 2001; Schmidt and Buxmann 2011; Simon 1962). Figure 3 summarizes the proposed conceptualization of complexity in IT architectures.

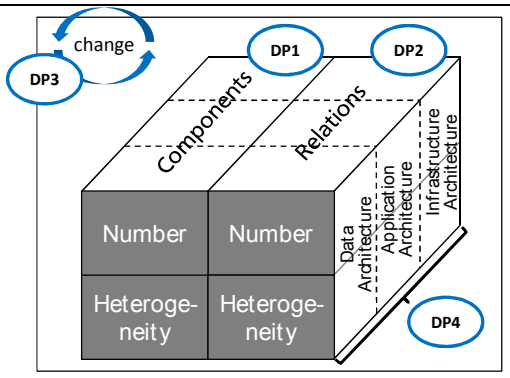

Figure 3. Conceptualization of Complexity in IT Architectures (cf. Schütz et al. (2013) and Schneberger and McLean (2003))

The following design principles (DP), while deduced from prior theory and literature, were subsequently discussed in the ADR team together with the practitioners prior to the start of implementation. The objective of doing this was to ensure practical grounding, in addition to theoretical grounding, of the DP's (see Table 3 for description). Note that DP 1 to DP 4 are based on the assumption that, to manage complexity, it is the needed to measure IT architecture complexity.

Table 3. Initial Set of Abstract Design Principles

Design Principle	Description: To master complexity…	Source
DP 1: Number and heterogeneity of components	…the number and heterogeneity of the system's components should be considered.	Henningsson and Hanseth (2011), Klir (2001), Schneberger and McLean (2003), Schütz et al. (2013), Simon (1962), Widjaja et al. (2012)
DP 2: Number and heterogeneity of relations	…the number and heterogeneity of the system's relations should be considered.	Henningsson and Hanseth (2011), Klir (2001), Schneberger and McLean (2003), Schütz et al. (2013), Simon (1962), Widjaja et al. (2012)
DP 3: Rate of change	…the rate of change of the system should be considered.	Henningsson and Hanseth (2011), Klir (2001), Schneberger and McLean (2003), Simon (1962)
DP 4: IT architecture domains	…the domains 'data/information architecture', 'application architecture' and 'infrastructure architecture' should be considered.	Richardson et al. (1990), Ross (2003), Schütz et al. (2013), Spewak and Hill (1993), The Open Group (2011)

Early workshops showed that the applied conceptualization of complexity was in accordance with the way of thinking of the IT architects of the bank, as the following quote during BIEC 1 of a leading IT architect illustrates: *"We needed to consider what 'IT complexity' means, which characteristics of complexity we want to examine and where we find objects in the company, which are suitable and reasonable to measure. […] Therefore, we focus on the structural complexity of a system or a cluster (thus several applications of a functional domain) that result from the number and diversity of its components and from the number and the diversity of the relationships between these components. […] And because we are architects, we should consider the architectural layers: data, application, and technology architecture." (Principal Architect)*

Iterative Revision and Emergence of Design Principles

In the following, we present the overall results that emerged from going through BIE cycles two, three, and four by building upon the theory-ingrained artifact of the first BIE cycle. During the course of the implementation project, i.e., during the cycles of the BIE-Stage (BIEC2-4), three of the initial stated design principles were confirmed (DP 1, 2, and 4), one had to be revised (DP 3), and three additional design principles emerged.

The bank's IT landscape is structured by a domain model, which is characterized by a three-stage hierarchy: the domains are composed of subdomains. The subdomains are furthermore subdivided into functional units. Each application in the IT landscape is unambiguously assigned to exactly one functional unit. Figure 4 illustratively shows the complexity measures of the bank's IT architecture and highlights how the proposed design principles are reflected in the implementation. The caption of the rows on the left hand side of Figure 4 shows the domain structure on top level. As an example the domain 'Data Warehouse' is extended, depicting the subdivision in – in this case – one subdomain and three functional units. The values in the columns 'quantity', 'diversity' and 'interdependency' represent the calculated complexity measures.

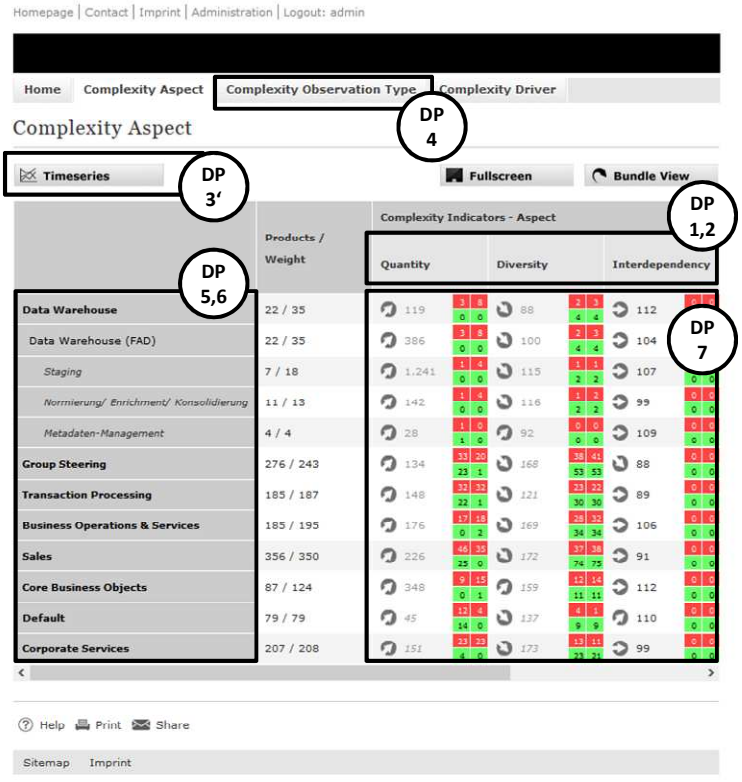

Figure 4. Design Principles located in the IT-COM Screen

These implemented complexity measures always consider the dimensions complexity aspect (quantity, diversity, and interdependency) and complexity observation type (considering the architectural layer – function, data, and technology). These terms are used in the internal language of the bank, but it should be noted that the complexity aspects correspond to the initial design principles DP 1 and DP 2 (number and heterogeneity of components and relations), and the complexity observation type corresponds to DP 4 (IT architecture domains, also shown in Figure 1).

During the entire course of the project, the applied conceptualization of complexity in IT architectures – manifested in DP 1 to DP 4 – formed the basis for the further development of the IT artifact. Whilst DP 1, DP 2 and DP 4 proved to be adequate, feedback provided by the IT architects through a workshop during BIEC4 prompted us to revise design principle DP 3 'Rate of change'. Building on the idea of Schneberger and McLean (2003) to consider the rate of change, we had discussions with the IT architects that led to an idea for an extension. 'Rate of change' addressed a certain amount of change of the IT architecture in a given period of time. It can therefore be interpreted as a 'slope' of a curve. In addition to this, the IT architects stressed the importance of also considering the development of the architecture over time. In order to be able to monitor the effects of IT architecture decisions on the complexity of the IT architecture, they needed information on how complex an IT architecture was at certain points of time:

"We should focus on a 'historical analysis': where do we come from, where do we stand, and where do we want to go. [...] We want to measure complexity on a quarterly basis, thus we have to compare the measurements of the quarters. [...] A presentation applying a timeline allows us to depict the overall development of the landscape or of a subset of the landscape." (Principal Architect)

This 'history' of complexity data can be interpreted as a 'shape' (as opposed to a 'slope') of a curve. In the context of the evaluation during the BIE stage (BIEC4) we therefore decided to set the focus on the *traceability of complexity over time in the form of time series* and rephrased the design principle DP 3 accordingly (cf. DP 3' in Table 4 and Table 5 respectively). Figure 5 illustrates how the revised DP 3' was realized in the artifact IT-COM. This figure shows the change in complexity of the application's interfaces for four domains (an exemplary selection).

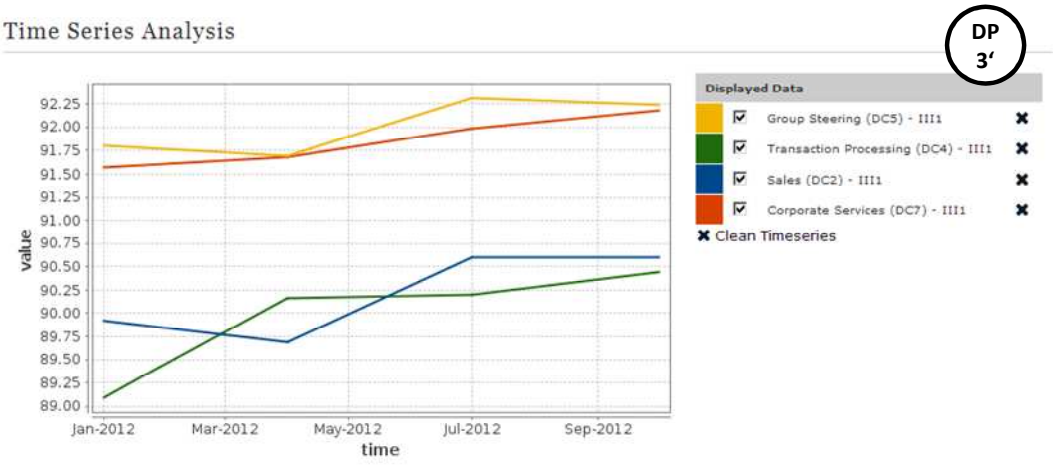

Figure 5. Time Series Analysis

From the very beginning of the development phase (BIEC2), the implementation team struggled with the import of the measurement data from different systems (e.g. SAP or the ARIS Repository), in various – and sometimes also changing – data formats. Another important and related aspect was data quality. During the data collection in the implementation phase of the measures it turned out that data quality varied greatly from data source to data source. This became apparent during BIEC 3 when the IT architects started to test the measures in order to derive recommendations for action. Because the calculations of the complexity measures were based on this data, the ADR Team had to take the issue 'data quality' into consideration.

What emerged from our discussions with the IT architects during BIEC 3 was that calling attention to the completeness and correctness of the underlying input data is a precondition for the usefulness of a complexity management information system. Because the measurement data should serve as a basis for decision-making it is not surprising that the IT architects emphasized this aspect:

"The data for IT-COM originates from various sources and systems. And we quickly realized that the quantity and the quality of the data varied strongly. Because we want to make decisions on the basis of the complexity measures, you should have a reliable data foundation. Therefore the user should be aware of the reliability of the underlying data." (Principal Architect)

This prompted us to introduce the design principle *'Data Quality'* (DP 7): "IT-COM signalizes and calls attention to the underlying data quality, the particular complexity measures are based on". Therefore, we proposed that IT-COM should inform the end user regarding the level of the respective data quality: In order to make the end user aware of a partly lower degree of the underlying data quality, an optical indication for the users was integrated. Values with a 'high' data quality are presented in dark characters, whereas values with 'medium' data quality are shown lighter and these with a 'low' data quality also lighter and in italic (cf. Figure 4). This indicates which measurement data has to be handled with care. The Principle Architect summarized the feedback received from the end users in the context of an internal presentation of IT-COM during BIEC4 as follows:

"The end users, who participated in internal presentations, appreciated the visual signal concerning the underlying data quality. They mentioned that as an important indication." (Principal Architect)

In the context of a project evaluation meeting during BIEC4, a novel subject emerged. The discussion between the implementation team and the IT architects moved from the topic of "how to realize access to IT-COM" to "who is planned to have access to IT-COM". The insights gained in this discussion led us to introduce the design principle called *'Openness'* (DP 5): Prior to this collaborative project, access to information systems providing architectural information was limited to the IT architects. A project manager for example, responsible for a project including architectural design decisions, did not have access to information systems to extract architectural data concerning the respective project-related applications. During the implementation of IT-COM we learned that the target group of IT-COM is not only composed of the IT architects. To reach an efficient IT complexity management, access to IT-COM should not be limited to this specific group of employees. Rather, all end users responsible for IT tasks related to architectural decisions needed access to IT-COM, as illustrated by the following quote during BIEC 4 of a leading IT architect of the bank:

"For us, IT-COM is an 'IT for IT' information system which should be available for all IT units. […] We do not want to exclude a certain target group – the measures should be accessible to all members of staff concerned." (Principal Architect)

Initially, the ADR Team made the decision to implement IT-COM as a web-based information system, because the technology was easy to integrate into the existing architecture of the bank. Implementing IT-COM as a web-based information system was thought to be an advantage, because this facilitated the accessibility of IT-COM to all employees on the intranet. The end users emphasized the availability of architectural data and the corresponding measurement data of their respective organizational context as a significant improvement. The Principle Architect summarized the feedback received from the head of payments during an internal presentation of IT-COM as follows:

"For some time he already promoted to increase the adherence to standards […] but such initiatives were hard to enforce because of the budgetary situation. And now I could show him, that the adherence to standards is even getting worse the last years. […] He told me that this may serve as a good argument to get budget for this task." (Principal Architect)

Because every change request with an impact on the IT architecture requires an 'IT architecture check', decision-makers were often confronted with the problem to collect the required information from different sources. Furthermore, the impact of the change was hard to assess or just evaluated on the basis of 'gut feeling'. This functionality of the implemented IT complexity management information system IT-COM also strengthened the position of the decision-makers.

In the course of our discussions regarding DP 5 another design principle emerged during the BIE-Stage (BIEC4), which is related to DP 5: *'Level of Abstraction and Drill-Down'* (DP 6): "IT-COM enables the

end user to freely chose the level of abstraction (domain-, subdomain- or functional unit level)". As mentioned above, the target group of IT-COM was extended significantly. Thereby, the requirements regarding the facilities for analysis were also extended. Because end users with various areas of responsibility and different perspectives on the IT architecture were addressed, the focus on the flexibility of analysis increased. For example, an application owner is interested in the measures concerning the application he or she is responsible for (e.g., the complexity of the interfaces of an application). A project manager has a broader perspective, not focusing on only one application but considering a specific subset of applications. However, a CIO, as a rule, does not need specific information of subsections of an IT architecture, but of the IT architecture as a whole or at least at a higher level of abstraction.

Another reason that caused the introduction of DP 6 was that end users needed the possibility to identify the source of complexity:

"The aggregated measures on top level can be understood as an indicator. At this level a 'recommendation for action' would be 'I have to do something'. But to find out what this 'something' is, you have to drill-down. If you have a domain including 50 applications, at a first glace – at the aggregated level – you cannot say what causes the complexity. Thus you have to drill-down. On this level you can identify the cause of complexity." (Principal Architect)

It could for example be possible that a top-level structure shows a high degree of complexity, but a detailed analysis reveals that the corresponding sub-structures itself are homogeneous and therefore less complex. Furthermore, once end users identified a need for action, they required a possibility to evaluate the effect of specific actions, which were not related to the IT architecture as a whole, but to a section of the IT architecture. This also emphasized the need to take into account different levels of abstraction.

The following Table 4 summarizes the instantiated design principles for "IT architecture complexity management in the environment of the bank". These instantiated design principles and the corresponding implementation product depict the material artifacts (Gregor and Jones 2007) and therefore a Level 1 contribution (Gregor and Hevner 2013) resulting from the ADR project.

Table 4. Instantiated Design Principles

Design Principle	**Instantiation of the DP at the bank**
DP 1: Number and heterogeneity of components	The implemented information system IT-COM considers the number of database objects, use cases (number of business functionality an application supports), occurrences in processes, users, user departments, functional implementations and infrastructure requirements as well as the heterogeneity of technologies of the bank.
DP 2: Number and heterogeneity of relations	IT-COM considers the number of interfaces and heterogeneity of implementation types of the architecture's relations as well as the interface interdependency on business and on technical level of the bank.
DP 3': Change	IT-COM enables the traceability of changes over time with the help of time series for each calculated measure.
DP 4: IT architecture domains	IT-COM considers the domains function, data, and technology as well as the interfaces between them.
DP 5: Openness	Access to IT-COM is not limited to a specific group of employees. Rather, all end users responsible for IT tasks related to architectural decisions need access to IT-COM.
DP 6: Level of Abstraction and Drill-Down	IT-COM enables the end user to freely choose the level of abstraction (domain-, subdomain- or functional unit level).
DP 7: Data Quality	IT-COM signalizes and calls attention to the underlying data quality, the particular complexity measures are based on.

Discussion of Results

Theoretical implications

The implemented artifact (IT-COM) – built during the ADR project – depicts a solution to the specific problem ("IT architecture complexity management in the environment of the bank"). Since IT-COM – as a piece of software – is a materially existent artifact, implemented for a specific context, it represents a Level 1 contribution (Gregor and Hevner 2013). The same applies to the instantiated design principles (shown in Table 4) which form the basis for abstracted design principles as a nascent design theory and therefore the Level 2 contribution (Gregor and Hevner 2013) of our ADR project (cf. Figure 6).

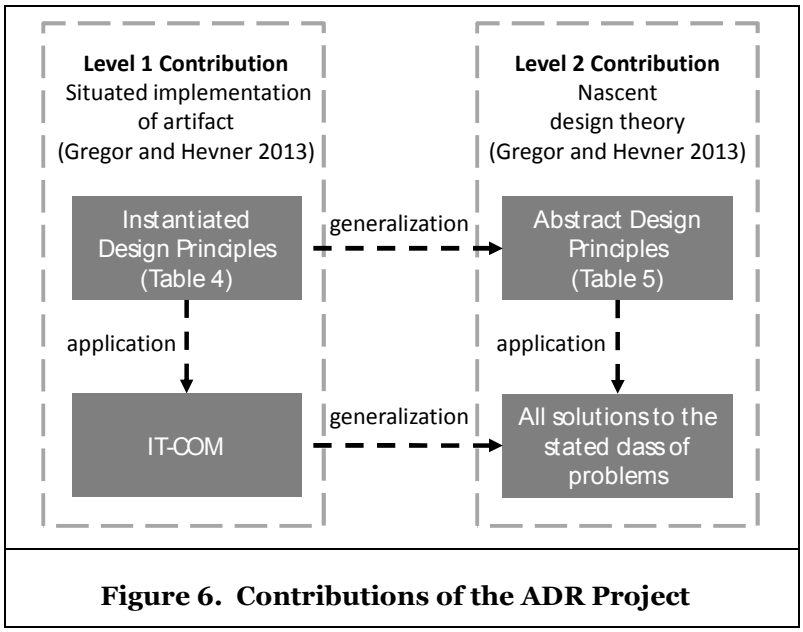

Figure 6. Contributions of the ADR Project

IT-COM is a representation of the set of solutions to the class of problems ("IT architecture complexity management"). We generalized the solution by deriving abstract design principles applicable to each representation of this class of problems. These generalized design principles are summarized in Table 5.

From a building design theory perspective, Gregor and Jones (2007) differentiate between 'Instantiations (material artifacts)', represented through an instantiated product or method, and '(Nascent) Theories (abstract artifacts)', such as design theories or principles. Adopting this lens, the implementation artifact as well as the instantiated design principles (see Table 4) can be thought of as material, instantiated artifacts. Because the abstract design principles (see Table 5) are not materially existent in the real world they depict an abstract artifact or nascent theory (Gregor and Jones 2007).

Next to the proposed abstracted design principles and the developed artifact IT-COM, which represents a solution to the stated problem of the bank, we validated the introduced conceptualization of complexity in IT architectures proposed by Schütz et al. (2013) and complemented by the "rate of change" proposed by Schneberger and McLean (2003) (cf. Figure 3). Through the continuous evaluation during the BIE-Cycles we validated the artifact IT-COM, the design principles, and, at the same time the applied conceptualization. The resulting design principles DP 1, 2 and 3' – quoting what should be considered when managing complexity of an IT architecture – highlight the comprehensiveness of the system theoretical perspective on IT architecture and DP 4 elucidates the applicability of this perspective to the mentioned subsystems of an IT architecture. Thereby, the presented research indicates the appropriateness of the introduced conceptualization (cf. subsection 'Theory-Ingrained Initial Design Principles' in section 'Results') of complexity in IT architectures for the stated class of problems.

Table 5. Derived Abstract Design Principles from the Design Research Outcomes			
Design Principle	**Description/Statement: To master complexity…**	**Rationale**	**Source**
DP 1: Number and heterogeneity of components	… the number and heterogeneity of the system's components should be considered.	The number and heterogeneity of components are key elements of the state-of-the-art conceptualization of complexity in IS.	Literature (cf. Table 3); confirmed in BIE-cycles
DP 2: Number and heterogeneity of relations	… the number and heterogeneity of the system's relations should be considered.	The number and heterogeneity of relations are key elements of the state-of-the-art conceptualization of complexity in IS.	Literature (cf. Table 3); confirmed in BIE-cycles
DP 3': Change	… it is needed to measure the changes of an IT architecture. Therefore, changes over time should be considered. (Availability of time series enable traceability of change)	Changes to the IT architecture are key elements of the state-of-the-art conceptualization of complexity in IS.	Literature (cf. Table 3); revised in BIE-cycles (BIEC4)
DP 4: IT architecture domains	… the domains 'data/information architecture', 'application architecture' and 'infrastructure architecture' should be considered.	State-of-the-art EAM knowledge suggests a differentiated view of layers for key design decisions.	Literature (cf. Table 3); confirmed in BIE-cycles
DP 5: Openness	… access to IT architectural data and the complexity measures should not be limited to a specific group of employees. Rather, all end users responsible for IT tasks related to architectural decisions should have access to the measurement data.	Information that is accessible to all involved stakeholders increases the quality of complexity management.	Emerged in BIE-cycles (BIEC4)
DP 6: Level of Abstraction and Drill-Down	… the IT architecture should be analyzed from different levels of abstraction.	Information that is provided on different levels of abstraction allows to identify sources of complexity and to derive practical needs for action.	Emerged in BIE-cycles (BIEC4)
DP 7: Data Quality	… the underlying data quality of the particular complexity measures should be signalized to the users.	Awareness of the accuracy of information increases the quality of decisions.	Emerged in BIE-cycles (BIEC3)

Furthermore, the emergence of design principles DP 5 and DP 6 emphasizes that complexity management in IT architectures is not only relevant to IT architects, but also to all end users responsible for IT tasks related to architectural decisions. IS literature proposes theoretical profiles of the IT function in organizations. Guillemette and Paré (2012) identify the 'Architecture Builder' as the function "to build and manage an IT infrastructure that supports business processes and reduces architectural complexity" (Guillemette and Paré 2012, p. 532). Cross et al. (1997) assign the IT function to reduce architectural complexity to the 'Infrastructure Planner'. In addition to those perspectives, we consider the management of architectural complexity as a task affecting a large group of stakeholders. Therefore, even

more functions than only the IT architects should be involved in the corresponding management processes. This is also reflected by the emerged design principles (cf. DP 5 and DP 6).

Practical implications

The developed artifact IT-COM represents (1) a solution to the stated problem of the bank and (2) an instantiation of the applied conceptualization of complexity in IT architectures. Thereby, the derived abstract design principles further represent the answer to the posed research question, which design principles should guide the design of an information system that supports IT architects to manage IT architecture complexity. The implementation of an information system for 'IT architecture complexity management' based on the proposed design principles allows practitioners to cope with the problem of a companywide management of IT complexity.

Detailed practical implications can be derived particularly from the observed consequences of introducing IT-COM into the organizational context. The following Table 6 summarizes our intervention of introducing the beta-version of IT-COM to a selective set of end users (IT architects, IT project manager, employees of the internal application development and decision-makers in the bank). The IT architects distributed IT-COM among the mentioned end users and received the presented feedback.

Table 6. Consequences of BIE of IT COM on the Environment

Design Principle	Consequences
DP 1-4	The applied conceptualization of complexity in IT architectures (DP 1-4) influenced the perspective of the IT architecture management team.
DP 5: Openness	End users, especially decision-makers and project team members, have access to the current state of the IT architecture and are able to simulate the impacts of a change on the whole or a particular section of the IT architecture. This increase of documentation and transparency has the potential to increase the quality of IT architecture decisions in the middle and long run.
DP 6: Level of Abstraction and Drill-Down	End users are facilitated to analyze the IT architecture from different levels of abstraction to evaluate the impact of a change on a small subsystem (such as their functional unit) and on a huge system (such as the whole company). This possibility enables IT architects to discuss whether a 'local optimum' (on subsystem level) has preference over a 'global optimum' (e.g. on system level).
DP 7: Data Quality	The introduction of an IT complexity management information system increases the need for the availability and accuracy of the data, required for the measurement. Inaccurate data carries the risk of a less solid base for decision-making, may leading to (from an IT architectural point of view) suboptimal or even wrong decisions.

Limitations and Future Research

Some limitations of the presented study should also be noted. Since the ADR approach is oftentimes characterized by a particular close relationship with a single company, the research benefits from an in-depth analysis in the organizational context. However, beyond this study that entails a single research site, it would be relevant to transfer the design principles in other contexts.

As stated in section 'Theoretical Background', we focused on IT architecture complexity. Therefore, we derived design principles that guide the design of an information system that supports IT architects to manage IT architecture complexity. Further research might explore to what extend the identified design principles are also applicable to EA complexity management as a whole.

In IS literature, there exist further design principles in the context of EA (e.g. Greefhorst and Proper (2011), Haki and Legner (2013), Richardson et al. (1990)). A further study investigating how the proposed design principles in this paper relate to existing EA design principles and how these can be integrated would be very interesting.

Methodically, Scrum turned out to be a suitable software development Framework for ADR. The agile concept and especially the sprints – as a basic unit of development – facilitated to carry out our research applying the ADR approach. Further research should examine and compare other software development frameworks in combination with the ADR approach. Furthermore building disjunctive groups in the ADR team also turned out as a fruitful decision, because this allowed for a focused evaluation and analysis during the BIE-cycles. A direction for further research would be to examine the effects of a non-disjunctive setup of the groups in the ADR team.

Conclusion

Our study highlights the holistic, system theoretical, and dynamic nature of the complexity perspective on IT architectures. This perspective complements and extends the existing perspectives in the literature, namely the planning perspective (Schmidt and Buxmann 2011), the strategy perspective (Ross 2003), and the standardization perspective (Boh and Yellin 2006). With the complexity perspective in addition to the other perspectives, we add our understanding about how to manage IT architectures, by focusing on the surprisingly neglected aspects of the complexity of a system, defined as the number and the heterogeneity of components and relations. In this paper we propose a different perspective on IT architectures, which may trigger new ways of thinking about this management phenomena and encourage future research. Of course IT architecture management is about strategy, standardization and planning, but we say it is especially also about complexity management.

Acknowledgements

First of all, the authors thank the anonymous reviewers for their valuable suggestions. Furthermore, we would like to especially thank the IT architects who participated in the ADR project for their valuable and constructive comments during the entire collaboration. We also thank the members of the implementation team for the great cooperation throughout the project.

References

Awerbuch, B., and Schulman, L.J. 1997. "The Maintenance of Common Data in a Distributed System," Journal of the ACM (44:1), pp. 86-103.

Benedikt, M., Libkin, L., Schwentick, T., and Segoufin, L. 2003. "Definable Relations and First-Order Query Languages over Strings," Journal of the ACM (50:5), pp. 694-751.

Berlinger, E. 1980. "An Information Theory Based Complexity Measure," in: Proceedings of the National Computer Conference. ACM, pp. 773-779.

Boh, W.F., and Yellin, D. 2006. "Using Enterprise Architecture Standards in Managing Information Technology," Journal of Management Information Systems (23:3), Winter2006/2007, pp. 163-207.

Briggs, R.O., Nunamaker Jr, J.F., and Sprague, R. 2011. "Applied Science Research in Information Systems: The Last Research Mile," Journal of Management Information Systems (28:1), Summer2011, pp. 13-16.

Cross, J., Earl, M.J., and Sampler, J.L. 1997. "Transformation of the IT Function at British Petroleum," MIS Quarterly (21:4), pp. 401-423.

Davis, J.S., and LeBlanc, R.J. 1988. "A Study of the Applicability of Complexity Measures," IEEE Transactions on Software Engineering (14:9), pp. 1366-1372.

Fischer, C., Winter, R., and Aier, S. 2010. "What Is an Enterprise Architecture Design Principle? Towards a Consolidated Definition," in Computer and Information Science 2010. Berlin, Heidelberg: Springer, pp. 193-205.

Francalanci, C., and Piuri, V. 1999. "Designing Information Technology Architectures: A Cost-Oriented Methodology," Journal of Information Technology (14:2), pp. 181-192.

Greefhorst, D., and Proper, E. 2011. Architecture Principles - the Cornerstones of Enterprise Architecture. Heidelberg: Springer.

Gregor, S., and Hevner, A.R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," MIS Quarterly (37:2), pp. 337-355.

Gregor, S., and Jones, D. 2007. "The Anatomy of a Design Theory," Journal of the Association for Information Systems (8:5), pp. 313-335.

Guillemette, M.G., and Paré, G. 2012. "Toward a New Theory of the Contribution of the IT Function in Organizations," MIS Quarterly (36:2), pp. 529-551.
Haki, M.K., and Legner, C. 2013. "Enterprise Architecture Principles in Research and Practice: Insights from an Exploratory Analysis," in: European Conference on Information Systems. Utrecht, Netherlands.
Harrison, W. 1992. "An Entropy-Based Measure of Software Complexity," IEEE Transactions on Software Engineering (18:11), pp. 1025-1029.
Henningsson, S., and Hanseth, O. 2011. "The Essential Dynamics of Information Infrastructures," in: International Conference on Information Systems. Shanghai, China.
Hevner, A.R., March, S.T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," MIS Quarterly (28:1), pp. 75-105.
IEEE Architecture Working Group. 2000. "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," IEEE Std 1471-2000), pp. i-23.
Jiang, Z.J., and Benbasat, I. 2007. "The Effects of Presentation Formats and Task Complexity on Online Consumers' Product Understanding," MIS Quarterly (31:3), pp. 475-500.
Kim, R.M., and Kaplan, S.M. 2006. "The Co-Evolutionary Dynamics of IS Engagement," in: European Conference on Information Systems. Göteborg, Sweden.
Klir, G.J. 2001. Facets of Systems Science. Springer.
Lindgren, R., Henfridsson, O., and Schultze, U. 2004. "Design Principles for Competence Management Systems: A Synthesis of an Action Research Study," MIS Quarterly (28:3), pp. 435-472.
March, S.T., and Smith, G.F. 1995. "Design and Natural Science Research on Information Technology," Decision Support Systems (15:4), pp. 251-266.
Markus, M.L., Majchrzak, A., and Gasser, L. 2002. "A Design Theory for Systems That Support Emergent Knowledge Processes," MIS Quarterly (26:3), pp. 179-212.
Mattern, F., Schönwälder, S., and Stein, W. 2003. "Fighting Complexity in IT," McKinsey Quarterly (1), pp. 57-65.
Morel, B., and Ramanujam, R. 1999. "Through the Looking Glass of Complexity: The Dynamics of Organizations as Adaptive and Evolving Systems," Organization Science), pp. 278-293.
Nan, N. 2011. "Capturing Bottom-up Information Technology Use Processes: A Complex Adaptive Systems Model," MIS Quarterly (35:2), pp. 505-532.
Orlikowski, W.J., and 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), pp. 121-134.
Porter, M.E., and Millar, V.E. 1985. "How Information Gives You Competitive Advantage." Harvard Business Review, Reprint Service.
Richardson, G.L., Jackson, B.M., and Dickson, G.W. 1990. "A Principles-Based Enterprise Architecture: Lessons from Texaco and Star Enterprise," MIS Quarterly (14:4), pp. 385-403.
Roberts, T.L., Cheney, P.H., Sweeney, P.D., and Hightower, R.T. 2004. "The Effects of Information Technology Project Complexity on Group Interaction," Journal of Management Information Systems (21:3), pp. 223-247.
Ross, J.W. 2003. "Creating a Strategic IT Architecture Competency: Learning in Stages," MIS Quarterly Executive (2:1), pp. 31-43.
Ross, J.W., Weill, P., and Robertson, D. 2006. Enterprise Architecture as Strategy: Creating a Foundation for Business Execution. Harvard Business Press.
Schmidt, C., and 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), pp. 168-185.
Schneberger, S.L. 1997. "Distributed Computing Environments: Effects on Software Maintenance Difficulty," Journal of Systems & Software (37:2), p. 101.
Schneberger, S.L., and McLean, E.R. 2003. "The Complexity Cross: Implications for Practice," Communications of the ACM (46:9), pp. 216-225.
Schütz, A., Widjaja, T., and Kaiser, J. 2013. "Complexity in Enterprise Architectures – Conceptualization and Introduction of a Measure from a System Theoretic Perspective," in: European Conference on Information Systems. Utrecht, Netherlands.
Schwaber, K., and Beedle, M. 2002. Agile Software Development with Scrum. Prentice Hall Upper Saddle River.
Sein, M., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. 2011. "Action Design Research," MIS Quarterly (35:1), pp. 37-56.

Simon, H.A. 1962. "The Architecture of Complexity," Proceedings of the American philosophical society (106:6), pp. 467-482.

Spewak, S., and Hill, S.C. 1993. Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology. New York: John Wiley & Sons

The Open Group. 2011. "TOGAF (the Open Group Architecture Framework) V. 9.1." Retrieved 30.04.2013, from http://pubs.opengroup.org/architecture/togaf9-doc/arch/

Van de Ven, A.H. 2007. Engaged Scholarship: A Guide for Organizational and Social Research. New York: Oxford University Press Inc.

Vardi, M.Y. 1982. "The Complexity of Relational Query Languages," in: Proceedings of the 14th annual ACM symposium on Theory of computing. ACM, pp. 137-146.

Widjaja, T., and Gregory, R.W. 2012. "Design Principles for Heterogeneity Decisions in Enterprise Architecture Management," in: International Conference on Information Systems. Orlando, USA.

Widjaja, T., Kaiser, J., Tepel, D., and Buxmann, P. 2012. "Heterogeneity in IT Landscapes and Monopoly Power of Firms: A Model to Quantify Heterogeneity," in: International Conference on Information Systems. Orlando, USA.

Winchester Mystery House, L. 2013. "Winchester Mystery House." Retrieved 02.04.2013, from http://www.winchestermysteryhouse.com/thehouse.cfm

Winter, R., and Aier, S. 2011. "How Are Enterprise Architecture Design Principles Used?," in: IEEE International EDOC Conference Workshops, Trends in Enterprise Architecture Research (TEAR). Helsinki, Finland: IEEE, pp. 314-321.

Xia, W., and Lee, G. 2005. "Complexity of Information Systems Development Projects: Conceptualization and Measurement Development," Journal of Management Information Systems (22:1), pp. 45-83.