Functional Service Domain Architecture Management: Building the Foundation for Situational Method Engineering

Daniel Stock, Robert Winter, Jörg H. Mayer

15 FUNCTIONAL SERVICE DOMAIN ARCHITECTURE MANAGEMENT: Building the Foundation for Situational Method Engineering

Daniel Stock
Robert Winter
Jörg H. Mayer
Institute of Information Management
University of St. Gallen
St. Gallen, Switzerland

Abstract *Functional service domains are logical design artifacts that are intended to achieve better business/IT alignment. Their widespread utilization clearly indicates their perceived usefulness in managing the complexity of aligning business structures with IT structures. However, a common understanding of functional service domains and the associated principles that govern their design and evolution is still missing. So far, the literature provides only little guidance in closing this gap. This article contributes to the foundations that allow for the design of a situational method for functional service domain architecture management. Reviewing current literature, a framework is proposed that supports the identification of functional service domain architecture management patterns. Based on a better understanding of functional domain architecture management approaches, situational method engineering for functional domains can be applied by identifying context types and goal vectors, designing fragments, and associating successfully adopted method fragments with specific situations. The validity of the proposed framework is tested by five case studies.*

Keywords Functional service domain, enterprise architecture management, situational method engineering

1 INTRODUCTION

1.1 Motivation

In large enterprises, the complexity of the information system landscape has grown constantly. This does not only concern the number of information systems, but also their interdependencies and connecting information flows. Functional service domains (for the sake of simplicity herein after referred to as *domains*) are clusters of linkages between business structures and IT structures on a maximum level of aggregation. Domains are a widespread concept that is intended to reduce the complexity of modeling and managing the information system landscape. Despite their perceived importance by practitioners, domains are rarely actively managed in enterprises to unfold their full potential. The literature provides only little guidance in closing this gap, discussing only very specific aspects of domain architecture such as domain modeling (Kurpjuweit 2009) or decoupling of domains (Schlamann 2004). A comprehensive approach to domain architecture management that allows for situational (e.g., context and/or goal specific) adaptations is missing. This article contributes to a situational design of domain architecture management by proposing a model that allows identifying patterns in domain architecture. The model is based on an analysis of five practice cases.

1.2 Functional Service Domain Architecture

While the definitions of *domain* vary by context, most authors agree that a domain represents a view on the information system landscape that is characterized by a high congruence from a business (not technical) point of view (Aier 2007; Engels et al. 2008; Schlamann 2004; Schwinn 2006). In order to differentiate domains from applications, Schelp and Winter (2008) point out that the main difference is the level of aggregation, which is in line with several authors that use the term *sub-domains* to specify a different level of aggregation (Dodd 2005; Engels et al. 2008; Richter et al. 2005). In addition it should be noted that domain is not a fully established term and that many synonyms are in use. Domains are, for example, designated as building blocks (Jung 2004), application clusters (Lankes et al. 2005), or service segments (FEA PMO 2007; Open Group 2009).

Architecture in general is defined by ANSI/IEEE 1471-2000 as "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" (IEEE 2000). Therefore, architecture serves a specific purpose (IEEE 2000; Lankhorst 2005; Rohloff 2008). In reference to this understanding, domain architecture can be defined as the fundamental organization of an enterprise's information system landscape, embodied in domains, the relationship between domains and to other enterprise architecture artifacts, as well as the principles governing their design and evolution.

This definition comprises two core elements that are subsumed under the term architecture: an aggregate model that constitutes the relations of a complex system, and guidelines for the design and evolution of the modeled system (Sinz 1999). According to this definition, domain architecture has method character since it consists of activity specifications (domain architecture specification guidelines/activities) and respective

result specifications (domain architecture model) (Winter et al. 2009). The application of domain architecture to achieve a specific purpose (in this case, business/IT alignment) is designated as domain architecture management.

1.3 Objectives

Domains are predominantly discussed in two contexts: enterprise architecture (EA) in general (e.g., FEA PMO 2007; Open Group 2009), and enterprise application integration/service-oriented architecture (EAI/SOA) in particular (e.g., Engels et al. 2008; Heutschi 2007; Josuttis 2008; Schlamann 2004).

In EA, domains are very generically understood as an artifact type that aligns business and IT structures, but without further guidelines for identification, specification, and evolution of domains. In EAI/SOA, the scope of domain architecture is limited to the decoupling of application clusters through services. This narrow, mostly IT-oriented focus tends to disregard further application scenarios of domain architecture such as the increase of transparency or flexibility from a business perspective. In summary, a comprehensive proposal for domain architecture management that can be tailored to specific goals is missing so far.

Situational method engineering (Harmsen 1997; Kumar and Welke 1992; van Slooten and Hodes 1996) aims at constructing methods that can be adapted to different design problem classes (situations). In general, two modification techniques can be differentiated: configuration and aggregation (vom Brocke 2003). The configuration technique follows the so-called adaptive principle: subsequent changes are explicitly allowed for and planned at the moment of the initial construction of the artifact. On the other hand, the aggregation technique follows the compositional principle, permitting subsequent changeability that is, at least to a certain degree, almost unrestricted.

Both techniques require the identification of situation characteristics that a certain base-method is tailored to (in the configuration case) or that serve as the basis for the combination and aggregation of method fragments (in the aggregation case) (Bucher et al. 2007). In order to specify situation characteristics, Bucher et al. differentiate between so-called project-type and context-type characteristics. Project-type characteristics are factors that influence the project and are under the control of the project (e.g., goals), while context-type characteristics are factors that influence the project but are beyond its control (e.g., company size, industry specifics). In order to apply situational method engineering for constructing domain architecture management artifacts, the different goals (or goal vectors) and their implications for the results and activities of domain architecture management need to be understood. Therefore, this article proposes a model that allows identifying patterns of domain architecture management in practice, which is needed for constructing a method that takes situational characteristics into account.

1.4 Research Methodology

Information systems research is mainly characterized by two paradigms: behavioral research and design research. While behavioral research concentrates on the develop-

ment and verification of explanatory, descriptive theories, design research focuses on the development of innovative, generic solutions for practical problems and, thereby, on accomplishing utility (Hevner et al. 2004; March and Smith 1995). According to Hevner et al. and March and Smith, the outcomes of a construction process under the design research paradigm can be classified as constructs, models, methods, and instantiations. The goal of this article is to contribute to the design of a situational method for domain architecture management by providing a morphological model (designated in the following as *morphology*) which can be used to identify and document patterns in practice as a starting base.

In order to develop the artifacts mentioned earlier, several reference models for the construction process have been proposed (Hevner et al. 2004; March and Smith 1995; Peffers et al. 2006; Rossi and Sein 2003). The process of March and Smith that specifies *build* and *evaluate* activities is predominant in literature (Hevner et al. 2004). This article focuses on the build part of the morphology through a review of current literature and uses a two-stage explorative validation with five practice cases. In step one, four cases are used for an indicative assessment of the differentiating potential of the proposed morphology. In step two, the artifact is evaluated with regard to completeness, clearness, and relevance through an in-depth analysis of one further practice case. In contrast to the four cases used for assessing the differentiating potential of the proposed morphology that has already been documented in the literature, the evaluation case is described here for the first time.

It should be noted that the explorative evaluation presented here is part of an iterative design approach, which seems the most promising in identifying and justifying adaptations to the morphology. Therefore, further evaluation of the artifact as well as the subsequent design of a situational method for domain architecture management is subject to further research.

This article is structured as follows. Section 2 derives a morphology for domain architecture in order to differentiate practice approaches along certain constituent dimensions. In section 3, the proposed morphology is used to analyze four practice cases from the literature in order to provide an indicative validation of its differentiating potential. Section 4 introduces a new case in detail in order to provide a first validation of the morphology's information value. The article closes with a discussion of the results and a proposal for further research in section 5.

2 DERIVATION OF DIMENSIONS FOR FUNCTIONAL SERVICE DOMAIN ARCHITECTURE SPECIFICATION

In order to present a compact overview of different approaches to domain architecture, this section derives a morphology from current literature as a basic structure to present practice cases. In a first step the dimensions of the morphology are identified by decomposing domain architecture into its components and assigning respective degrees of freedom. In a second step each dimension is detailed through potential values that an instantiation might realize. The descriptive value of the resulting morphology is then evaluated against a first explorative set of cases in the next section.

As discussed earlier, domain architecture in general constitutes results and activities that are tailored to a specific target state (goal orientation). This definition of domain architecture comprises three high-level components (*target, results,* and *activities*) that are subject to further detailing:

- Derived from general EA goals, the target of domain architecture can be specified by its *application scenarios* (Winter et al. 2007) and its *stakeholders* (Niemi 2007; Op't Land et al. 2009; Ylimäki 2006).
- The result of domain architecture is the domain model, which is specified by the *domain definition/separation* and the included artifacts and relationships (Aier et al. 2009). The selection of the latter is defined by the *viewpoints* that are needed to satisfy the concerns of the respective stakeholders.
- According to the constituent elements of a method (Gutzwiller 1994; Heym 1993), the activities category can be decomposed into *design and evolution principles* (what is done), *implementation approach* (how is it done), and *organization* (who does it).

This results in seven dimensions that fall into the three above categories. For each of these dimensions, potential values need to be defined:

- According to the literature review of Heutschi (2007), domain architecture is used in three application scenarios: (1) *Increasing transparency* (for business/IT alignment), for example, by providing a map/inventory of available enterprise services within each domain. (2) *Reducing complexity/interdependencies* (Schlamann 2004) by decoupling interdomain linkages, for example, using some bus technology. (3) *Decentralization of responsibilities* (see FEA PMO 2007; Josuttis 2008; Open Group 2009), for example, by allocating domain overarching and domain specific responsibilities.
- According to the literature review of Winter and Fischer (2006), domains are elements of the architectural alignment layer that is positioned between the business and IT layers of enterprise architecture. Therefore, the generic stakeholders of a domain model can be specified as *business*, *IT*, or *business and IT*.
- Domain definition/separation is structured along *business processes* (of business units or product lines), *business entities*, or *business dimensions* (e.g., channels, products, customer segments) (see Cherbakov et al. 2005; Engels et al. 2008; Pohland 2000). It should be noted that pure forms of these concepts can be rarely found in practice, even if they tend to demonstrate a dominant approach.
- Kurpjuweit (2009) identifies two basic viewpoints: *inventory* and *landscape*. The former presents a list of applications within a domain, while the latter specifies the information flows between applications. These two viewpoints may have varying levels of detail. On the one hand, the inventory viewpoint can either constitute a simple listing (*application inventory*), or on top specify the functionality of and business entities maintained by the respective applications (*functions inventory*). On the other hand, the landscape viewpoint might specify interdomain information flows only (*domain landscape*) or both inter- and intradomain information flows (*application landscape*).
- Design and evolution principles operationalize the strategic targets (application scenarios): *consistency* in modeling artifacts and relationships to increase trans-

parency, *reuse* to increase business/IT alignment, and *loose coupling* (through services) for reducing complexity and installing autonomy (see Heutschi 2007).

- Hafner and Winter (2008) present several distinct approaches to implement the design and evolution principles that range from a purely passive to a purely active mandate: *architecture communication*, *architecture lobbying*, and *architecture enforcement*. Within an architecture communication approach, the organizational reach-through is limited to the publication of information material. In contrast, an architecture enforcement approach implies a respective governance structure in order to actively push architecture guidelines into the organization. Architecture lobbying constitutes a compromise of these two, where the architecture unit is consulted to promote their design and evolution principles, but the right of decision remains out of their scope.
- Niemann (2006) identifies four different models to organize architecture management according to whether strategic and/or operational architecture management are implemented in a centralized or decentralized form: *centralized* architecture management (strategic and operational architecture management within one central functional unit), *diversified* organization (strategic and operational architecture separated in two central functional units), *distributed* architecture management (operational architecture management decentralized), and *decentralized* architecture management (strategic and operational architecture management decentralized).

According to the remarks above, each dimension is associated with three to four potential values. It should be noted that, in some cases, the values are not mutually exclusive. This is due to the fact that some dimensions imply a kind of maturity logic. For example, it can be assumed that architecture communication is a prerequisite for architecture lobbying and in the same course a prerequisite to architecture enforcement. Similar argumentation holds true for the dimensions application scenario, viewpoints, and design and evolution principles. In these cases, the different values are lined up from left to right in order of increasing maturity. This results in the morphology that is illustrated by Table 1.

Table 1 Proposed Morphology for Functional Service Domains

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	Business	IT	Business and IT	
Results	Domain definition	Business processes	Business entities	Business dimensions	
	Viewpoints	Application inventory	Functions inventory	Domain landscape	Application landscape
Activities	Design and evolution	Consistency	Reuse	Loose coupling	
	Implementation approach	Architecture communication	Architecture lobbying	Architecture enforcement	
	Organization	Centralized	Diversified	Distributed	Decentralized

3 ASSESSMENT OF THE DIFFERENTIATING POTENTIAL OF THE PROPOSED MORPHOLOGY

This section presents four practice cases for domain architecture: Credit Suisse (Hafner and Winter 2008; Hagen 2003; Schlamann 2004), Swisscom IT Services AG (Schwinn 2006), Axpo Informatik (Schwinn 2006), and PostFinance (Dietzsch 2008). By structuring each case along the findings of the previous section, a first explorative validation of the proposed morphology and its potential to identify patterns for later construction of a situational method are provided.

3.1 Practice Cases

3.3.1 Domains in Credit Suisse Architecture Management

Credit Suisse is one of the largest banks in Switzerland and operates globally. Domain architecture at Credit Suisse[1] primarily aims at the reduction of complexity that arises from interdependencies between applications. This is addressed by grouping applications along the core business entities into domains, while information flows across domain boundaries are loosely coupled through services. A domain is detailed through its applications and each application through its public interfaces, which offer functionality and access to data across domain boundaries. The domain architecture is enforced by the central Integration Architecture Group. Within Credit Suisse, the domain architecture is primarily targeted for the use within IT. An overview of this approach to domain architecture is given by Table 2. The respective values are highlighted, while dimensions without information are marked unavailable.

3.1.2 Domains in Swisscom IT Services Architecture Management

Swisscom IT Services is not only the IT service provider of the largest Swiss telecommunications company, but also a large provider of IT services to other companies, primarily in Switzerland. Domain architecture at Swisscom IT Services[2] primarily aims at the identification of interdependencies in order to undertake integration efforts in development projects. Therefore, domains are structured across the core business processes and the information flows between domains are modeled in a consistent manner to specify interdomain interdependencies. Information about the implementation approach and organization of the dimensions is not available. An overview is provided by Table 3.

---

[1]https://www.credit-suisse.com/ch/en/index.jsp (Credit Suisse Private Banking).
[2]http://www.swisscom.com/IT/content/home.htm?lang=en.

Table 2 Credit Suisse Approach Specified Using the Proposed Methodology

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	**Reducing complexity**	Decentralization of responsibilities	
	Stakeholder	Business	**IT**	Business and IT	
Results	Domain definition	Business processes	**Business entities**	Business dimensions	
	Viewpoints	Application inventory	**Functions inventory**	Domain landscape	Application landscape
Activities	Design and evolution principles	Consistency	Reuse	**Loose coupling**	
	Implementation approach	Architecture communication	Architecture lobbying	**Architecture enforcement**	
	Organization	Centralized	Diversified	Distributed	**Decentralized**

Table 3 Swisscom IT Services Approach Specified Using the Proposed Morphology

Category	Dimension	Potential values			
Target	Application scenarios	**Increasing transparency**	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	Business	IT	**Business and IT**	
Results	Domain definition	**Business processes**	Business entities	Business dimensions	
	Viewpoints	Application inventory	Functions inventory	**Domain landscape**	Application landscape
Activities	Design and evolution principles	**Consistency**	Reuse	Loose coupling	
	Implementation approach	**Not available**			
	Organization	**Not available**			

3.1.3 Domains in Axpo Informatik Architecture Management

Axpo Informatik is the IT service provider of a large network of power utility companies in Switzerland. Domain architecture at Axpo Informatik[3] is primarily targeted at facilitating the communication between business and IT. At an aggregate level, inter-domain information flows are modeled in a consistent manner to make the necessary integration efforts during development processes quantifiable. Information about the implementation approach and organization of the dimensions is not available. An overview is provided by Table 4.

[3]http://www.axpo.ch/internet/axpo/en/ueberuns/gruppe/informatik.html.

Table 4 Axpo Informatik Approach Specified Using the Proposed Morphology

Category	Dimension	Potential values			
Target	Application scenarios	**Increasing transparency**	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	Business	IT	**Business and IT**	
Results	Domain definition	**Business processes**	Business entities	Business dimensions	
	Viewpoints	Application inventory	Functions inventory	**Domain landscape**	Application landscape
Activities	Design and evolution principles	**Consistency**	Reuse	Loose coupling	
	Implementation approach	**Not available**			
	Organization	**Not available**			

3.3.4 Domains in PostFinance Architecture Management

PostFinance is the financial services business unit of the Swiss Postal Service. PostFinance is the largest payment processor in Switzerland and is offering an increasing number of other financial service. Domain architecture at PostFinance[4] is positioned as a passive instrument that makes interdependencies transparent. Domains are primarily structured according to products and channels. Domain architecture is managed by a central body outside IT. Information about the stakeholder and viewpoints of the dimensions is not available. An overview is provided by Table 5.

3.2 Learnings from Practice Cases

This first assessment of the resulting distribution of characteristics along the dimensions and values of the proposed morphology (see Table 6) indicates improvement potentials especially in the *target* category. It could be the case that *decentralization of responsibilities* is not a discrete application scenario and that *increasing transparency* is a too general scenario that needs further detailing. The same might hold true for the dimension *design and evolution principles* in the *activities* category where the validity of the potential value *reuse* and the granularity of the value *consistency* need further investigation. Finally, within the *stakeholder* dimension, *business* does not seem to be a discrete value, which might be attributed to the generic IT affinity of artifacts such as applications, services, and information flows.

In contrast, the dimension *domain definition* is the only dimension whose potential values are all reflected in the sample, indicating a good differentiating potential of the

[4]http://www.postfinance.ch/pf/content/en.html.

Table 5 PostFinance Approach Specified Using the Proposed Morphology

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	**Not available**			
Results	Domain definition	Business processes	Business entities	Business dimensions	
	Viewpoints	Not available			
Activities	Design and evolution principles	Consistency	Reuse	Loose coupling	
	Implementation approach	Architecture communication	Architecture lobbying	Architecture enforcement	
	Organization	Centralized	Diversified	Distributed	Decentralized

Table 6 Overview Occurrences of Potential Values in Practice Cases

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	Business	IT	Business and IT	
Results	Domain definition	Business processes	Business entities	Business dimensions	
	Viewpoints	Application inventory	Functions inventory	Domain landscape	Application landscape
Activities	Design and evolution principles	Consistency	Reuse	Loose coupling	
	Implementation approach	Architecture communication	Architecture lobbying	Architecture enforcement	
	Organization	Centralized	Diversified	Distributed	Decentralized

Zero occurrences — Four occurrences

morphology in this regard. Due to the small sample size and some missing values, no general assessment should be made for the remaining dimensions *viewpoints*, *implementation approach*, and *organization*. However, these preliminary findings will be further investigated in the following section.

4 EXPLORATIVE EVALUATION OF THE PROPOSED MORPHOLOGY

This section presents the domain architecture management approach of Suva,[5] a Swiss insurance company. By structuring this case along the findings of the previous section, a first explorative validation of the proposed morphology and its potential to identify patterns for later construction of a situational method for domain architecture management is provided. For the analysis of the Suva case, presentations and process documentation were analyzed and an interview with one of the lead architects of Suva was conducted. The results that are presented herein were reviewed thoroughly and approved by Suva's architecture team and communications department.

4.1 Company Profile

Formed in 1918, Suva has a total workforce of around 2,900 employees who are based at its head office in Lucerne, at its two rehabilitation clinics in Bellikon and Sion, and at its 19 agencies throughout Switzerland. A financially independent body incorporated under public law, Suva insures around 110,000 companies and 2 million employees (as well as unemployed people) against the consequences of accidents and occupational diseases. It is also responsible for military insurance by government mandate. Its range of services encompasses prevention, insurance, and rehabilitation. Suva communicates this wide range of services under the following brands: SuvaPro (occupational safety), SuvaLiv (leisure time safety), SuvaRisk (premiums and capital investment), and SuvaCare (claims management and rehabilitation). Table 7 provides a short overview on Suva in general and its financial performance in 2008.

Table 7 Company Profile of Suva

Suva (2008)	
Head office	Lucerne
Industry sector	Insurance
Business segments	Prevention, insurance, and rehabilitation
Turnover (in m CHF)	7,919
Profit (in m CHF)	-149
Companies insured	114,882
Insurees	2,008,000
Employees (average)	2,904
Contact	http://www.suva.ch

[5]http://www.suva.ch/en/home_en.

4.2 Domains in Suva Architecture Management

The domain architecture is one strategic aspect of Suva's service-oriented architecture management (SOAM) program that was started in the first half of 2007. The domain architecture is targeted to serve three consecutive purposes. In step one, transparency is created on (the most relevant) business services that are needed to support Suva's operations. In step two, the potential for reusing existing business functionality is identified and responsibilities for consolidation efforts are delegated. Finally, in step three, the flexibility is increased by decoupling domains through an enterprise service bus. So far, step one is implemented in full, while the process of consolidation and decoupling is still in progress.

The Suva domain model is clustered into four domain types and the domain definition/separation is predominantly structured along business processes (see Figure 1). A suitable set of functional service domains was identified through a joint business-IT project. This approach accounted, on one hand, for the existing interdependencies in the actual application system landscape (bottom-up analysis of IT) and, on the other hand, incorporated a business perspective on the target state (top-down analysis of business).

Dependent on the individual complexity, a domain can comprise several sub-domains. Each domain/sub-domain is then characterized by (unambiguous) associated business services (e.g., claim elicitation) and respective business entities. Each business service is implemented by an application system that provides access to the functionality of the business service through one or more dedicated interfaces (see Figure 2).

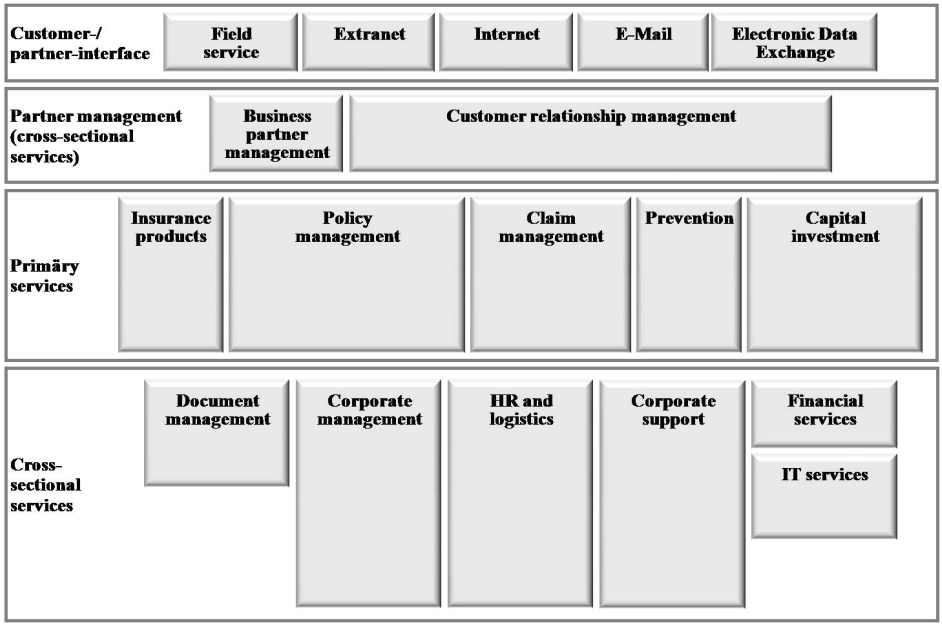

Figure 1 Suva's Functional Service Domain Model

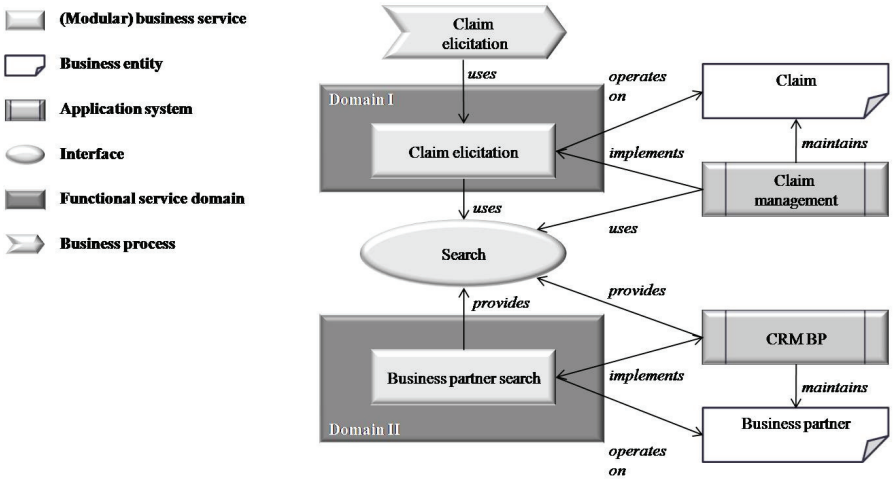

Figure 2 Elements of a Functional Service Domain at Suva

In order to reduce interdomain dependencies, Suva is about to implement several design and evolution principles. The most relevant principles to ensure loose coupling of domains are (so far)

- An application system needs to be unambiguously assigned to one single functional service domain/sub-domain.
- The access of business functionality in a different domain must occur through centrally managed interfaces.
- One business service needs to be implemented through a single application system. One application system can implement one or several business services.

Every development project is enforced to comply with these and other design and evolution principles. If a project is assessed to have architectural relevance during its specification phase, the solutions design needs approval by a central architecture board (AB). The AB bases its decision on the assessment/recommendations of the domain center of excellence (DCE) and the architecture center of excellence (ACE). The DCE is comprised of the responsible domain architect and the responsible business analyst. It assesses the functional architecture concept. The ACE is comprised of the responsible application systems architect and the responsible integration architect. It assesses the technical architecture design. Even though the involvement of central architects is mandatory, they are not entitled to strictly enforce the above specified design and evolution principles. Currently the respective governance is revised, but it remains unclear if this will result in a strengthening of the role of the central architects. In summary, the domain architecture management approach of Suva can be structured along the proposed morphology as specified in Table 8.

Table 8 Suva Domain Architecture Specified Using the Proposed Morphology

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	Reducing complexity	Decentralization of responsibilities	
	Stakeholder	Business	IT	Business and IT	
Results	Domain definition	Business processes	Business entities	Business dimensions	
	Viewpoints	Application inventory	Functions inventory	Domain landscape	Application landscape
Activities	Design and evolution principles	Consistency	Reuse	Loose coupling	
	Implementation approach	Architecture communication	Architecture lobbying	Architecture enforcement	
	Organization	Centralized	Diversified	Distributed	Decentralized

4.3 Learnings from Suva Case

In order to identify improvement potential for the proposed morphology, its information value was collaboratively assessed with one of the leading architects of Suva in a semi-structured interview. The following five questions framed the discussion:

1. Are the dimensions and potential values complete (with respect to the findings of the previous section)?
2. Is each dimension/potential value relevant for the identification of patterns in practice?
3. Is the Suva case unambiguously classifiable along the morphology?
4. Do the dimensions and potential values yield the appropriate level of detail, differentiating enough on one hand, but still abstract enough to identify patterns (with respect to the findings of the previous section)?
5. Is the assumed maturity logic in the dimensions *application scenario*, *viewpoints*, *design and evolution principles*, and *implementation approach* compelling?

While the derived morphology was assessed to fulfill the requirements stated in question 2, the interviewee challenged the morphology regarding questions 1, 3, 4, and 5. Regarding question 1, the interviewee suggested adding a further dimension in category *results*. In addition to the dimension *domain definition*, a dimension *domain definition approach* should be added to assess whether the domain landscape was structured from a functional perspective (*top-down*), from an IT perspective (*bottom-up*), or a combination of both (*meet-in-the-middle*). This dimension could reveal interesting relations to the targeted audience and the design and evolution principles applied. Regarding question 3 (in combination with question 5), the Suva case was not unambiguously classifiable in the dimension *design and evolution principles* since the assumed

Table 9 Adapted Morphology After Explorative Evaluation

Category	Dimension	Potential values			
Target	Application scenarios	Increasing transparency	Consolidating	Increasing flexibility	Decentralization of responsibilities
	Stakeholder	IT			Business and IT
Results	Domain definition	Business processes	Business entities		Business dimensions
	Domain definition approach	Top-down	Bottom-up		Meet-in-the-middle
	Viewpoints	Application inventory	Functions inventory	Domain landscape	Application landscape
Activities	Design and evolution principles	Consistency	Reuse		Loose coupling
	Implementation approach	Architecture communication	Architecture lobbying		Architecture enforcement
	Organization	Centralized	Diversified	Distributed	Decentralized
x New x Adapted *x* Deprecated					
Target	*Application scenarios*	*Increasing transparency*	*Reducing complexity*		*Decentralization of responsibilities*
	Stakeholder	*Business*	*IT*		*Business and IT*

maturity logic does not hold true for this dimension. The intention for reuse is not a prerequisite for the loose coupling of an application landscape. Therefore, the interviewee suggested either concentrating on the predominant design and evolution principle of a company (comparably to the dimension *domain definition*) or allowing in this particular case for multiple answers. Regarding question 4, the potential values of the dimension *application scenarios* were not capable of reflecting the three-level approach of Suva (first, install transparency; second, consolidate; third, increase flexibility). In order to increase the differentiating potential of the morphology, the potential value *reducing complexity* could be broken down into the two components *consolidating* (reduce data and function redundancy) and *increasing flexibility* (while on top reducing the interface complexity by means of bus technologies). The adapted morphology that results from these lessons from of the Suva case is illustrated in Table 9.

5 CONCLUSIONS

This article proposes a morphology to classify domain architecture management approaches in practice. The dimensions and respective values of the model were initially derived from literature and validated/amended in a two-stage approach. In step one, a first assessment of four cases from the literature resulted in a couple of open questions regarding the differentiating potential of the proposed morphology. In step two, a single

case was analyzed in-depth in order to answer the open questions of step one and to validate the completeness, clearness, and relevance of the proposed model. Following the ideas of iterative artifact construction in design research, the lessons from this case resulted in various adaptations to the proposed morphology. While the resulting artifact has undergone first validations and evaluations, it does certainly lack representativeness. Therefore, further cases need to be analyzed to allow for a thorough validation.

In the cause of further research, three successive steps are necessary. First, as mentioned above, further evaluation of the proposed morphology is required to validate its utility and strengthen its recommendation character. The necessary, thorough evaluation can be achieved through applying the proposed morphology in a field experiment with additional practice cases. An ongoing iterative approach for conducting these case studies seems most promising in order to identify and justify possible adaptations to the morphology. Second, a representative survey should be conducted in order to identify patterns along the refined morphology. Cluster analysis could be used to reveal a limited number of relevant use cases for domain architecture management and common associations between certain values in different dimensions. Based on the findings of the cluster analysis, the third and final step would be the design of a situational approach to domain architecture management. By the means of either method configuration or method fragment aggregation, activities as well as roles, result specifications, and techniques will then be constructed in a way that allows them to be adapted to a particular positioning of domain architecture management in a company or a government agency.

References

Aier, S. 2007. *Integrationstechnologien als Basis einer nachhaltigen Unternehmensarchitektur - Abhängigkeiten zwischen Organisation und Informationstechnologie*, Berlin: Gito.

Aier, S., Kurpjuweit, S., Saat, J., and Winter, R. 2009. "Enterprise Architecture Design as an Engineering Discipline," *AIS Transactions on Enterprise Systems* (1:1), pp. 36-43.

Bucher, T., Klesse, M., Kurpjuweit, S., and Winter, R. 2007. "Situational Method Engineering: On the Differentiation of 'Context' and 'Project Type,'" *Situational Method Engineering: Fundamentals and Experiences*, S. Brinkkemper, B. Henderson-Sellers, and J. Ralyte (eds.), Boston: Springer, pp. 33-48.

Cherbakov, L., Galambos, G., Harishankar, R., Kalyana, S., and Rackham, G. 2005. "Impact of Service Orientation at the Business Level," *IBM Systems Journal* (44:4), pp. 653-668.

Dietzsch, A. 2008. *Enterprise Architecture als Teil der strategischen Unternehmensentwicklung*, St.Galler Anwenderforum, St. Gallen, Switzerland.

Dodd, J. 2005. "Practical Service Specification and Design Part 1: Planning the Services," *CBDi Journal,* March, pp. 22-29.

Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J., Voss, M., and Willkomm, J. 2008. *Quasar Enterprise - Anwendungslandschaften serviceorientiert gestalten*, Heidelberg: Dpunkt.verlag.

FEA PMO. 2007. "FEA Practice Guidance," Federal Enterprise Architecture Program Management Office (available online at http://www.whitehouse.gov/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf).

Gutzwiller, T. A. 1994. *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*, Heidelberg: Physica.

Hafner, M., and Winter, R. 2008. "Processes for Enterprise Application Architecture Management," in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, Los Alamitos, CA: IEEE Computer Society Press, pp. 396-405.

Hagen, C. 2003. "EAI@CS - Credit Suisse Integration Architecture," in *Proceedings des EAI Tag Berlin*, Berlin: Lehrstuhl für Systemanalyse und EDV der Technischen Universität Berlin, pp. 1-26

Harmsen, A. F. 1997. *Situational Method Engineering*, dissertation, University of Twente, Twente, The Netherlands.

Heutschi, R. 2007. *Serviceorientierte Architektur*, dissertation, Universität St. Gallen, St. Gallen, Switzerland.

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.

Heym, M. 1993. *Methoden-Engineering - Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme*, dissertation, Universität St. Gallen, St. Gallen, Switzerland.

IEEE. 2000. "IEEE Recommended Practice for Architectural Description of Software Intensive Systems," IEEE Std 1471-2000, Software Engineering Standards Committee of the IEEE Computer Society, New York (available online at http://www.dia.uniroma3.it/~cabibbo/ids/altrui/ieee1471.pdf).

Josuttis, N. 2008. *SOA in der Praxis - System-Design für verteilte Geschäftsprozesse*, Heidelberg: Dpunkt.verlag.

Jung, E. 2004. "Ein unternehmensweites IT-Architekturmodell als erfolgreiches Bindeglied zwischen der Unternehmensstrategie und dem operativen Bankgeschäft," *Wirtschaftsinformatik* (46:4), pp. 313-314.

Kumar, K., and Welke, R. J. 1992. "Methodology Engineering: A Proposal for Situation-Specific Methodology Construction," in *Challenges and Strategies for Research in Systems Development*, W. W. Cotterman and J. A. Senn (eds.), New York: John Wiley & Sons, pp. 257-269.

Kurpjuweit, S. 2009. *Stakeholder-orientierte Modellierung und Analyse der Unternehmensarchitektur*, dissertation, Universität St. Gallen, St. Gallen, Switzerland.

Lankes, J., Matthes, F., and Wittenburg, A. 2005. "Softwarekartographie: Systematische Darstellung von Anwendungslandschaften," *Internationale Tagung Wirtschaftsinformatik*, Heidelberg: Physica Verlag, pp. 1443-1462.

Lankhorst, M. 2005. *Enterprise Architecture at Work: Modelling, Communication and Analysis,* Berlin: Springer.

March, S. T., and Smith, G. F. 1995. "Design and Natural Science Research on Information Technology," *Decision Support Systems* (15:4), pp. 251-266.

Niemann, K. D. 2006. *From Enterprise Architecture to IT Governance: Elements of Effective IT Management*, Wiesbaden: Vieweg.

Niemi, E. 2007. "Enterprise Architecture Stakeholders: A Holistic View," in *Proceedings of the 13th Americas Conference on Information Systems*, Keystone, Colorado, August 9-12.

Open Group. 2009. "The Open Group Architecture Framework (TOGAF), Version 9, Enterprise Edition," Document No. G091, The Open Group, Reading, Berkshire, United Kingdom.

Op't Land, M., Proper, E., Waage, M., Cloo, J., and Steghuis, C. 2009. *Enterprise Architecture: Creating Value by Informed Governance*, Berlin: Springer.

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., and Bragge, J. 2006. "The Design Science Research Process: A Model for Producing and Presenting Information

Systems Research," in *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology*, Claremont, CA, February 24-25, pp. 83-106.

Pohland, S. 2000. *Globale Unternehmensarchitekturen – Methode zur Verteilung von Informationssystemen,* Berlin: Weißensee-Verlag.

Richter, J. P., Haller, H., and Schrey, P. 2005. "Serviceorientierte Architektur," *Informatik Spektrum* (28:5), pp. 413-416.

Rohloff, M. 2008. "Framework and Reference for Architecture Design," in *Proceedings of the 14th Americas Conference on Information Systems*, Toronto, Canada, August 14-17, pp. 1-14.

Rossi, M., and Sein, M. K. 2003. " Design Research Workshop: A Proactive Research Approach," presentation to the 26th International Systems Research in Scandinavia (IRIS) Conference, Porvoo, Finland, August 9-12 (available online at http://www.cis.gsu.edu/~emonod/epistemology/Sein%20and%20Rossi%20-%20design%20research%20-%20IRIS.pdf, 23.04.2009).

Schelp, J., and Winter, R. 2008. "Entwurf von Anwendungssystemen und Entwurf von Enterprise Services – Ähnlichkeiten und Unterschiede," *Wirtschaftsinformatik* (50:1), pp. 6-15.

Schlamann, H. 2004. "Service Orientation: An Evolutionary Approach," *Cutter IT Journal* (17:5), pp. 5-13.

Schwinn, A. 2006. *Entwicklung einer Methode zur Gestaltung von Integrationsarchitekturen für Informationssysteme,* dissertation, Difo-Druck, Bamberg, Germany.

Sinz, E. J. 1999. "Architektur von Informationssystemen," in *Informatik-Handbuch*, P. Rechenberg, and G. Pomberger (eds.), Munich: Hanser Fachbuch, pp. 1035-1046.

van Slooten, K., and Hodes, B. 1996. "Characterizing IS Development Projects," in *Method Engineering: Principles of Method Construction and Tool Support*, S. Brinkkemper, K. Lyytinen, and R. Welke, London: Chapman & Hall, pp. 29-44

vom Brocke, J. 2003. *Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen*, Berlin: Logos Verlag.

Winter, R., Bucher, T., Fischer, R., and Kurpjuweit, S. 2007. "Analysis and Application Scenarios of Enterprise Architecture: An Exploratory Study" (Reprint), *Journal of Enterprise Architecture* (3:3), pp. 33-43.

Winter, R., and Fischer, R. 2006. "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture," in *Proceedings of the EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006)*, Los Alamitos, CA: IEEE Computer Society Press, pp. 30-37.

Winter, R., Gericke, A., and Bucher, T. 2009. "Method Versus Model – Two Sides of the Same Coin?," in *Advances in Enterprise Engineering III*, A. Albani, J. Barijis, and J. L. G. Dietz (eds.), Berlin: Springer, pp. 1-15.

Ylimäki, T. 2006. "Potential Critical Success Factors for Enterprise Architecture," *Journal of Enterprise Architecture* (2:4), pp. 29-40.

About the Authors

Daniel Stock is a research assistant at the Institute of Information Management, University of St. Gallen (HSG), Switzerland. He received a Diploma (Master equivalent) in Business Engineering (2007) from University Karlsruhe, Germany. Daniel can be reached by e-mail at daniel.stock@unisg.ch.

Robert Winter is a full professor of business and information systems engineering at University of St. Gallen (HSG), director of HSG's Institute of Information Management, and academic director of HSG's Executive Master of Business Engineering programme. After master

studies in business administration and business education at Goethe University, Frankfurt (Germany), he joined Frankfurt's institute of information systems for 10 years before being tenured in St. Gallen in 1996. Dr. Winter's research interests are information logistics management (since 1999), enterprise architecture management (since 2000), integration management (since 2002), healthcare networking (since 2005), and corporate controlling systems (since 2006). He is department editor of *Business & Information Systems Engineering* (formerly *Wirtschaftsinformatik*) and member of the editorial boards of *Information Systems and e-Business Management*, *Enterprise Modelling and Information Systems Architectures*, *AIS Transactions on Enterprise Systems*, and *International Journal of Organizational Design and Engineering*. Dr. Winter can be reached by e-mail at robert.winter@unisg.ch.

Jörg H. Mayer is a project manager at the Institute of Information Management, University of St. Gallen (HSG), Switzerland. He received a Diploma (Master equivalent) in Business Administration and Industrial Engineering (1994) from University Darmstadt, Germany. As a research assistant with University Darmstadt, he received a doctorate in social sciences for his work in the field of executive information systems (1999). Dr. Mayer can be reached by e-mail at joerg.mayer@unisg.ch.