A Method Base for Enterprise Architecture Management

Sabine Buckl, Florian Matthes, and Christian M. Schweda

Technische Universität München, Institute for Informatics,
Boltzmannstr. 3, 85748 Garching, Germany
{sabine.buckl,matthes,christian.m.schweda}@mytum.de
http://wwwmatthes.in.tum.de

Abstract. Responding to the increasing complexity and diversity of information systems development, method engineering provides techniques and tools for the analysis, design, and evolution of information systems. Similar challenges can be found in the research area of enterprise architecture (EA) management, whose main goal is to enhance the alignment of business and IT. While a multitude of methods and models to support EA management have been proposed over the years, situational factors as the goals pursued or the organizational context, in which the management function has to be embedded, are typically neglected.
In this paper, we present a building block base for the design of situated EA management functions based on a comprehensive collection of best practice methods and models for EA management. Therefore, we discuss related work from the area of situational method engineering and pattern-based development and design. Based on these foundations, a building block base and the contained building blocks are presented and are applied alongside a case study from industry. The discussion is complemented by a prototypic tool implementation, which can be used to support the configuration process for situational methods.

1 Motivation

The enterprise forms a complex structure constituted of a large number of highly interdependent elements. The constant need to adapt this structure in response to changing external influences, as economic factors or new regulations, calls for an embracing approach to control and govern the necessary transformations. Enterprise architecture (EA) management is a discipline aiming to provide guidance for the enterprise transformation by taking a holistic perspective on the enterprise, covering concepts from the business to the IT infrastructure level, but also accounting for cross-cutting aspects as strategies, projects, or standards.

The embracingness of the management subject raises different implications relevant to EA management as a function. Most obvious, the holistic perspective taken requires a large amount on information about the architecture elements as well as their interdependencies. Collecting the relevant information, but also keeping the information up-to-date, communicating it to the interested parties

(*stakeholders*) in the organization, or performing analyses are tasks, whose complexity grows with the rising amount of information to handle. This and the plurality of possible stakeholders as well as goals to pursue are two reasons that have promoted the development of the plethora of EA management approaches as found in today's literature. These approaches, e.g. The Open Group Architecture Framework (TOGAF) [29], the Archimate language [20], or Core Business Metamodel [22], encompass general prescriptions on how to manage the EA together with conceptual meta-models, the so-called *information models*, for the corresponding management body. Accounting for the fact that each organization has its specific understanding of EA management and the associated goals, the general prescriptions are often complemented with statements that highlight the need to adapt to the using organization. When it nevertheless comes to concrete artifacts describing the *design* of both organization-specific EA management methods and EA description languages, literature becomes more scarce.

Recent publications of Leppänen et al. [21] and of Riege and Aier [24] emphasized the topic of adapting EA management prescriptions to the specifics of the using organization, delineating potential *contingency factors* of EA management. Winter et al. further analyze in [30] the plurality of EA management goals as pursued in practice. These publications indicate towards a more mature understanding of the field (see Section 2 for a detailed discussion). From this dedicated method engineering approaches for EA management can be considered the next research step, which is nevertheless aggravated by the typical challenges of EA management research as outline by Buckl et al. in [8]: practice-relevant EA management research is usually carried out in close cooperation with the industry, such that the projects follow the industry partner's pace and have to deliver their benefits early. On the contrary, the field itself has a broad subject, is rooted in a multi-disciplinary background, and the effects of measures taken usually manifest in the long run after a couple of years. In Section 3 we describe an approach for designing situated EA management functions based on the foundation of method engineering. For the aspect of administering the knowledge base of the approach, we discuss how a pattern-based understanding of EA management, as taken by Buckl et al. in [5], is called upon. Section 4 delineates the steps of applying the approach both from a theoretical perspective and along an anonymized practical example. Final Section 5 summarizes the findings of the article and gives a brief outlook.

2 Related work

An EA management function can be understood as a design product embedded into the context of using organization. Riege and Aier conducted in [24] an exploratory analysis of the contingency factors, that result from this, and derived a *contingency framework* consisting of three factors:

- *adoption of advanced architectural design paradigms and modeling capabilities*: targeting properties as the coverage of current and target states of the EA in the architectural models as well as of transformation plans

- *deployment and monitoring of EA data sets and services*: concerning the control and governance for the EA management processes via performance reviews and the availability of dedicated EA management marketeers
- *organizational penetration of EA*: aiming at the organizational perception of EA management in the IT departments and business units as well as the usage of EA management-provided services in these departments.

Based on the empiric results on these factors and the constituting items, Riege and Aier cluster the analyzed EA management functions into three different types: *engineering functions*, *incepting functions* and *extended IT architecture functions*. Former distinction supported from the empiric point of view nevertheless reveals a main limitation of the analysis. The analysis' results fail to support a clear distinction between the contingency factors and their effects. This may ascribe to the special nature of the interplay between the EA management function and its management body, but provides only minor support for understanding the impediments and catalysts of managing the EA.

In [21] Leppänen et al. pursue a different approach in deriving the contingency factors of EA management. Based on the findings and experiences of the Finnish EA research program, they elicit their contingency framework (EACon) for EA management, providing the following categories of contingency factors:

- *EA method goals* reflect the stakeholder's requirements that the EA management function is meant to satisfy. These stakeholders can be located in different organizations participating in the EA management.
- *EA principles* delineate constraints pertaining to the EA management function, such governance rules. These principles may be local to one organization or be shared in the organization network.
- *Roles* refer to the people to be involved in both the EA management function as well as in the corresponding governance. Possible roles are the enterprise architect, as *method user*, and the *EA method engineer*.
- *Resources* reflect manpower, monetary supplies and tool support that is available to the EA management function in the participating organizations.
- *Cluster* describes the organizational environment into which the EA management function is to be embedded. Such cluster can be single organization or a network of enterprises cooperating to provide networked services.

Each of the above factors can according to Leppänen et al. [21] be further detailed. For the *cluster* this reads as a more detailed understanding of the *organizational culture* and *organizational structure*, of which the latter is closely related to the factor *decision rights* constituting a part of the *roles*. Making explicit these different factors as well as the intricate relationships inbetween is the core contribution presented in [21] by Leppänen et al. This clearly mirrors the focus of the article, that seeks to contribute to *a contingency framework for engineering an EA planning method*. It is hence not surprising that the particular factors of EACon remain decoupled from concrete solutions, guidelines, or prescriptions on how to optimally manage the EA given certain contingencies.

In [5] Buckl et al. take a different perspective on the field of EA management. Experiencing the need for concrete and practice-proven solutions to recurring EA management problems, the authors translate the notion of the *pattern* (cf. Alexander et al [2]) to the field of EA management research. For the field of EA management, Buckl et al. introduce three different types of pattern as follows:

- *method pattern* define steps to be taken in order to address a given problem. Furthermore, as a guidance for applying the method, statements about its intended usage context are provided.
- *viewpoint pattern* define the notations used by the methods, i.e. describe ways to present information necessary for performing one or more methods as stored according to one or more information model patterns.
- *information model pattern* supply models structuring the information needed by one or more methods and visualized in one of more viewpoints.

For solving a particular EA management problem in a given organization, a user selects an appropriate set of method, viewpoint, and information model patterns. Based on these constituents, a user composes his specific EA management function, i.e. defines what would in line with Gutzwiller [14] be called an *organization-specific EA management method*. Former term sheds a light on the slightly uncommon understanding of *method* in the work of Buckl et al., e.g. in [5]. While usually design-oriented methods are understood as constituted from *roles*, *tasks*, *techniques*, *design results*, and a corresponding *meta-model*, Buckl et al. separate the roles, tasks and technique (the *method* in their understanding) from the design results with their corresponding meta-model (views according to *viewpoints* and *information models* in their terms). This separation can be justified against the background of the stereotypic tasks employed in EA management and manifests the so-called *method-language-dichotomy* as alluded to e.g. by Schelp and Winter [26] or by Buckl et al. in [8]. This dichotomy is utilized in Section 3 to formulate independent building-blocks for an EA management function in refinement of the EA management patterns.

Helping the user in selecting the suitable EA management pattern, the EA management pattern catalog [11] refines the basic idea and supplies a set of relationships between the different patterns. In particular, every method pattern references all viewpoint patterns that can be used in the method, whereas each viewpoint pattern relates to the information model pattern, covering the needed information. Pattern of all three types are described using a template resembling the so-called *canonical pattern form* (cf. Ernst [12]). Each pattern states:

- its *usage context* in which it can be applied,
- the *problem* that it has proven to solve,
- the *solution* which it applies to the given problem,
- the contradictory *forces* framing the space of observed solutions, and
- the *consequences* observed to result from the pattern's application.

With this standardized structure, the patterns can serve as valuable starting point for developing an approach for designing situated EA management functions. Such approach nevertheless has to deal with the inherent weaknesses of EA

management patterns, e.g. their tendency to repeat themselves especially with respect to methods for documenting the EA, as well as terminological plurality of the pattern descriptions, resulting from the fact that patterns are observed in different practice cases without consistent overarching terminology.

3 Designing a situated EA management function

In line with the understanding of Harmsen that *there is no method that fits all situations* [15, page 6], we subsequently propose a situated approach to design an EA management function based on existing best practices. A situated approach according to Harmsen in [15] accomplishes standardization and at the same time flexibility to match the situation. A *situation* thereby refers to the combination of circumstances at a given point in time in a given organization [15]. In order to address these requirements, for each situation a suitable solution[1] – so-called situational solution – is constructed that accounts for these circumstances. Reflecting the method-language dichotomy in EA management, two different types of solution constituents, *method building blocks* (MBBs) and *language building blocks* (LBBs), are used in the construction process and are configured as well as adapted with the help of formally defined guidelines.

3.1 Foundations

A complex and intricate research area like designing EA management functions represents a topic that is not easy to research. The heavy involvement of stakeholders, the broadness of the subject, and the delayed effects (see Section 1) call for a suitable structuring of the research subject. This structuring can be performed either using a *vertical* or *horizontal* domain decomposition strategy. In the *vertical* domain decomposition only a limited number of EA-related problems is addressed in an embracing manner with a comprehensive solution. In contrast, the *horizontal* domain decomposition addresses a variety of EA-related problems with either suitable management methods or modeling languages.

While the former type of decomposition is a frequently used one for approaching the area of EA management(cf. Johnson and Ekstedt in [17], which focus on EA analysis or Spewak in [28] emphasizing on EA planning aspects), we in line with Schelp and Winter in [26] opt for a horizontal decomposition of the domain of EA management, which reflects the method-language dichotomy as discussed above. Such an approach is further backed by TOGAF, which contains the *architecture development method* – reflecting the methodical perspective – and the content framework – representing the language part (cf. Open Group in [29]). In contrast to the approach taken by TOGAF, we advocate for making the interconnection points between the methodical and the language parts explicit. In order to do so, we introduce the *variable* concept in the MBBs serving as a placeholder for language aspects. *Viewpoint variables* for instance are

[1] We subsequently employ the term solution instead of *method* in line with argumentation at the end of Section 2.

used during method description to indicate placeholders for visual architecture descriptions, which must be filled during the organization-specific configuration.

The idea of interrelating best practice fragments to design an organization-specific EA management function can only be realized against the basis of a common understanding and terminology of the topic. Although no such common understanding has yet evolved if the definition of the term *EA* or *EA management* is considered (cf. Schönherr in [27] and Schelp and Winter in [25]), consensus on the fundamental activities and tasks that make up an EA management function exists (cf. [1]). In [10], Buckl et al. revisit different approaches to EA management, e.g. Frank [13], Riege and Aier [24], and The Open Group [29] to devise a method framework for EA management consisting of four activities as shown in Figure 1:

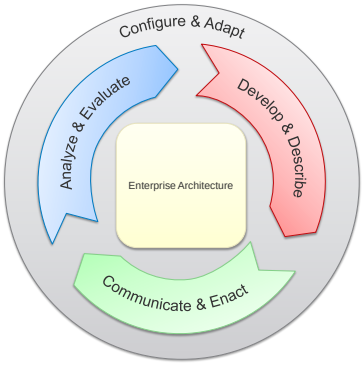

Fig. 1. Method framework of BEAMS

Develop & describe a state of the EA, either a current state describing the as-is architecture, a planned state representing a medium-term future state or a target state, i.e. a vision of the EA.

Communicate & enact architecture states and principles guiding the evolution of the EA to EA-relevant projects and to related management functions, e.g. project portfolio management.

Analyze & evaluate the current state to identify potentials for improvement, evaluate architectural scenarios (planned states), or analyze whether a planned state helps to achieve the target state or not.

Configure & adapt the EA management function itself, e.g. in response to an under achievement of the desired results or a changed situation decide on the addressed management concerns, pursued goals, and used methods.

From the perspective of architectural descriptions, the former activities are characterized as *creating*, *using*, and *augmenting* the EA description. The latter activity – configure & adapt – incorporates the nature of a meta-activity as it is concerned with the design of the former three activities. In terms of the situational method engineering this activity encompasses the *process of situational method engineering* [15, page 45], i.e. the steps *characterization of the situation, selection of method fragments,* and *assembly of method fragments.*

Similar to the method aspect, also the language aspect can be subdivided. An EA management-relevant problem can be described by a concern, i.e. an area of interest, and a goal, i.e. an abstract objective. A concern represents an organization-specific conceptualization of the management subject, while goals complement this static perspective via a time-dependence or a notion of better and worse. In that sense the achievement of a goal, e.g. increase homogenization, can be operationalized via measurements, such that a goal provides an evaluation function, which can be used to guide the EA planning process.

3.2 Structure of the building block

As motivated above, two different types of building blocks to design an EA management function exits – MBBs and LBBs, of which the latter are subdivided into building blocks concerned with the conceptualization, i.e. areas of interest – information model building blocks (IBB) – and building blocks containing best practice visual representations – viewpoint building blocks (VBB).

Method building blocks

An MBB describes the different *tasks* that are performed in order to achieve a certain goal in a given organizational *context*. The MBB further specifies the ordering of the tasks and execution alternatives. For every alternative path the MBB also describes the *conditions* that apply during task execution. Furthermore, a task can devise different *techniques* to be utilized. To perform an expert-based analysis of a planned state for example, a pattern-based technique or an-indicator based technique can be utilized. Thus, each technique is linked to *forces* describing the benefits and drawbacks of the different techniques to be selected. Reflecting the method-language dichotomy each MBB contains a *concern variable*, i.e. a placeholder for the area of interest on which the tasks operate. During the configuration, the concern variable has to be replaced by the actual concern.

In order to be applied, the *pre-conditions* specified by an MBB need to be met. An exemplary precondition of an MBB dedicated to the analyze & evaluate activity is that the concern specified by the concern variable is already documented. Supplementary, each MBB also specifies *post-conditions* that are fulfilled after executing the MBB. In this vein, consistency checks can be executed ensuring a sensible configuration and ordering of MBBs. The above exemplary pre-condition illustrates the make-up of pre- and post-conditions, representing a combination of an area-of-interest, i.e. a concern, and a so-called *meat-attribute*, e.g. documented, acknowledged, or publicized, describing a property of the concern related to the MBB. In addition to the post-conditions, an MBB can specify consequences, which result from applying the building block. The notion consequence thereby does not only refer to negative side-effects but is also used to describe positive add-ons. In contrast to the post-conditions, consequences are described in an informal manner utilizing natural language descriptions.

Complementing each MBB contains a *trigger variable*, which specifies the trigger starting the execution of the tasks. In configuring the EA management function this variable is filled with an actual trigger. Each task is executed by a corresponding actor represented by an *actor variable* in the description of the

method. The notion of actor variable similar to the notion of trigger variable is used to denote that the description of the MBB does not specify distinct actors or roles in the using organization, but merely describes a responsibility of a person or group. Further, the MBB can specify that the actor variable is bound in respect to its organizational role, e.g. might express that an escalation based enactment mechanism only works, if a superordinate actor can be called upon. Beside to the mandatory relationship to the executing, i.e. *responsible* actor variable, each task may relate to other actor variables as well, namely variables representing actors that are *consulted* or *informed* during task execution. The distinction between the different levels of involvement pertaining to a single task is based on the RACI model of CobiT (see e.g. [16]), while a slightly different perspective is taken on the involvement level *informed*. For the purpose of describing MBBs, we assume that any actor involved in a task is informed, such that the responsible actor as well as consulted actors are counted as informed, too. The participation of actors in tasks is enabled via *viewpoint variables*, which designate that the actor takes a specific viewpoint on the information relevant during performing the given task. The notion of the variable here again describes that the MBB does not make concrete prescriptions on the viewpoint to be used, but in turn allows to select a specific viewpoint for accomplishing the task.

Language building blocks

In designing the language for EA descriptions, both VBBs and IBBs are employed. Understanding a language in line with e.g. Kühn [18] as constituted of *abstract syntax*, *semantics* and *notation*, the two types of building blocks are used to specify notation and syntax, respectively. The semantics is specified denotationally in the glossary complementing the building block base (cf. Section 3.3 below). Each information model building block specifies the *types*, *attributes* and *relationships* that conceptualize the corresponding part of the EA, i.e. cover a stakeholder's concern or reflect the information necessary for assessing the attainment of a goal. For the latter purpose, the IBBs introduce a distinction between different kinds of types, most notably distinguishing between *classes* and *mixins*. As Buckl et al. outline in [9], latter concepts can be used to formulate specialized IBBs are able to describe specific EA goals as *availability* regardless the actual EA concept, e.g. *business application* or *business capability*, they are attached to. A user can hence select the IBB reflecting a specific concern and combine it with a specific goal to his relevant EA problem, see Figure 2.

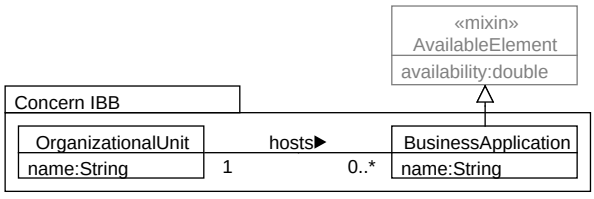

Fig. 2. Integrated information model

The VBBs focus on the notational aspects of the EA description language, providing visual primitives, e.g. *rectangles*, and visualization rules, e.g. *clustering*, for defining how the information is visualized. The VBBs further relate to the information model or parts of it, specifying concepts of which type are mapped to which kind of visualization element. Thereby, an executable transformation from the syntactic concepts to their visual counterparts is defined based on the VBBs. With the focus of this article on the method and information model perspective, we abstain from going into the details of the mechanism behind the transformation. More information can be found in [6].

3.3 Structure and administration of the building block base

Critical prerequisite to the design of a situated EA management function, is the provision of standardized building blocks, which are stored and retrievable from what is typically called a *method base* (cf. Brinkkemper in [3]) or *component base* (cf. Kumar and Welke in [19]). Due to the method-language dichotomy of our application domain, we abstain from reusing the misleading notions and introduce the term *building block base* for the repository. The structure of this repository is outline below and complemented by a description of the configuration and administration process. Enabling the selection of appropriate building blocks for a given situation requires the development of concepts and techniques to analyze and compare the incorporated building blocks. Following the idea of Pries-Heje and Baskerville in [23], we use the concepts of

problem A problem represents the issue to be solved by applying the building block. A problem in the area of EA management typically consists of a
  goal representing an abstract objective, e.g. increase homogeneity, provide transparency, and a
  concern , i.e. area of interest in the enterprise, e.g. business support, application systems.

organizational context The organizational context represents the situation in which the EA management function operates. Typical factors which are considered in the organizational context are the organizational culture, management commitment, or involved stakeholders.

Figure 3 illustrates the components and the structure of the building block base. To outline the administration of the building block base, we exemplify its development along the best practices for EA management as contained in the pattern catalog from [11]. Therefore, the problems addressed by the different patterns are analyzed and the abstract goals as well as the concerns are identified. Thereby, an exemplary goal reads as follows “increase homogeneity” and the respective concern is “technology used by a business application” [4]. The concerns and the respective information model patterns serves as input for the development of IBBs. Thus, also establishing a concern hierarchy, i.e. an evolution path, as introduced in [7]. Furthermore, the usage context descriptions are investigated for descriptions of organizational contexts in which the respective

pattern has been applied, e.g. “centralized IT organization”, “upper management support”, or “own budget for the EA management initiative”. The MBBs are derived from existing method patterns, thereby, the textual description of the steps to be taken is used as input to derive tasks, responsible actors, and forces. Furthermore, the consequence section of the patterns serve as input for the pre- and post-conditions of the building blocks. Consequences which can be formalized in terms of a meta-attribute and a respective concern are reformulated as pre- and post-conditions. The identified goals and organizational contexts are used as input for characterizing the situation.

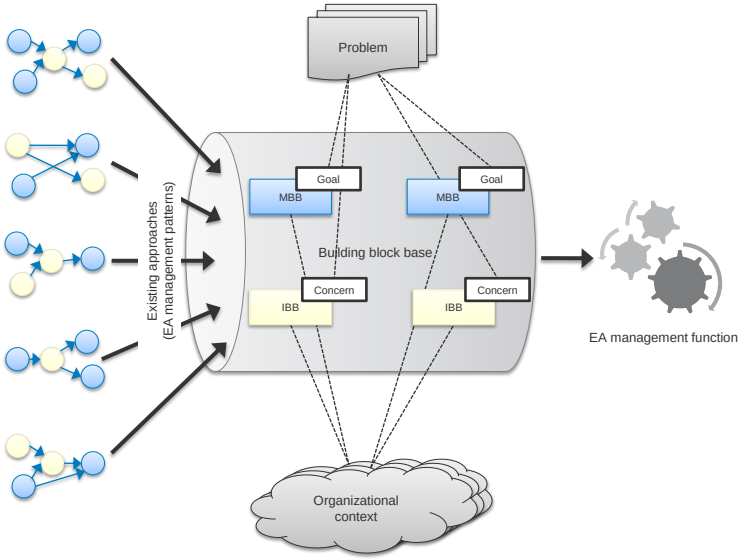

Fig. 3. The components and structure of the building block base

To relate the components of the building block base, the above identified goals of and organizational contexts are the derived building blocks. The suitability of a building block for any combination of the goals and organizational conditions can then be defined utilizing a *fitting matrix* with the building blocks on the y-axis, the identified organizational contexts (goals) on the x-axis, and a scoring of the fitting function for the MBBs (IBBs) in the cell. The fitting function can thereby take a value form the set *required, excludes,* or *helpful* and serves as a decision-support system for the selection of building blocks. The assembly of building blocks is performed utilizing the *variable* concept, i.e. the IBBs are used to configure the concern variable, and the pre- and post-conditions of BBs, which determine an ordering.

4 Applying the building block base

The building block base is applied using three steps: *characterization of the situation*, *selection of building blocks*, and *assembly of building blocks*, described below along an example from a financial service provider BSM.

Characterization of the situation
In this phase the existing organizational context in the enterprise is described. Further, the specific EA management goal to be pursued and the related EA concern are delineated. For doing so, the using organization can call on the list of contexts, goals and concerns as contained in the building block base.

> Resulting from previous acquisitions, BSM operates a highly heterogeneous landscape of business applications. Especially, maintenance of business applications developed a non-standard solutions for the formerly independent companies has become a costly task. Therefore, BSM seeks to *increase homogeneity* of the *business applications hosted at the different locations*. Due to the maintenance problems, the EA management can rely on *high-level management support* and is driven by a small EA management team located in a staff unit of the CIO's office. This team has to deal with the *highly decentralized structure* of the IT departments, such that the EA management process should be design to promote itself.

Selection of fragments
In this phase the building block base is searched for MBBs that match the organizational context and for IBBs that reflect the EA management problem. Figure 4 shows the results of a query against the building block base, returning IBBs containing the concept *application*. Similar searches are performed on the IBBs that reflect the cross-cutting aspect of standardization as well as on the MBBs starting with ones supporting the activity *develop & describe*.

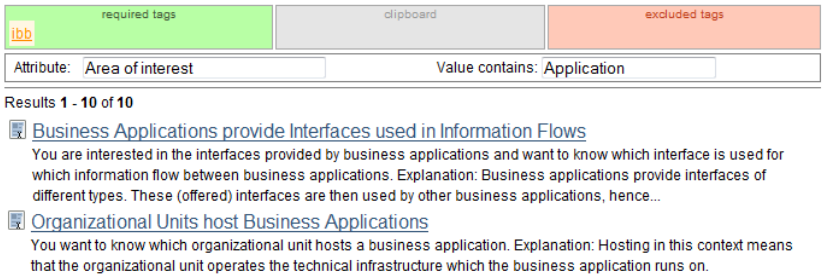

Fig. 4. Searching the building block based for suitable IBBs

The search for the applicable methods presents two MBBs as well-suited for the organizational context: *describe by interview* and *describe by workshop*, as both these MBBs are helpful to market the EA management endeavor. For the aspect of standardization, three models for standards reflected in different IBBs are provided by the building block base: *simple standardization*, *standardization via book of standards*, and *standardization by individual prescriptions*. Reading through the consequences of the different building blocks BSM decides to chose an interview-based gathering of information about business applications and their hosting

organizational units. For the aspect of standardization, a book of standards is to be created, marking certain technologies as standard or non-standard, respectively. For the phase *communicate & enact* the MBB *publish architectural descriptions* is chosen.

Assembly of fragments

In this phase four sub-phases are conducted to compose the building blocks to a comprehensive EA management function:

Integrate IBBs: the different IBBs reflecting the concerns and goals are integrated into composite models covering specific sub-problems of the EA management-problem to address. Thereby, manageable information models are created which can subsequently be linked to the MBBs.

Integrate and configure MBBs: the concern variable of each selected MBB is linked to the integrated information model that reflects the corresponding concern. If different concerns are to be treated by similar methods, the MBBs can be duplicated in this case. The configured MBBs are integrated into a process, and the consistency between the pre-conditions and post-conditions of consecutive MBBs is checked.

Configure actors and triggers: for each sequence of MBBs the triggering event is configured. Further, the actor variables defined by the MBBs are bound to actual organizational roles of the using enterprise.

Add and configure VBBs: for each actor, who participates in a task, a viewpoint variable exists. This variable is set to a composition of VBBs, expressing how certain parts of the corresponding concern (information model) are to be visualized. If the viewpoint variable is designed *read-only*, nearly no limitations on the type of viewpoint exist, whereas a *read-write* viewpoint is bound to comprise VBBs in a combination that represents an *updateable view*. This e.g. means that all information model elements intended to be written are represented 1:1 in the corresponding visualization.

During each of the aforementioned sub-phases, consistency checking is applied. During the integration of the IBBs the approach analyzes, if an incompatible semantic mapping is created, i.e. if *homonyms* in terms of the glossary of the building block base are created by unifying types with a distinct meaning. Concerning the configuration of the VBBs, the approach analyzes whether the visualized information is available in the method's corresponding concern variable. Further, it checks whether the transformation from syntactic to visual primitives is bidirective, or not, such that the latter case is diagnosed as non-updateable view, which cannot be used for write access.

BSM integrates the IBBs covering business applications, hosting organizational units and used technologies into a single information model. In contrast the information, whether a certain technology is standard according to the book of standard, is separated into a different information model. Former model, is assigned to the concern variable of *describe by interview*, whereas latter model is linked to *describe by*

workshop. The post-conditions of the two configure MBBs then read as OrgUnit-BusinessApplication-Technology.*documented* and as Technology.isStandard.*documented*. To the concern variable of the MBB *publish architectural descriptions* the information model subsuming both information models is assigned.
Further concretizing the documentation-specific MBBs, BSM decides that the interviews on the business applications and related information are held every time, when an IT-project finishes. An *enterprise architect* asks the *application owner* of the corresponding business application. The standardization-related information is decided upon every six months during a workshop by a board comprised of *enterprise architects*, *application owners* and *IT project managers*. A re-publishing of the architectural description is triggered every time, when new information about business applications is available. The *enterprise architects* are responsible for creating the corresponding architectural description, which is made available to *application owners*, *IT project managers*, and the *CIO*. With the viewpoint being read-only, BSM decides to use a clustered visualization containing only organizational units and business applications, of which the latter are colored red, if they use at least one non-standard technology, or green otherwise.

5 Summary and outlook

In this article we motivated the need for organization-specific EA management functions, i.e. management functions that account for the specificities of the using organization with both respect to the context and the management goals. Reflecting on the related work in Section 2, we showed what current research can already contribute to the design of such EA management functions and were able to delineate the omissions of current approaches. In Section 3 we applied the basic notions of method engineering onto the subject of EA management, describing an approach building on three types of building blocks for EA management functions, namely MBBs, VBBs, and IBBs. We further outlined how concrete building blocks look like and how they are interrelated in the *EA management building block base*, a EA management-specific implementation of the notion of the method base. In Section 4 we described how the building block base can be used to design an organization-specific EA management function. Thereby, we employed an example at a financial services provider.

With its roots in the EA management patterns of Buckl et al. [5, 11], the approach can rely on a sound and practice-proven basis of best-practice methods, viewpoints and information models. The rigorous mechanism for translating these patterns into building blocks further helps to ensure that the made prescriptions are applicable in practice. Notwithstanding, more in-depth evaluations of the usefulness of the approach remain to be conducted. Thereby, especially a comparison to less formal methods, e.g. TOGAF, are of interest. Such analyses are nevertheless subject for future research. In the context of the necessary

long-term analyses, it would further be interest to analyze, if the approach's prescriptions are beneficially for governing the EA management function, i.e. for evolving the function in response to organizational changes.

References

1. S. Aier, S. Buckl, U. Franke, B. Gleichauf, P. Johnson, P. Närman, C. M. Schweda, and J. Ullberg. A survival analysis of application life spans based on enterprise architecture models. In *3rd International Workshop on Enterprise Modelling and Information Systems Architectures*, pages 141–154, Ulm, Germany, 2009.
2. C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel. *A Pattern Language*. Oxford University Press, New York, NY, USA, 1977.
3. S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280, 1996.
4. S. Buckl, A. M. Ernst, J. Lankes, F. Matthes, and C. M. Schweda. Enterprise architecture management patterns – exemplifying the approach. In *The 12th IEEE International EDOC Conference (EDOC 2008)*, Munich, Germany, 2008. IEEE Computer Society.
5. S. Buckl, A. M. Ernst, J. Lankes, K. Schneider, and C. M. Schweda. A pattern based approach for constructing enterprise architecture management information models. In *Wirtschaftsinformatik 2007*, pages 145–162, Karlsruhe, Germany, 2007. Universitätsverlag Karlsruhe.
6. S. Buckl, J. Gulden, and C. M. Schweda. Supporting ad hoc analyses on enterprise models. In *4th International Workshop on Enterprise Modelling and Information Systems Architectures*, 2010.
7. S. Buckl, F. Matthes, and C. M. Schweda. Conceptual models for cross-cutting aspects in enterprise architecture modeling. In *5th International Workshop on Vocabularies, Ontologies, and Rules for the Enterprise (VORTE 2010)*, 2010.
8. S. Buckl, F. Matthes, and C. M. Schweda. From ea management patterns towards a prescriptive theory for desinging enterprise-specific ea management functions – outline of a research stream. In *Multikonferenz Wirtschaftsinformatik (MKWI2010)*, pages 67–78, Göttingen, Germany, 2010.
9. S. Buckl, F. Matthes, and C. M. Schweda. A technique for annotating ea information models. In J. Barjis, editor, *6th international workshop on Enterprise & Organizational Modeling and Simulation 2010*, Lecture Notes in Business Information Systems. Springer, 2010.
10. S. Buckl, F. Matthes, and C. M. Schweda. Towards a method framework for enterprise architecture management – a literature analysis from a viable system perspective. In *5th International Workshop on Business/IT Alignment and Interoperability (BUSITAL 2010)*, 2010.
11. Chair for Informatics 19 (sebis),Technische Universität München. Eam pattern catalog wiki. http://eampc-wiki.systemcartography.info (cited 2010-07-01), 2010.
12. A. M. Ernst. *A Pattern-Based Approach to Enterprise Architecture Management*. PhD thesis, Technische Universität München, München, Germany, 2010.
13. U. Frank. Multi-perspective enterprise modeling (memo) – conceptual framework and modeling languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 2002)*, pages 1258–1267, Washington, DC, USA, 2002.

14. T. A. Gutzwiller. *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen.* PhD thesis, Universität St.Gallen, 1994.
15. A. F. Harmsen. *Situational Method Engineering.* PhD thesis, University of Twente, Twente, The Netherlands, 1997.
16. IT Governance Institute. Framework Control Objectives Management Guidelines Maturity Models. `http://www.isaca.org/Knowledge-Center/cobit` (cited 2010-06-18), 2009.
17. P. Johnson and M. Ekstedt. *Enterprise Architecture – Models and Analyses for Information Systems Decision Making.* Studentlitteratur, Pozkal, Poland, 2007.
18. H. Kühn. *Methodenintegration im Business Engineering.* PhD thesis, Universität Wien, 2004.
19. K. Kumar and R. J. Welke. Methodology engineering: A proposal for situation-specific methodology construction. In *Challenges and Strategies for Research in Systems Development*, pages 257–270, West Sussex, England, 1992. John Wiley.
20. M. M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis.* Springer, Berlin, Heidelberg, Germany, 2nd edition, 2009.
21. M. Leppänen, K. Valtonen, and M. Pulkkinen. Towards a contingency framework for engineering and enterprise architecture planning method. In *30th Information Systems Research Seminar in Scandinavia (IRIS)*, pages 1–20, 2007.
22. H. Österle, R. Winter, F. Hoening, S. Kurpjuweit, and P. Osl. Business Engineering: Core-Business-Metamodell. *Wisu – Das Wirtschaftsstudium*, 36(2):191–194, 2007.
23. J. Pries-Heje and R. Baskerville. The design theory nexus. *MIS Quarterly*, 32(4):731–755, 2008.
24. C. Riege and S. Aier. A contingency approach to enterprise architecture method engineering. In *Service-Oriented Computing – ICSOC 2008 Workshops*, pages 388–399, 2009.
25. J. Schelp and R. Winter. On the interplay of design research and behavioral research – a language community perspective. In *Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology (DESRIST2008), May 7-9, 2008, Westin, Buckhead, Atlanta, Georgia, USA,*, pages 79–92, Georgia State University, Atlanta, Georgia, USA, 2008.
26. J. Schelp and R. Winter. Language communities in enterprise architecture research. In *DESRIST '09: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, pages 1–10, New York, NY, USA, 2009. ACM.
27. M. Schönherr. Towards a common terminology in the discipline of enterprise architecture. In S. Aier, P. Johnson, and J. Schelp, editors, *Pre-Proceedings of the 3rd Workshop on Trends in Enterprise Architecture Research*, pages 107–123, Sydney, Australia, 2008.
28. S. H. Spewak and S. C. Hill. *Enterprise Architecture Planning – Developing a Blueprint for Data, Applications, and Technology.* John Wiley & Sons, New York, USA, 1993.
29. The Open Group. TOGAF "Enterprise Edition" Version 9. `http://www.togaf.org` (cited 2010-02-25), 2009.
30. K. Winter, S. Buckl, F. Matthes, and C. M. Schweda. Investigating the state-of-the-art in enterprise architecture management method in literature and practice. In *Proceedings of the 5th Mediterranean Conference on Information Systems*, 2010.