A method to develop EA modeling languages using practice-proven solutions

Sabine Buckl, Florian Matthes, and Christian M. Schweda

Chair for Software Engineering of Business Information Systems (sebis),
Technische Universität München,
Boltzmannstr. 3, 85748 Garching, Germany
{sabine.buckl,matthes,christian.m.schweda}@mytum.de
http://wwwmatthes.in.tum.de

Abstract. Enterprises are unique in the way of doing business. This uniqueness is typically reflected in the overall make up of the enterprise – the enterprise architecture (EA). Globalized markets, changing legal regulations, and technological innovations thereby force enterprises to continually adapt their EA to the changing environment. As response, enterprises aim at a strategic management of the EA providing a holistic model of the key elements and relationships of an enterprise. Different supporting modeling languages have been proposed but none of them has gained broad acceptance due to the above described uniqueness.
In this paper we present a method to develop organization-specific EA modeling languages based on building blocks representing practice-proven solutions. Following the common understanding of modeling languages as consisting of syntax, semantics, and notation, we provide three different types of building blocks: information model building blocks that specify the syntax, glossary building blocks that textually define semantics, and viewpoint building blocks that specify the notation of the language. The applicability of the method for integrating building blocks to a consistent EA modeling language is illustrated along a case study from the public sector. The exposition of the method concludes with an outlook on further areas of research.

1 Motivation and overview

Enterprise architecture (EA) management is a discipline, which has recently gained increased attention from academia and practice. Thereby, a few topics which are nowadays regarded to be part of EA management, have a long history in information systems research. This can be exemplified with the topic of business-IT-alignment discussed e.g. by Henderson and Venkatraman in the late nineties as strategic alignment [1]. While these discussions might have catalyzed the evolution of EA management, the overall discipline is still subject to ongoing development. This in particular applies as different research communities continue to argue on the perspective, from which EA management should be approached (cf. discussions by Frank [2] or Wegman [3]). The approaches

nevertheless agree that EA management needs to provide a holistic view on an enterprise, accounting for aspects from all layers, ranging from business to IT.

Independent from the question of perspective, other indications for the ongoing development of the EA management discipline exist. EA modeling represents a prominent example. Although most EA management approaches emphasize on the importance of modeling the EA, no common metamodel, called **information model** in accordance with Buckl et al. in [4], has yet been established. In the last years, many information models were proposed but none of them has gained broad acceptance. Some researchers even challenge the hypothesis that such a model exists (cf. [5, 6]). They expect enterprises to have largely different expectations on the benefits of EA management, and therefore assume that an information model is an enterprise-specific artifact. In Section 2 we discuss possible reasons for this kind of specificity, when we revisit how today's EA modeling languages position themselves as tools for supporting EA management functions. In [5] Buckl et al. discuss three different ways for developing organization-specific EA modeling languages, of which two approaches, namely the *customization approach* and the *integration approach* try to leverage established best-practices. Understanding that most of today's EA and EA management frameworks pursue a customization-based approach, we discuss the shortcomings of such approach which motivates the research objective of this article:

> Introduce a method for developing EA information models based on composeable best-practice solutions for defining such languages, and show the applicability of the method.

This objective is approached in Section 3, where we outline our method for developing organization-specific EA modeling languages, which is based on reusable **building block**s for such languages. The method is based on a specific understanding of EA modeling languages, according to which the language's syntax is specified in an information model, the notation is defined via **notation function**s as well as **representation function**s, and the semantics is defined textually in a **glossary**. For any of these constituents, the presented method supplies specific building blocks, which are further composed into a comprehensive EA modeling language. The applicability of the method is exemplified in Section 4. Final Section 5 summarizes the article's findings and gives an outlook on further research to follow in the field.

2 EA modeling: theoretic foundations and state-of-the-art

A plethora of different EA management approaches has been proposed in the past that typically either focus on a language to model the EA or on a method how to document, analyze, and communicate EA-related aspects. This section provides an overview on selected EA management approaches with particular focus on the proposed EA modeling language(s). Preparing the analysis of existing EA modeling languages, we establish a theoretic foundations and revisit a conceptual framework for EA design as discussed by Buckl et al. in [7]. Central

to this framework is the understanding that EA modeling has both an intensional and an extensional nature. Extensionally, languages are used to reflect certain architectural properties, i.e. phenomena, whereas the intensional nature of a language reflects the fact that the corresponding models are created for 'doing something'. Understanding EA management as a design process, i.e. as engineering process targeting the transformation of the enterprise, Buckl et al. apply in [7] the propositional framework for design of Simon [8]. This framework promotes a formal understanding of the activity of design, based on the central notion of *means-end*-relationships. Any design activity is carried out in the light of domain-inherent characteristics and relationships ("natural laws" [8]) that connect dedicated means to corresponding ends. This conversely means that a designer with a specific end (goal) in mind searches for means by which the design artifact will achieve those ends. For the context of EA management, this means that enterprise architects "given the constraints and fixed parameters, find values for the command variables that satisfy the utility function" [8]. Building on this, Buckl et al. explored in [7] how the constituents of EA-specific design propositions look alike and devised a mapping distinguishing between:

- the **current** and **future** make-up of the EA (command variables),
- the **strategies & projects** affecting and changing the EA (means),
- the **principles & standards** guiding the evolution of the EA (constraints),
- the **visions & goals** describing a target state of the EA (ends), and
- the **KPIs & metrics** measuring and evaluating an EA state (utility function).

Above distinction pertains to the EA modeling language, more precisely to the language constituent of the syntax that introduces the primitives used in an EA model. In line with Ernst et al. [9], we assume that the language syntax is represented in an **EA information model**, containing the classes, properties, and associations used to describe a particular **area-of-interest** in the EA. These areas-of-interest can be distinguished to extensional ones (**concerns**), i.e. ones covering the command variables, and intensional ones, covering **cross-cutting** aspects, as strategies, goals, standards, or metrics. This distinction is to be kept in mind with respect to the review of the state-of-the-art undertaken in this section, but also for devising our development method in Section 3. The distinction gives rise to a particular conception of the EA modeling language(s) used to support a particular EA management approach.

TOGAF proposes to structure the EA in different architecture domains representing subsets of the overall EA [10]. TOGAF distinguishes between

- *business architecture* concerned with strategic, governmental, organizational, and process-related aspects,
- *data architecture* describing the structure of an organization's data assets and data management resources,
- *application architecture* considering the application systems, their interactions, and their relationships to the business processes, and
- *technology architecture* describing the software and hardware capabilities required to support business, data, and application services.

Answering the question which elements should be considered in an EA management endeavor, TOGAF presents the *core content metamodel*. Therein, the entities and relationships that make up an EA are described [10]. The core content metamodel provides entities for all architectural layers. Besides the entities, which can be grouped to one of the architectural layers, TOGAF also introduces crosscutting entities associated with all objects among others principle, requirement, work package. Thereby, the kind of relationship is not discussed. TOGAF provides six metamodel extensions [10], e.g the *motivation extension* to enable measurement of business performance by introducing concepts as driver, goal, and objective. While each of these extensions supplies concepts for modeling and described the intended usage context, these concepts are not formulated as cross-cutting aspects pertaining to arbitrary architecture elements.

In the 1970s and the 1980s several EA-related frameworks have been developed. In response to the emerging number of frameworks in this area, the *International Task Force on Enterprise Integration* was established aiming at the development of a reference framework that supports comparison and evaluation of existing approaches [11]. As a result of the investigation, the Task Force developed the *Generalised Enterprise Reference Architecture and Methodology* (GERAM). GERAM consists of nine components, of which with respect to the EA modeling language, three components are of interest. These do not impose particular languages but define criteria for an EA management approach [12]:

- *Generalised Enterprise Reference Architecture* (GERA): GERA describes the basic concepts to be used in enterprise engineering and integration projects. According to GERAM these concepts can be categorized as human-oriented concepts, process-oriented concepts, and technology-oriented concepts.
- *Enterprise Modeling Languages* (EMLs): EMLs define the generic modeling constructs for enterprise modeling. In particular, the EMLs provide constructs to describe and model human roles, operational processes, supporting information, and technologies.
- *Generic Enterprise Modeling Concepts* (GEMCs): GEMCs define and formalize the generic concepts of enterprise modeling. The following ways of formalization exists: natural language explanations (*glossaries*), meta models describing the elements and their relationships (*information models*), and theories defining the semantics of enterprise modeling languages (*ontologies*).

GERA defines a life-cycle for each constituting concept of the enterprise, which consists of the phases *identification*, *concept*, *requirements*, *(preliminary and detailed) design*, *implementation*, *operation*, and *decommission*. While most of the aforementioned phases are self-explanatory, the *concept* phase deserves a more in depth analysis with respect to our analysis framework. The phase is concerned with the definition of the entity's mission, vision, strategies, objectives, etc. [12]. Thus, linking the cross-cutting aspects of strategies, projects, visions, and goals to any concept considered during enterprise transformation. In line with the objective of GERAM to define requirements for EA (management) frameworks, no description how this relation should be conceptualized is given. Similarly, the concept of *life history* is discussed and the link to different kind of projects is explored and related to the phases of the EA concepts.

In addition, to the generalized propositions for a language for EA descriptions as discussed above, the EMLs define two requirements to enable integration of

special purpose modeling languages (cf. [12]). First, every area as represented in the modeling framework must be covered for every enterprise entity type, and second, any model developed must be able to be integrated with models of other subject areas, if the information content of the model requires integration. The need to integrate different languages results from the distinct 'expressive powers' related to the intended purpose, e.g. description vs. analysis, of the languages.

Against the background of over 15 years of practice, Dietz [13] has developed a "methodology for (re)designing and (re)engineering organizations" called DEMO. With its sound theoretical foundation in a theory called Ψ-theory the method takes a different perspective on the enterprise focusing on the so-called "enterprise ontology". Dietz uses this term to denote a "coherent, comprehensive, consistent and concise model of the essence of the enterprise". Critical to his definition is thereby the notion of "essence" that in the sense of Dietz targets the deep behavioral nature of the enterprise, but not realization and implementation specific details. The method of DEMO provides an approach to develop enterprise ontologies in a systematic way [13], i.e. reflects commitment-related information. In line with the four basic axioms of Ψ-theory the ontological model of the enterprise is constituted of four distinct submodels (*construction model*, *state model*, *process model*, and *action model*) The *construction model* specifies the construction of the organization embodied in the identified transaction types as well as actor roles. Detailing the coordination aspect of the transactions, the *process model* describes causal and conditional relationships between different transactions. Complementing this perspective the *state model* outlines the state space of P-facts, i.e. of production results, while P-acts are not part of the state model, as they may be derived from the corresponding process model. The *action model* describes the enterprise ontology on the most detailed model, such that – as Dietz states in [13] – the other models may be derived from the action model, and are hence only provided for ease of use. The different abstracted models (construction, process and state model) are complemented each with a specific description language, of which especially the language behind the state model deserves special attention. The so called "world ontology specification language" (WOSL) (cf. Dietz [14]) provides the basic language elements for describing *rigid* and *non-rigid* structures, i.e. states that exist universally over time and states that may change. The construction and process model languages present themselves as two sides of the same coin taking a blackbox and a whitebox perspective on the organizational transactions further mirrored in the prescriptive understanding of an EA complementing the enterprise ontology with "functional" and "constructional principles" [13].

3 Designing an EA modeling language

Different EA management problems call for a distinct understanding of the EA and hence entail a different way of modeling architectural properties. For each such problem, the corresponding understanding of the problem domain, i.e. the relevant part of the EA, can be expressed in a specific EA modeling language.

This language is *domain appropriate* in terms of Krogstie [15], i.e. constrains itself only to relevant syntax, semantics, and notation. For developing one or more EA modeling languages, e.g. representing different problems, we have to design the corresponding syntax, semantics, and notation. Prior to introducing our design method, we provide a conceptual model that describes the interplay of these constituents. Central thereto is the **information model**, which in line with our considerations in [9], specifies the syntax of the modeling language via MODEL ELEMENTs. Semantics is considered as a function assigning exactly one **predicator** [16] to each MODEL ELEMENT. This predicator acts as surrogate for a corresponding part of the conceptualization, i.e. for an *architecture property* in the sense of Dijkman et al. [17]. Such property represents a characteristic of the architecture relevant in respect to one or more architecture stakeholders. Complementing the notation is described as a representation function assigning one VISUALIZATION ELEMENT[1] to each MODEL ELEMENT. In consequence, a predicator (or the thereby represented architectural property) is assigned a particular visualization element by the modeling language.

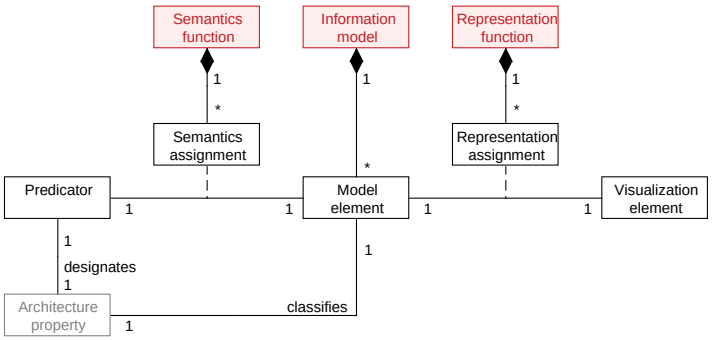


Fig. 1. General constituents of an EA modeling language

Different languages in turn can cover different sets of predicators and can assign different visualization elements thereto. Figure 1 depicts the conceptual model for modeling languages utilizing aforementioned terms. The architecture property is therein shown in gray, as it represents a purely intrapersonal concept, whereas the three basic constituents of the language syntax (INFORMATION MODEL), semantics (SEMANTICS FUNCTION), and notation (REPRESENTATION FUNCTION) are highlighted.

Combining different modeling languages in a consistent manner builds on the notion of consistency as introduced by Dijkman in [18]. In particular, the question of integrating the underlying information models is of relevance, as these can contain different MODEL ELEMENTs reflecting the same architecture property. Such MODEL ELEMENTs originate from different ways of perceiving the

[1] In line with Kamlah and Lorenzen [16] arbitrary *speech acts* can notate an element. With the focus of the domain, we constrain our subsequent considerations to visual elements.

underlying property. For the context of the method, the predicator, i.e. the glossary entry, identifies the corresponding property. Two information models can be interlinked by three different types of relationships, namely **overlap**, **subsume**, and **conflict**. Moving from the instance-level perspective on relationships as taken by Dijkman in [18] to an understanding on a conceptual level, we use the predicators and define the basic relationship overlap as expressing that two information models share at least one predicator. Building on this, the two other types of relationships are defined as follows:

- Two information models conflict with each other, if they overlap, but make contradictory statements with respect to the MODEL ELEMENTs assigned to overlapping predicators.
- One information model subsumes another one, if the subsuming model is completely overlapped by the subsumed one, i.e. if all predicators of the subsuming information model are also present in its subsumed counterpart.

Subsume and **conflict** are closely related to each other in the sense that for two conflicting information models it is not possible to find or create a third information model subsuming both. Contrariwise, a subsumed information model completely overlaps the subsuming one without raising conflicts. The types of relationships between information models provide a basis for our development method in a threefold manner. Firstly, the users can be prevented from selecting conflicting information models to be integrated into the comprehensive information model supporting their EA management function. Secondly, existing overlaps between the selected information models can be accounted for during integration of the information models. Finally, the subsume-relationship can be used to derive possible paths for evolving the information model.

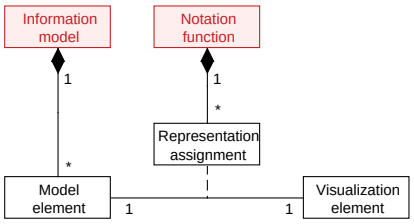


Fig. 2. Notation function

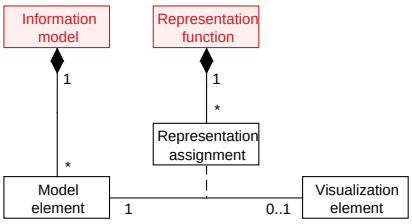


Fig. 3. Representation function

Linking information models to visualizations, we have to define two types of functions, namely the **notation function** and the **representation function**. For each modeling language the notation function establishes a bijective mapping between MODEL ELEMENTs and VISUALIZATION ELEMENTs. The representation function of other modeling languages contrariwise does not establish one-to-one relationships, but can provide an aggregated perspective on the information model elements, e.g. by abstracting relationships or calculating sums. Put in terms of our conceptual model for the constituents of an EA modeling

language, we can describe the general distinction between notation functions and representation functions via the existence of representation assignments as shown in Figures 2 and 3, respectively.

The solutions provided by the different approaches to EA management, discussed in Section 2, provide partial EA modeling languages, which are documented on different levels of abstraction, giving several specifics of the application, e.g. the employed terminology. EA management approaches with an embracing appeal tend to mitigate specifics of a singular prescription and to present the solutions in a *stratified* terminology, thereby increasing readability and comparability. In this sense the assignments between the PREDICATORs and the MODEL ELEMENTs as contained in different EA modeling languages of the approach are adapted. Any EA modeling language can supply at most one MODEL ELEMENT assigned to a particular PREDICATOR from the stratified terminology of the approach. Nevertheless, different languages can bring along elements assigned to the same predicator. Figure 4 displays the conceptual model bringing together different EA modeling languages backed in a consistent terminology.

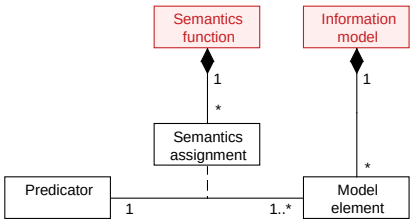


Fig. 4. Semantics assignments in a set of EA modeling languages

Our development method has to target the three aspects of EA modeling languages with corresponding building blocks. These building blocks have been proven to work in practice [19]. In particular *information model building block* (IBB) supplying an information model, *viewpoint building block* (VBB) supplying a representation function, and *glossary building block* (GBB) supplying a semantics function. These building blocks are complemented with techniques facilitating their integration and combination. Using the techniques and building blocks of the different types a consistent set of EA modeling languages can be developed. Figure 5 shows the interplay of the different practice-proven building blocks complementing the development method.

With respect to IBBs, our method offers an additional distinction between MODEL ELEMENTs that reflect architecture elements, representing the extensional nature, and those that reflect goals of EA management, representing the intentional nature. Revisiting the analysis of EA modeling languages from Section 2, we understand goals as particular instantiation of a more general principle of describing EAs. Similar to goals, also projects or standards raise certain characteristics that do not supply an *identity condition* (IC) for the corresponding model elements. Put in other words, while goals, projects, and standards them-

selves supply an IC, they relate to dependent model elements that do not supply an IC, are *dispersive* types [20]. Based on this analysis, we establish a distinction between two types of IBBs: one building around identifiable architecture elements (**concern IBB**s) and one centering around cross-cutting architecture characteristics (**cross-cutting IBB**s). These concepts are further summarized under the term **area-of-interest**.

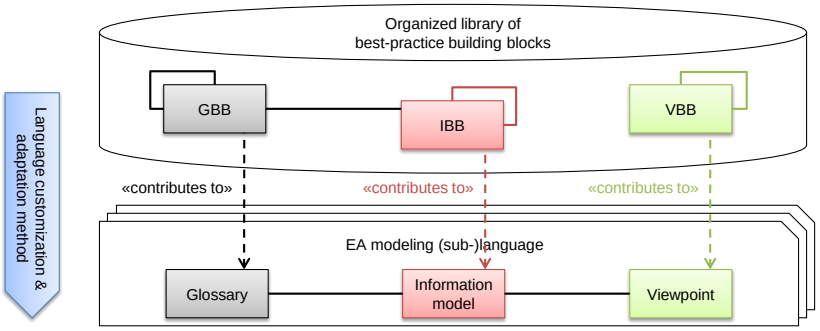


Fig. 5. Interplay of language building blocks

An IBB defines a part of the syntax of an EA modeling language by specifying the corresponding MODEL ELEMENTs, i.e. the TYPEs, RELATIONSHIPs, and PROPERTYs, that make up the syntax. In developing a specific information model, cross-cutting IBBs corresponding to goals and questions, projects, or standards are applied, i.e. added, to concern IBBs. Via supplied semantics assignments, the model elements are linked to predicators, i.e. classifying terms from the glossary, which in turn reflect a particular architecture property. The different values that an architecture property can take are codified into instantiations of the corresponding MODEL ELEMENT. If this concept, for example, is a PROPERTY[2] these instances can be identified with the range of the corresponding data type, e.g. INTEGER or STRING. Similar considerations also apply for TYPEs and RELATIONSHIPs. With the IBBs specifying re-usable information models bound to a consistent underlying terminology, the relationships applying on information models can also be applied to interrelate IBBs. The IBB-relationships are of interest in extending the method's underlying organized collection of IBBs as exemplified in Figure 6. Each of these relationships is therein set effective by at least one predicator.

In order to ensure usability of the method in general and the techniques for selecting and integrating the IBBs in special, any newly added IBB has to be embedded into the net of IBB-relationships, if overlapping. Forcing the contributors of IBBs to establish these relationships manually would aggravate any extension of the organized library of building blocks, exponentially growing with the number of already supplied IBBs. Resolving the aforementioned difficulty, we

[2] We use the different typeset to avoid confusion with the architectural property.

decide to derive these relationships from the relationships between predicators and model elements as well as the containment of model elements in the IBBs.

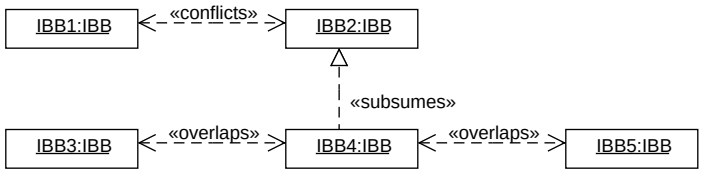
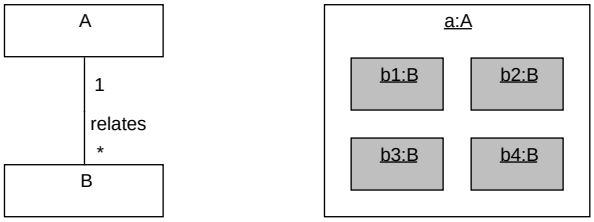


Fig. 6. Exemplary net of IBB-relationships

Another important aspect of developing EA modeling languages from practice-proven solutions is the notational aspect. Any language brings along symbolic representations for the MODEL ELEMENTs and their relationships, which must in turn match the specific notational expectations of the corresponding users. With respect to the method, this means that the users must be able to select proven-practice notations and adopt them to fit their particular EA modeling languages. The VBBs provide such proven-practice representation assignments as derived from the analyzed EA management approaches. These assignments are of abstract nature, i.e. do not relate to a particular underlying information model, but to an abstract conception of the information to be represented. Latter forms the **viewmodel**[3], which can be understood as a 'virtual' information model, acting as domain for a representation function. Revisiting the exemplary visualization shown in this section, we explain the notion of the viewmodel along the example of the *clustered visualization*, which according to Wittenburg [21] is frequently used in representing EA information[4].



A clustered visualization depicts instances of an outer TYPE A and instances of an inner TYPE B, which are related via some RELATIONSHIP.

The viewmodel supplies enough information to create the visualization but not more, such that the representation assignments constitute a bijective representation function. It would nevertheless be possible to create the visualization

[3] A similar concept is discussed by Fowler in `http://martinfowler.com/eaaDev/PresentationModel.html` (cited 2010-12-23).

[4] Wittenburg calls this type of visualization "cluster map" [21, page 78–79].

in cases, where additional information was supplied. In mathematical terms this means that any particular kind of visualization, i.e. any VBB, can be applied on any information model capable of supplying the **sufficient information**. For being in turn able to perform graphical modeling, the underlying information model must not supply additional information, i.e. has to supply only the minimal **necessary information**. Such information model is closely related to the viewmodel of the corresponding VBB. These considerations take a different point of view on what differentiates notation functions and graphical representation functions. While the former relate visualizations to their necessary and sufficient information model (the viewmodel), latter functions can build equivalently on any sufficient information model. Being sufficient relates to the concept of the subsume-relationship introduced above. Any information model that is subsumed by visualization's necessary and sufficient information model is itself sufficient to create the visualization. Understanding the subsume-relationship as a basis for evolving the information model of an organization, we can ensure that any subsumed information model is still sufficient to create the once selected graphical representations. Aforementioned indications give rise to the following definition of VBBs. In line with our considerations in [22] we detail on the contribution that a VBB makes with respect to specifying a viewpoint. According to the provided contribution, we distinguish different types of VBBs:

- **Symbol VBB**s that assign a mapping between a MODEL ELEMENT and a visible VISUALIZATION ELEMENT, i.e. a symbol in terms of [9].
- **Structural VBB**s that assign a complex structure of MODEL ELEMENTs to a set of interrelated VISUALIZATION ELEMENTs. Such building blocks are mostly not self-contained but reference other VBBs as sub-assignments.
- **Decorating VBB**s that assign a mapping between a MODEL ELEMENT and a VISUAL PROPERTY of a VISUALIZATION ELEMENT, which in turn results from a mapping specified in a different VBB.
- **Hybrid VBB**s that specify to some extent a combination of the above. Such VBBs can be constituted from other VBBs, e.g. a structural VBB referencing an elementary VBB to which additionally a decorating VBB is applied.

Any viewpoint is configured in terms of at least one structural VBB that determines the overall make-up of the corresponding visualization. This understanding aligns with the principle of the *base map* as established by Wittenburg in [21] used to describe the basic nature of the analyzed *software maps*. It is nevertheless not necessary for a viewpoint to build on an isolated structural VBB, as also a hybrid VBB containing at least one structural VBB can serve as **base VBB** for a viewpoint. Finally, the GBBs are used to provide a consistent underlying understanding of the concepts employed in the IBBs. Central thereto is the notion of the predicator, which is complemented with a textual description of the according semantics. This description can in turn link to related predicators, although the relationships established thereby are not typed, but give indications on a corresponding connectedness.

Complementing above considerations on the building blocks and their interrelationships to each other, we subsequently outline how the organized library

of building blocks is used by the development method proposed in this paper. In general building block-based development of EA modeling languages consists of three distinct activities, namely **information modeling**, **viewpoint definition**, and **glossary adaptation**. Latter activity is elemental in its nature, meaning that the users of the method browse through an existing glossary and rename one or more of its entries. Thereby, the corresponding predicator is shadowed with an organization-specific predicator, that consistently inherits all semantic assignments of its underlying predicator. Information modeling starts with an empty information model and iteratively applies the following four steps:

1. Select EA management-relevant goal to pursue and choose cross-cutting IBB reflecting the goal on an adequate level of abstraction.
2. Select EA concern on which the goal applies and choose admissible concern IBB reflecting the concern on an adequate level of abstraction.
3. Integrate the cross-cutting IBB with the concern IBB to build a problem-specific information model by specifying the goal-relevant MODEL ELEMENTS.
4. Integrate the problem-specific information model with the already existing information model. (omitted in first iteration)

After above steps have been executed once, an information model is maintained by the method in the organization-specific configuration. This information model is used to determine, whether a concern IBB is admissible or not. The latter is the case, when the IBB conflicts with an already integrated one. Another subtlety applies to this step, as the concern descriptions change with the adaptation of the glossary, thus ensuring that the users perform the selection based on their adapted terminology. The activity of viewpoint definition is closely interrelated to the one defining the EA management function in general. The method part of the function specifies particular tasks, in which actors in EA management and EA stakeholders have to be informed or must provide information on certain architectural aspects. With the viewpoints, in line with Krogstie [15], being the vehicles for externalizing, comprehending, and communicating information, any relationship between an actor or stakeholder and a task has to be supplied with a specific viewpoint. Thereto, the following two step approach is applied:

1. Select base VBB and establish admissible links from its virtual information model to the model of available information.
2. Repeat: add VBB to detail existing viewpoint and establish admissible links from the corresponding virtual information model to the model of available information.

Two conceptions in the former approach deserve special attention. Firstly, it can be the case that in some tasks of the EA management function not all information according to the integrated information model is available. This particularly applies for documentation-related tasks, such that any VBB applied to define a viewpoint is confined to the information actually available. Put in other words, the VBB has to be configured against the model of available information. Secondly, while there are in general only a few restrictions on the VBB to apply for a specific task, the configuration of the corresponding viewpoint is restricted with respect to the intended usage thereof. Having selected a task-actor-relationship, according to which the actor has to provide information about an EA concern,

the configured viewpoint must supply representation assignments that constitute a notation function for the information under consideration. Put in other words, the actors must have the possibility to model the corresponding part of the enterprise using the established modeling language. In this sense, any mapping configuration is analyzed by means of the provided techniques with respect to its suitability to maintain a bijective relationship with the underlying information model, i.e. if this information model is isomorphic to the VBB's viewmodel.

4 Exemplifying the design approach

A public authority providing IT services to several federal ministries has over the years 'grown' a highly heterogeneous application landscape, causing high maintenance costs and requiring a diverse set of IT operating skills. In order to address this IT-related problem, the authority decides to introduce an EA management function specifically pursuing the goal of *standardization* on the level of *business applications* and their underlying *technologies*. Having chosen goal and concern, two IBBs become available, one describing how to operationalize standardization, another defining the concepts needed to describe the relationship between applications and technologies. Figure 7 depicts the information model derived from these IBBs via integration.

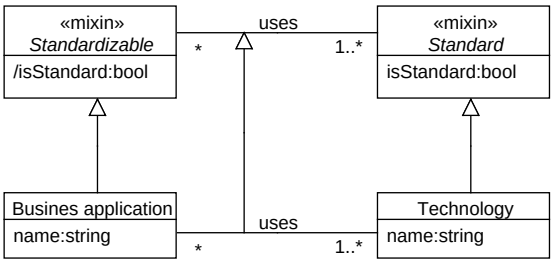


Fig. 7. Information model

The complementing GBBs for the information model elements, introduce definitions. The public authority decides to apply the definitions without adaptations. Finally, the authority designs a viewpoint for communicating the information about standardization. The viewpoint builds on a basic cluster-VBB extended with a VBB for color-coding, according to which non-standard technologies are colored red, whereas standard technologies are colored green.

5 Conclusion and outlook

Balancing organization specificity on the one hand and the demand for a general method to model EAs, we proposed in this paper a method to develop

organization-specific EA modeling languages based on practice-proven building blocks. To enable organization-specific configuration, three different types of building blocks have been proposed. Information model building blocks describing the syntax of the language, i.e. the concepts that make up the EA, glossary building blocks that specify the semantics of the concepts reflecting the organization-specific terminology, and viewpoint building blocks providing a stakeholder-specific notation for visualizing EA-related information. While the different building blocks enable flexible configuration to an organization-specific EA modeling language, aspects of consistency have to be accounted for. In presenting our development method we discussed how consistency can be ensured.

While the utilization of the development method in a case study with an industry partner from the public sector provides first indications for the applicability and utility of the presented method, further case studies should be conducted. Furthermore, a tool supporting the user of the development method in conducting the single steps of the method can be regarded useful. In particular as such a tool could be used as a configurator for that would enable initial design as well as adaptation of EA modeling languages thus supporting the enterprise architects in adapting to changing environmental influences and problems to be addressed. The resulting configuration could further be used as input for dedicated EA management tools to facilitate the customization thereof.

References

1. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Systems Journal **32**(1) (1993) 472–484
2. Frank, U.: Multi-perspective enterprise modeling (memo) – conceptual framework and modeling languages. In: 35th Hawaii International Conference on System Sciences (HICSS 2002), Washington, DC, USA (2002) 1258–1267
3. Wegmann, A.: On the systemic enterprise architecture methodology (seam). In: SEAM). Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003. (2003) 483–490
4. Buckl, S., Ernst, A.M., Lankes, J., Matthes, F., Schweda, C.M., Wittenburg, A.: Generating visualizations of enterprise architectures using model transformation (extended version). Enterprise Modelling and Information Systems Architectures – An International Journal **2**(2) (2007) 3–13
5. Buckl, S., Ernst, A.M., Lankes, J., Schneider, K., Schweda, C.M.: A pattern based approach for constructing enterprise architecture management information models. In Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler, eds.: Wirtschaftsinformatik 2007, Karlsruhe, Germany, Universitätsverlag Karlsruhe (2007) 145–162
6. Kurpjuweit, S., Winter, R.: Viewpoint-based meta model engineering. In Reichert, M., Strecker, S., Turowski, K., eds.: 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007). LNI, Bonn, Germany, Gesellschaft für Informatik (2007) 143–161
7. Buckl, S., Matthes, F., Roth, S., Schulz, C., Schweda, C.M.: A conceptual framework for enterprise architecture design. In Aalst, W., Mylopoulos, J., Sadeh, N.M.,

Shaw, M.J., Szyperski, C., Proper, E., Lankhorst, M.M., Schönherr, M., Barjis, J., Overbeek, S., eds.: Trends in Enterprise Architecture Research. Volume 70 of Lecture Notes in Business Information Processing., Springer Berlin Heidelberg (2010) 44–56
8. Simon, H.A.: The Sciences of the Artificial. 3rd edn. MIT Press, Cambridge, Massachusetts, USA (1996)
9. Ernst, A.M., Lankes, J., Schweda, C.M., Wittenburg, A.: Using model transformation for generating visualizations from repository contents – an application to software cartography. Technical report, Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany (2006)
10. The Open Group: TOGAF "Enterprise Edition" Version 9. `http://www.togaf.org` (cited 2010-02-25) (2009)
11. Bernus, P., Nemes, L., Schmidt, G.: Handbook on Enterprise Architecture. Springer, Berlin, Heidelberg, Germany (2003)
12. IFIP-IFAC Task Force on Architecture for Enterprise Integration: Geram: The generalised enterprise reference architecture and methodology. In Bernus, P., Nemes, L., Schmidt, G., eds.: Handbook on Enterprise Architecture, Berlin, Heidelberg, Germany, Springer (2003) 21–63
13. Dietz, J.L.: Enterprise Ontology. Springer, Heidelberg, Germany (2006)
14. Dietz, J.L.: A world ontology specification language. In: On the Move to Meaningful Internet Systems 2005: OTM Workshops. (2005) 688–699
15. Krogstie, J.: A semiotic approach to quality in requirements specifications. In: Proceedings of the IFIP TC8 / WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of Information Systems, Deventer, The Netherlands, The Netherlands, Kluwer, B.V. (2002) 231–249
16. Kamlah, W., Lorenzen, P.: Logische Propädeutik: Vorschule des vernünftigen Redens. 2nd edn. Metzler, Stuttgart, Germany (1967)
17. Dijkman, R.M., Quartel, D.A., van Sinderen, M.J.: Consistency in multi-viewpoint design of enterprise information systems. Information and Software Technology **50**(7–8) (2008) 737 – 752
18. Dijkman, R.M.: Consistency in multi-viewpoint architectural design. PhD thesis, Enschede (2006)
19. Buckl, S., Matthes, F., Schweda, C.M.: Utilizing patterns in developing design theories. In: 2010 International Conference on Information Systems (ICIS 2010). (2010)
20. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, CTIT, Centre for Telematics and Information Technology, Enschede, The Netherlands (2005)
21. Wittenburg, A.: Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften. PhD thesis, Fakultät für Informatik, Technische Universität München, Germany (2007)
22. Buckl, S., Gulden, J., Schweda, C.M.: Supporting ad hoc analyses on enterprise models. In: 4th International Workshop on Enterprise Modelling and Information Systems Architectures. (2010)