Towards a Conceptual Framework for Interactive Enterprise Architecture Management Visualizations

Michael Schaub, Florian Matthes, Sascha Roth
{michael.schaub | matthes | sascha.roth}@in.tum.de

Abstract: Visualizations have grown to a de-facto standard as means for decision-making in the management discipline of enterprise architecture (EA). Thereby, those visualizations are often created manually, so that they get soon outdated since underlying data change on a frequent basis. As a consequence, EA management tools require mechanisms to generate visualizations. In this vein, a major challenge is to adapt common EA visualizations to an organization-specific metamodel. At the same time, end-users want to interact with the visualization in terms of changing data immediately within the visualization for the strategic planning of an EA. As of today, there is no standard, framework, or reference model for the generation of such an interactive EA visualization.

This paper 1) introduces a framework, i.e. an interplay of different models to realize interactive visualizations, 2) outlines requirements for interactive EA management visualizations referring to concepts of the framework, 3) applies the framework to a prototypical implementation detailing the therein used models as an example, and 4) compares the prototype to related work employing the framework.

1 Introduction

Today's enterprises cope with the complexity of changes to highly interconnected business applications whereas local changes often result in global consequences, i.e. impact the application landscape as a whole. At the same time, change requests to business applications or processes are required to be fast and cost-effective in response to competitive global markets with frequently changing conditions[WR09, Ros03]. Enterprise Architecture (EA) management promises to balance between short time business benefit and long term maintenance of both business and IT in an enterprise [MWF08, MBF+11]. Thereby, having a holistic perspective of the EA is indispensable. In this vein, visualizations have grown to a de-facto standard as means for strategic decision making in the management discipline of EA. Concepts formally describing EA management visualizations are summarized as system cartography[1], whereby the generation of visualizations out of existing data is not yet described in depth [Wit07], i.e. currently there exists no standard, framework, reference architecture, or best-practice for generating EA visualizations.

Slightly later than the discipline itself, also tool support for EA management emerged [MBLS08, BBDF+11]. With respect to their visualization capabilities, the range of tools for EA management reaches from mere drawing tools to a model-driven generation of vi-

[1] Formerly known as software cartography [Mat08].

sualizations. The former approach has clear drawbacks since visualizations are created manually in a handcrafted, error-prone, and inefficient process. The later approach is often limited to a single information model aka metamodel. Thereby, such an information model has to try to capture the entirety of all relevant entities across all business domains and industry sectors. Since this is an endeavour doomed to fail, EA vendors chose to offer mechanisms for extending a 'core' information model. Since no standard information model for EAs exists, enterprises tend to use an organization-specific information model reflecting their information demands and tend to adopting the enterprise's terminology, i.e. aforementioned extension mechanisms are frequently used. At the same time, respective visualization algorithms do not adapt to those changes automatically, i.e. the visualizations have to be adapted to the extensions leading to extensive configuration or additional implementation/customization efforts. Since there is no standard, framework, or reference model for generating such an interactive EA visualization, we conclude with the following research question:

> 'How does a common framework or reference model for generating interactive EA visualizations look like?'

The remainder of this chapter is structured as follows: Section 2 introduces a conceptual framework for generating interactive visualizations in general and in particular for EA management. An outline of requirements for interactive EA management visualizations is given in Section 3. The framework is then applied to a prototypical implementation in Section 4. Subsequently, Section 5 revisits related approaches and compares them to the prototype employing the introduced framework. Finally, Section 6 concludes this paper and gives a brief outlook on open research questions.

2 Generating interactive visualizations

Figure 1 illustrates an overview of a conceptual framework to generate interactive visualizations that is detailed in the following. The framework consists of:

A **data model** which is considered as the actual data d within a data source that can be retrieved by a query q. Depending on the nature of the data source, different fields may have different access permissions [BMR+10, BMM+11]. Therefore, a **data interaction model** d_i captures the different access permissions for each concrete $x \in d$, i.e. access rights and permissions on data level but not schema level. As an example users of a certain department might only get information about business applications in their particular business unit. An **information model** that describes the schema i_m that the data model d is based on. "An information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse" [Lee99]. An **interaction model** i that subsumes the interactions that are allowed upon the information model level, i.e. which entity can be created, read, updated, or deleted. For instance, a certain role can only create business applications but is not allowed to create business units. An **abstract information model** which can be a template for a certain information model or type/entity therein. Based on the observations of Buckl et al. in [BEL+07], organizations use recurring patterns to describe their managed information. Especially in [Sch11], Schweda shows that recurring patterns of information models

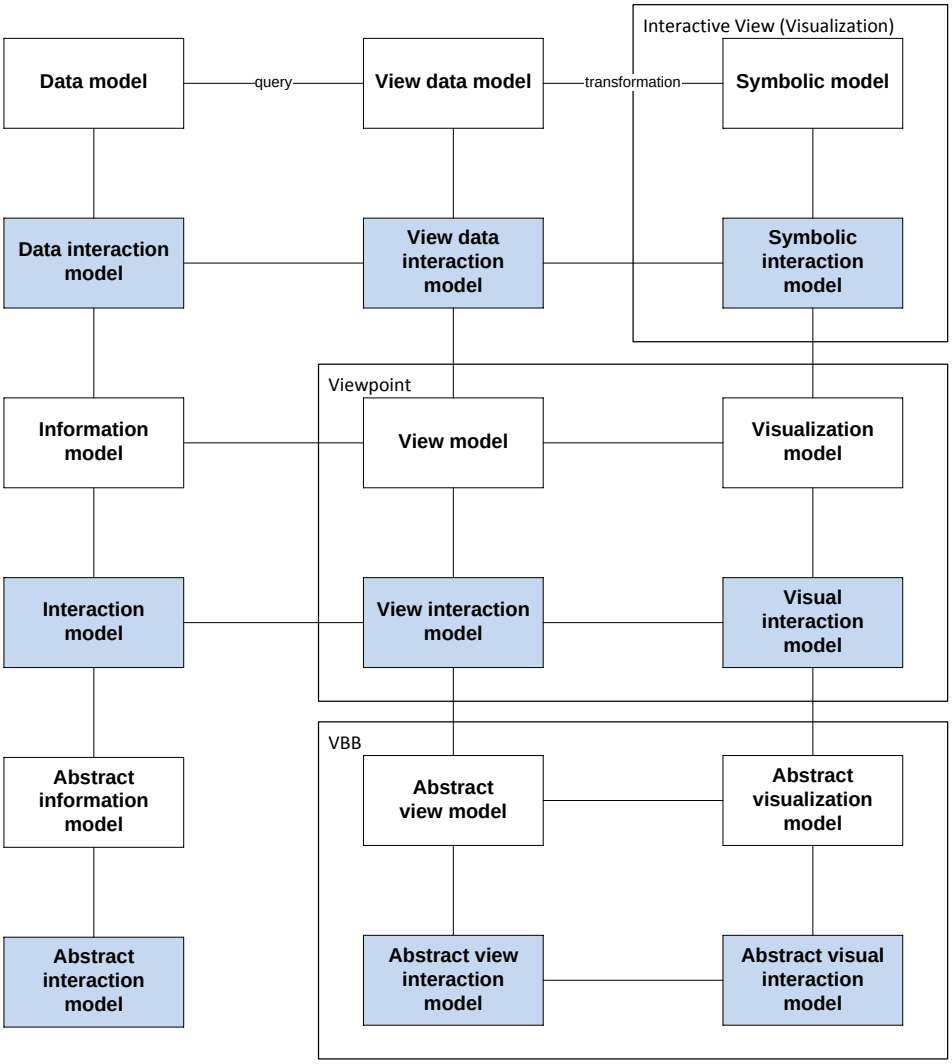


Figure 1: A conceptual framework to generate interactive visualizations

have been observed which he synthesized to so-called information model building blocks (IBBs). Such an information model template, fragment, or building block comes with a certain **abstract interaction model** that describes e.g. predefined access rights synthesized as best practices.

A **view data model** $v = q(d)$ such that $v \subseteq d \cup q_1$, whereby q_1 are results of q that are calculated out of d, e.g. aggregations or average values. A **view data interaction model** $\subseteq d_i$ which is derived from q. In some cases, q reduces d_i not only by the selected values of d, but also additional interactions, e.g. aggregated values cannot be edited regardless of access rights for a specific $x \in d$. A **view model** is the schema v_m of v, such that $v_m \subseteq i_m \cup q_2$ is derived from q, whereas q_2 describes the part of the schema, which has been created entirely by q, i.e. in general $q_2 \nsubseteq i_m$. A **view interaction model** $v_i \subseteq i$ which is determined by q, i.e. depending on a particular q interactions of i are enabled or not by v_i. For instance, on aggregated values, updates are prohibited, whereas relationships and transitive relationships[2] could be updated[3] based on i. An **abstract view model** which defines the information demands for a particular visualization blueprint. The abstract view model v_a can be used as a basis to perform a pattern matching, i.e. matching for the pattern given by v_a on i_m (see e.g. [BURV11, BHR+10a]). An **abstract view interaction model** which defines permitted interactions based on the information demands v_a.

A **symbolic model** s_m summarizes the rendered symbols, i.e. instances of shapes like rectangles, lines, etc., such that ultimately s_m is the visualization as such. A **symbolic interaction model** offers interactions on the actual visualization. These interactions are of general concern for all s_m, e.g. navigation or adaptive zooming [CG02], and do not relate to d or i_m. A **visualization model** vis_m is the definition of visual primitives, i.e. shapes like rectangles, lines, etc. and simple compositions thereof. Thereby, s_m is an instantiation of vis_m which has been fully configured, e.g. a red dotted line. A **visual interaction model** vis_i are the interactions, that come with selections $s \subseteq vis_m$. Thereby, s is e.g. a rectangle which is draggable and may change its size on user manipulation. The instantiated and configured mapping of v_m to vis_m can be summarized as a **viewpoint** in line with ISO/IEC 42010:2007 (cf. [Int07]). An **abstract visualization model** vis_a that describes more complex compositions of elements of vis_m. Thereby, vis_a is not an instance of a vis_m but a predefined composition, i.e. blueprint or building block, with additionally specified variability points defined in vis_a that may modify the actual appearance of s_m. An **abstract visual interaction model** that describes possible interactions from the pure visual point of view with respect to the predefined configurations. For instance, a text not fitting inside a rectangle is cut after reaching a maximum length and a '...' string is appended. In addition to cutting off the over-sized text so that the text object visually fits, a tool tip is added to give the end-user feedback of the actual text contained. Such a behaviour is independent from a concrete visualization and thus can be defined in an abstract manner therefore constituting a separate model. A described mapping of v_a to vis_a including variability points can be summarized as a **viewpoint building block (VBB)** in line with Buckl et al. [BDMS10].

[2] As used e.g. in the visualizations introduced in [BMR+10].

[3] Such an update may require to create stub objects or require additional user interventions depending on the concrete information model.

3 Requirements for interactive EA visualizations

With a focus on EA management, we identified the following requirements for interactive EA visualizations which will be explained with regard to the aforementioned conceptual framework.

As outlined above, in the context of EA management enterprises tend to use organization-specific information models since there is no common standard EA model to describe an entire organization's information demands. Consequently, it has to be ensured that an arbitrary information model can be visualized dynamically, i.e. an EA visualization tool should be able to generate visualizations out of data without the need to manually adapt to information models (**Re1**). More technically speaking, this also implies that the mapping of an information model to a visualization model must be performed dynamically at runtime and be configurable by end-users (**Re1.1**).

EA management visualizations are not only used to view data but are also consulted when making strategic decisions. These often require impact analyses which can be performed best in a graphical manner, i.e. by manipulating the symbolic model directly (**Re2**) performing 'what-if' analyses. EA management has many perspectives and angles to view at depending on different stakeholders with different concerns ending up in stakeholder-specific visualizations highlighting relevant data for a special issue [AKRS08, IL08, Mat08, BGS10]. Interactive EA visualizations must be able to visualize a subset of the data model or the information model (**Re2.1**) while offering valid interactions and keeping consistency [DQvS08]. Thereby, these manipulations should not only influence the visualization but also underlying data so that changes to the symbolic model are propagated to the respective data model and information model (**Re2.2**), being permitted and constrained by an underlying data interaction model and interaction model.

Interactions with the visualization, i.e. the symbolic model, should be preferably smooth. Following [Nie94] "the limit for having the user feel that the system is reacting instantaneously" is about 0.1 second. To provide EA visualizations in a decentralized manner (cf. [BMNS09]), a solution is intended to use a client/server architecture allowing a centralized data model and information model while the generated visualizations can be decentrally viewed and manipulated (**Re3**). In this vein, a major challenge is the reduction of needed round-trips for propagating changes in the symbolic model, which is client-sided to the data model, possibly located at the server. During such a round-trip all kinds of interactions have to be locked in order to guarantee that the semantic integrity of the data model is not violated through any further incompatible interactions following the ACID (atomicity, consistency, isolation, durability) properties. Possible round-trips may take up to a couple of seconds leading to decreased user adoption. As a consequence as many as possible restrictions related to the permitted user interactions, defined by the interaction model, should be available intermediately within the client such that manipulations to the symbolic model are limited to a minimum and hence increase usability (**Re3.1**).

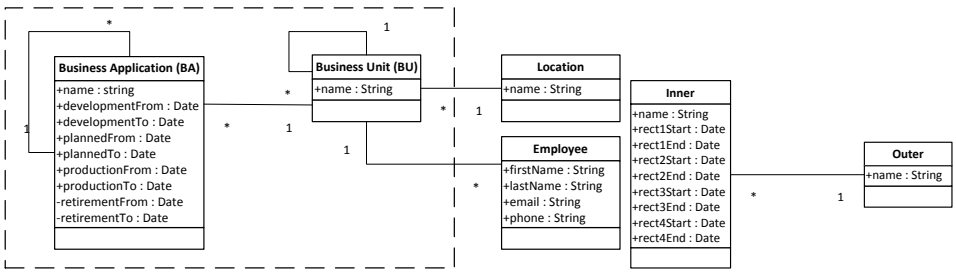
In [BELM08], Buckl et al. have shown that visualizations, so-called V-Patterns, recur in the discipline of EA management. Buckl et al. also synthesized these V-Patterns in so-called viewpoint building blocks (VBB). Considering the framework explained above,

these V-Patterns are viewpoints whereas the paradigm of VBBs is adapted. Buckl et al. showed in [BELM08] that recurring patterns are reused and combinations thereof. Therefore, EA visualizations must be defined as pre-configured, parameterized[4] building blocks (**Re4**) in order to increase re-usability of existing software artifacts and accelerate development periods. Moreover, EA visualizations must be generated employing such building blocks allowing combinations thereof (**Re4.1**) in order to enable more complex combinations of visualizations out of building blocks by end-users.

4 Prototypical implementation

Based on the framework introduced in Section 2 a prototypical implementation is developed which will be described in this section referring to the requirements of the previous section.

In the following, the process of generating an EA visualization, i.e. generating a symbolic model, is explained in detail by an exemplary information model and data model. The EA visualization generated is taken from Buckl et al. (V-Pattern 26 in [BELM08]) since they used a pattern-based approach, i.e. they observed this kind of visualization with underlying an information model at least three times[5] in practice[6].



(a) Information model with view model (in dashed lines) (b) Abstract view model

Figure 2: Pattern Matching of abstract view model and information model

Figure 2(a) shows an excerpt from an information model consisting of *business applications* related to each other and *business units* that use them and are based at a certain *location* having *employees* that work at business units. An exemplary instantiation of this information model is illustrated in Figure 3 which is used in the following example to generate a time-interval map (cf. [BELM08] or Figure 5).

The first step towards generating a visualization is to define a VBB as an abstract template (**Re4**). Thereby, an abstract view model (see Figure 2(b)) is created stating that 'outer' and 'inner' entities linked via a 1:n relationship are the information demands for

[4] In this context, parameterized means explicitly defined variability points.

[5] For an explanation of the 'rule of three' see [AIS+77].

[6] This proves practical relevance as desired for a design science approach (cf. [HMPR04]).

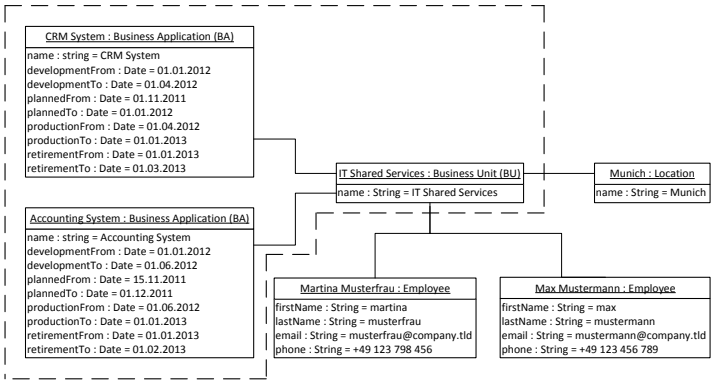
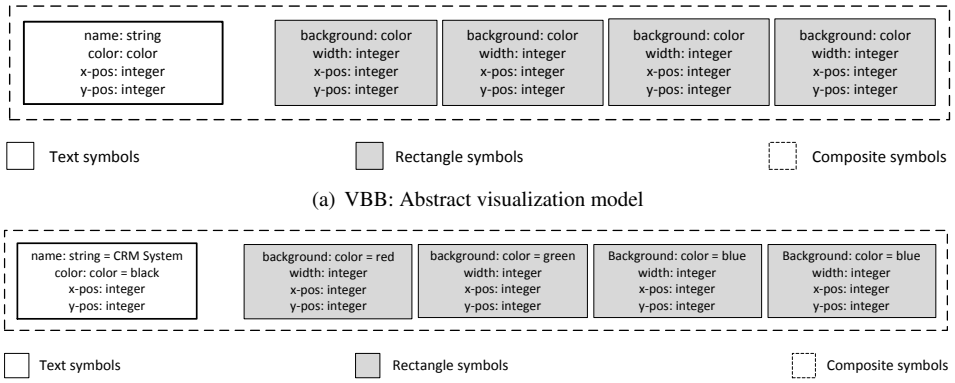


Figure 3: Data model with view data model (in dashed lines)

this VBB. This formal specification of the information demands is used to match the pattern of abstract view model against a given information model with the pattern matching using library IncQuery [BHR+10b]. The pattern matcher searches the information model to match for potential 'outer' entities required to have a *name* attribute connected via a 1:n relationship to 'inner' entities that are required to offer four pairs of start-fields and end-fields and a *name* attribute. This technique is used to offer an end-user the possibility of visualizing data with an arbitrary information model (**Re1**) by choosing the information of interest from a list of potential 'outer' and 'inner' entities (**Re1.1**). Besides the information demands, the VBB defines symbols that are used to visualize the chosen information, i.e. it describes variability points of elements of the visualization model in an abstract visualization model (Figure 4(a)). For instance, symbols like rectangles or circles commonly offer different color-configurations or even could be used interchangeably (e.g. use circles instead of rectangles). Commonly, elements of the visual model become visible in a symbolic model, while the composite symbol, shown as dotted line in Figure 4(a), is used as logical container for a set of symbols and is not directly visible. As shown in Figure 4(a) the abstract visualization model defines that each inner entity is represented through three kinds of symbols, namely a composite symbol, a text symbol and four rectangle symbols. Furthermore, the composite symbol is conceived to enable the setting of constraints/rules for all symbols contained therein. Figure 4(a) illustrates text and rectangle symbols embodying different attributes that are used for the transforming process later on. Finally the VBB defines a mapping between abstract view model and abstract visualization model. Therefore, the VBB states whether an object/attribute of the abstract view model is directly bound to a corresponding object/attribute of the abstract visualization model and how exactly. Moreover, the VBB also defines objects/attributes of the abstract view model that are employed to calculate [7] one or more objects/attributes of the abstract visualization model. In the given example the name attribute of an 'inner' entity is directly bound to the name attribute of a text symbol, whereas pairs of start- and end-date attributes are used to calculate the width of rectangle symbols. After matching

[7] We call these *derived attributes* which can be the result of any calculation, e.g. a sum of values, transitive relationships in the information model, etc.



(a) VBB: Abstract visualization model

(b) VBB: Visualization model

Figure 4: The connection between abstract and non-abstract visualization model

the pattern of information demand of an abstract view model to determine which part of an information model is needed the view data model highlighted with dashed lines in Figure 3 is extracted from the data model (**Re2.1**). A so-called *viewpoint configuration* is used to set relevant parameters so that the viewpoint can process the view data model that is passed over to it. Thereby, the viewpoint configuration states which fields of the information model are mapped to which fields of the abstract view model which are chosen from a list of all possible entities and combinations thereof by the end-user (**Re1.1**) after the pattern matching. In the given example each pair of *from* and *to* values is mapped to one pair of *rectXStart* and *rectXEnd*, i.e. *developmentFrom* is used for *rect1Start* and *developmentTo* is mapped to *rect1End*. All other *rectX* fields of the abstract view model are mapped in a similar way. Furthermore, the field *name* of a *business application* entity is mapped to the inner *name* attribute of the abstract view model. Besides the concrete mapping of an information model to an abstract view model, the abstract visualization model is parametrized. In our example, the colors of the rectangles and the font-size of the text are set. The viewpoint configuration itself is passed to a restful Web Service as a JSON string where it is processed and passed over to the VBB. The resulting runtime models that are created through parameterizing the abstract view model and abstract visualization model are the view model (Figure 2(a)) and elements from the visualization model (Figure 4(b)) of the viewpoint. Being fully configured the viewpoint is finally used to process all entities of the view data model. In this step for each entity of the view data model, one *row* of the time-interval map, whose structure is defined in the visualization model, is added to the symbolic model that can be seen in Figure 5. At this point, also the layout is done, i.e. setting x/y position and width parameters are calculated on the basis of the attributes of the view data model. In our prototypical implementation, this result is JavaScript code utilizing the Raphaël framework to generate the visualization in the web browser of the client (**Re3**). In addition, the VBB not only specifies symbols or groups thereof in terms of composite symbols, but also equips them with predefined interactions so that the user can manipulate the visualization (**Re2**). In order to be able to set only permitted interactions, different information sources are used. As explained in Section 2 the interaction

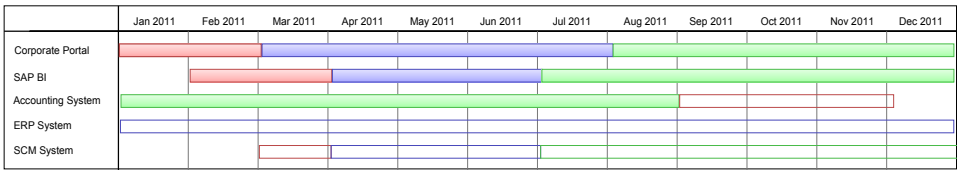


Figure 5: Symbolic model

model determines which actions are allowed upon the data model without affecting the integrity of the information model, whereas the data interaction model checks access rights on a particular element in the data model. On the other hand, the abstract view interaction model states which kind of interactions are allowed upon the abstract view model within this VBB. In the given example (cf. Figure 6) the fields *rectXStart* and *rectXEnd* of the inner entities can be changed (in italic) , whereas the *name* field must remain the same (in bold). Propagating these changes to the data model is only allowed because they are based upon bijective functions. In contrast, when using derived attributes that are calculated using aggregated values, interactions are not permitted since an interaction not based upon a bijective function would inevitably cause trouble while propagating changes to the data model (**Re2.2**). Additionally the abstract visual interaction model determines the permit-

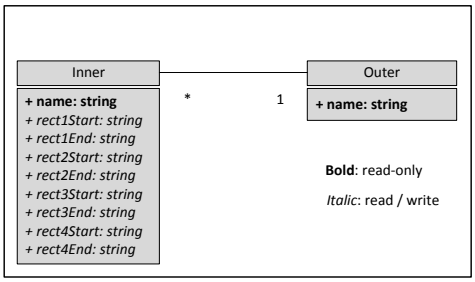
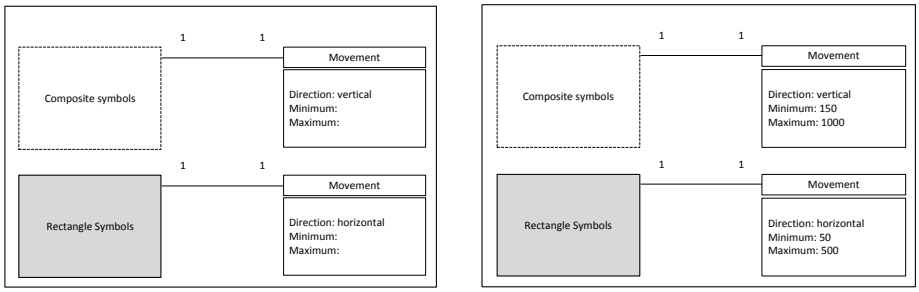


Figure 6: Abstract view interaction model

ted interactions upon the different symbols that are used for rendering the symbolic model later on, i.e. changes to the width of an element can be interpreted as changes of a date. Interactions are represented by constraints that are attached to different symbols, whereas for each possible interaction that can be triggered, like moving or dragging and dropping symbols, a constraint is implemented with individual parameters that can be customized to restrict this interaction. As an example, the rectangles of the time-interval map can be moved horizontally only, whereas the composite symbols are limited to vertical movement. Figure 7(a) shows each composite and rectangle symbol equipped with a *Movement* constraint to achieve the ascribed functionality. The *Movement* constraint itself has three parameters, direction, minimum and maximum, that have to be set up in order to enable the equipped symbol to be moved in the given direction and between the minimum and maximum value. Accordingly, if a symbol should be moved horizontally and vertically, two *Movement* constraints will have to be attached to the symbol. Besides *Movement*, fur-

ther constraints, like *Resizing* or *Containment* have been implemented, but are not needed for this particular kind of visualization. Each of these constraints has its own parameters set up individually in a VBB. In addition to interaction constraints, the symbolic interactions we used focus on direct user feedback, i.e. tool tip texts or highlight on selection. In our example tool tip texts are used when hovering over or dragging an end of rectangles (*rectXStart* or *rectXEnd*). As shown in Figure 5, some of the rectangles are not filled reflecting read-only access gathered from the data interaction model. With the abstract view interaction model describing which interactions are allowed upon the abstract view model and the abstract visual interaction model stating which user interactions are allowed upon the different symbols of the abstract visualization model, the mapping between these two models, that is aligned with the mapping between the abstract view model and abstract visualization model, it is specified how the permitted user interactions of the abstract visual interaction model affect the attributes of the entities of the abstract view model so that round-trips are omitted in the first place (**Re3.1**). In the given example the abstract visual interaction model prescribes that the rectangles can only be moved horizontally. Furthermore the abstract view interaction model states that only the values of *rectXStart* and *rectXEnd* of inner entities can be changed. In addition the mapping between these models describes that a horizontal movement of a rectangle symbol causes the *rectXStart* and *rectXEnd* fields of the corresponding inner entity to be updated to the current positions. On instantiation, the VBB is configured to a viewpoint with the visualization configuration



(a) VBB: Mapping of abstract visualization model and abstract visual interaction model

(b) Viewpoint: Mapping of visualization model and visual interaction model

Figure 7: Abstract visual interaction model and visual interaction model

that may contain parameters for the abstract visual interaction model which can be seen in Figure 7(b). In our prototype, different VBBs or combinations thereof (**Re4.1**) can be used. After processing all entities of the view data model the information gained from the view interaction model and visual interaction model of the fully configured viewpoint is used to enrich the created symbolic model with the permitted possibilities of user interactions (**Re2.2 & Re3.1**), in terms of symbols being equipped with the corresponding interaction constraints.

5 Related Work

This section gives a brief overview of interactive visualizations. Some of these interactive visualizations constitute a visual domain-specific language, whereas others are mere drawing tools and are not bound directly to a data and/or information model.

JS Library D3[8]. With the JS based library D3 it is possible to create manifold visualizations from structured data. Examining the structure of D3 it is shown that some kind of view model can be found even within this framework, specifying the structure of the data that can be loaded in order to generate a visualization. At this point it has to be mentioned that due to the static view model a mapping between an arbitrary information model and this view model has to be implemented separately. Thus only one kind of information model can be processed at a time without creating a new mapping between a second information model and the view model. Besides the view model D3 contains a visualization model constituting the symbols to be used for generating visualizations. Without an extension D3 contains circles, squares, crosses, diamonds and triangles as possible symbols. Looking at D3's possibilities for user interaction it can be seen that only rudimentary functions are available. For example, it is possible to select a subset of the view data model and update the visualization with this extract but there exists neither a possibility of changing the visualized data nor can changes be propagated to the view data model not to mention to the data model with respect to data integrity to the respective information model. With regard to the above identified requirements it shows that a mapping between an arbitrary information model and the view model has to be implemented separately, which is why **Re1** is just partly fulfilled and end-user configuration (**Re1.1**) is not offered by D3. In contrast, **Re2** can not be fulfilled entirely, since D3 focuses on interactions that center around giving user feedback. D3 does provide functions for selection of subsets of the view data model (**Re2.1**). In general, changes upon the symbolic model cannot be propagated back to a data model (**Re2.2**). **Re3** is fulfilled in partially, because D3 being a web framework written in JavaScript, a client/server architecture can be implemented, but communication with the server, respectively with the information model and data model would have to be implemented separately. Hence, complete fulfillment of **Re3** is not given. Additionally, as all client/server communication would have to be implemented (**Re3.1**) is thought not be fulfilled. Furthermore D3 offers different possibilities for definition of predefined parametrize visualization types (Re4) that can be reused or even combined with manageable effort in order to create new kinds of visualizations, leading to (**Re4**) and (**Re4.1**) being fulfilled.

yFiles. yFiles [WEK02] is a Java class library for rendering and analyzing visualizations, especially graphs. Therefore, it provides separate packages to analyze, layout, or draw visualizations on a Java Swing form. An exemplary application showing all of the main features of yFiles is yEd, a tool for creating visualizations of graphs, networks, and diagrams[9]. Within yFiles there exists a single static view model for all kinds of visualizations that can be rendered with this framework, mainly consisting of 'nodes' and 'edges'. Additionally, a separate visualization model can be found for each type of visual-

[8] See `http://mbostock.github.com/d3/` last accessed: Oct. 26, 2011.

[9] See `http://www.yworks.com/de/products_yed_about.html` last accessed Oct. 26, 2011.

ization, determining the symbols to be used for visualizing the entities of the view model and for combining them in order to generate the symbolic model. Furthermore, there exists a visual interaction model for each visualization model stating which interactions are permitted for each symbol of the generated symbolic model. Using yFiles comes with a mapping that has to be prepared in order to process an organization-specific information model and corresponding data model, thus **Re1** is partially fulfilled since the view model is static. Visualizations generated using yFiles include possibilities of parameterizing these in a user-friendly manner (**Re1.1**). As yFiles contains manifold possibilities for user interaction (**Re2**) is fulfilled, whereas selections of subsets of the data model are not included (**Re2.1**). **Re2.2** cannot be fulfilled completely as changes to the symbolic model are propagated to the view model but not to the data model. Since yFiles is implemented using Java there is a possibility of implementing a solution as an applet or using Java Web Start technology to transfer interaction constraints to a client (**Re3**). As yFiles offers no possibility for propagating changes to the data model (**Re3.1**) is not fulfilled. Only a few types of visualizations can be generated without substantially extending yFiles and other visualizations cannot be predefined (**Re4**). Also, yFiles does not include any possibility of combining different visualization types (**Re4.1**).

Visio. Microsoft Visio is a desktop application to create any kind of symbolic models, reaching from business processes in BPMN notation to construction blueprints. Among the possibility of creating all these visualizations by hand, Visio offers the possibility of creating these out of data files, or databases out of a predefined format. However this option is severely restricted as it uses a static view model and does only provide a small amount of parameters to set when querying an information model. For instance, generating an organizational chart out of a spreadsheet or database can serve as an example, as just a few parameters have to be mapped to possible fields that can be shown in the visualization. Thereby, one of them is indicating the relationship between the entities. In this context Visio contains a static view model, being bound to a very limited information model. Besides this view model there exist visualization models and visual interaction models within Visio for each kind of visualization that can be rendered. Visio's potential to be used for generating EAM visualizations can be shown by considering the above mentioned requirements. Visio is able to use different information models, thus **Re1** can be fulfilled partially. So as **Re1.1**, because Visio offers a small set of possibilities of parameterizing visualizations. As Visio offers manifold possibilities for user interaction **Re2** is completely fulfilled, but Visio lacks the functionality of selecting subsets of the data model, hence **Re2.1** is not fulfilled. The missing functionality of propagating changes within the symbolic model to the view model or the data model leads to **Re2.2** not being fulfilled. As Visio is a desktop application **Re3** and **Re3.1** are not covered, as no client/server architecture can be implemented though no statement about round-trips can be made. Similar to yFiles, Visio includes a limited set of visualization types that can be parametrized in some cases, but cannot be predefined (**Re4**) and does not support combination thereof (**Re4.1**).

Generic Modeling Environment (GME). The main purpose of the GME is to create a (visual) domain-specific language (DSL) with separated information model and its representation in the sense of a symbolic model. Therefore, GME uses a metamodel which is represented through MetaGME that describes the main aspects of the employed infor-

	D3	yFiles	Visio	GME	GMF
Re1	◑	◑	◑	●	●
Re1.1	○	●	◑	○	○
Re2	◑	●	●	●	●
Re2.1	◑	○	○	○	○
Re2.2	◔	◑	○	●	●
Re3	◑	◔	○	○	○
Re3.1	○	○	○	○	○
Re4	●	○	○	●	●
Re4.1	◕	○	○	○	○

Table 1: Visualization capabilities of the presented approaches

mation model. Furthermore, model related constraints can be integrated using the Object Constraint Language (OCL). These constraints are equipped with priorities and corresponding actions that have to be performed in case of a violation of themselves. Evaluating GME against our requirements, GME shows that it is possible to generate visualizations out of any kind of information model (**Re1**), but only one concrete information model can be processed at a time and no configuration at runtime is offered, especially when it comes to support for an end-user configuration (**Re1.1**). Moreover, GME offers far-reaching possibilities for user interaction (**Re2**), but a selection of subsets of the data model is not included (**Re2.1**). All changes to the symbolic model are propagated to the data model leading to (**Re2.2**) being fulfilled entirely. Just like Visio, GME is a desktop application, wherefore **Re3** and **Re3.1** cannot be fulfilled for aforementioned reasons. As it is one of GME's main purposes it includes functionalities for predefined visualization types (**Re4**), but it does not provide any possibility for combining two or more of these types in order to create new kinds of visualizations (**Re4.1**).

Graphical Modeling Framework (GMF). Like GME, GMF aims at providing a framework to construct a DSL with separated information model and graphical representation of entities thereof. Given that GMF is based on the Graphical Editing Framework (GEF) [RWC11] it has the same capabilities for constructing and manipulating models as the GEF. GMF offers a wide range of possibilities of implementing constraints, i.e. constraints can be formulated in OCL, as regular expressions or implemented as Java code. Due to their similarity the GME and GMF provide similar capabilities of fulfilling the requirements. GMF also has the ability to process any kind of information model (**Re1**), but it has to be adapted to each information model that shall be used (**Re1.1**). GMF also provides manifold possibilities for user interaction (**Re2**), but does not include any functionalities for a selection of subsets of the data model (**Re2.1**). Users changes to the symbolic model are propagated to the respective data model (**Re2.2**) while the integrity of the information model is guaranteed. Like GME, GMF is a desktop application, which is why (**Re3**) and (**Re3.1**) cannot be fulfilled. As GMF allows the definition of predefined visualization types but does not allow the combination of these types (**Re4**) is fulfilled, while (**Re4.1**) cannot be fulfilled.

6 Conclusion and Outlook

After motivating interactive visualizations for EA management, this paper introduced a conceptual framework to realize interactive visualizations. In this paper we demonstrated how the framework can be used 1) to compare different visualization frameworks and 2) as a reference architecture to describe and implement visualization tools. During the latter, we showed how we implemented the different models and how they interact with each other for a specific visualization. Thereby, the chosen visualization originates from practice and was found as recurring pattern in the EA management domain.

In line with the design science approach of Hevner [HMPR04], further research will detail and refine the models of the introduced framework by incorporating feedback from practical applications. We expect that for the EA management discipline only a number of relevant viewpoint building blocks and respective interactions have to be developed while the remaining ones are combinations thereof. In particular parameters, variability points of visualizations and further relevant interactions for industry have to be found by empirical evaluation of the created design artefact. Currently, the prototypical implementation has been applied to a pattern-based case. Further research can broaden the scope these visualizations are applied to and also observe interactions actually employed by end-users.

In order to describe interactions, a formal language for describing interactivity is currently missing. As of today, images with 'arrows and boxes' summarized as mock-ups are used in combination with a full-text description of possible interactions so that behavior and semantics become clear to end-users. Further research could address this issue incorporating different ways currently used to describe interactive behavior of visualizations and prove the usability by end-user studies. Such a language could facilitate the way end-users evaluate mock-ups of interactive visualizations in general and in particular for the domain of EA management.

References

[AIS+77] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language*. Oxford University Press, New York, NY, USA, 1977.

[AKRS08] Stephan Aier, Stephan Kurpjuweit, Christian Riege, and Jan Saat. Stakeholderorientierte Dokumentation und Analyse der Unternehmensarchitektur. In Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, and Christian Scheideler, editors, *GI Jahrestagung (2)*, volume 134 of *LNI*, pages 559–565, Bonn, Germany, 2008. Gesellschaft für Informatik.

[BBDF+11] Marcel Berneaud, Sabine Buckl, Arelly Diaz-Fuentes, Florian Matthes, Ivan Monahov, Aneta Nowobliska, Sascha Roth, Christian M. Schweda, Uwe Weber, and Monika Zeiner. Trends for Enterprise Architecture Management Tools Survey 2011. Technical report, Technische Universität München, 2011. (to appear).

[BDMS10] Sabine Buckl, Thomas Dierl, Florian Matthes, and Christian M. Schweda. Building Blocks for Enterprise Architecture Management Solutions. In Frank et al. Harmsen,

editor, *Practice-Driven Research on Enterprise Transformation, second working conference, PRET 2010, Delft*, pages 17–46, Berlin, Heidelberg, Germany, 2010. Springer.

[BEL+07] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Kathrin Schneider, and Christian M. Schweda. A pattern based Approach for constructing Enterprise Architecture Management Information Models. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and Schnizler, editors, *Wirtschaftsinformatik 2007*, pages 145–162, Karlsruhe, Germany, 2007. Universitätsverlag Karlsruhe.

[BELM08] Sabine Buckl, Alexander M. Ernst, Josef Lankes, and Florian Matthes. Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008). Technical report, Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany, 2008.

[BGS10] Sabine Buckl, Jens Gulden, and Christian M. Schweda. Supporting ad hoc Analyses on Enterprise Models. In *4th International Workshop on Enterprise Modelling and Information Systems Architectures*, 2010.

[BHR+10a] G. Bergmann, A. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Okrös. Incremental Model Queries over EMF Models. In *ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems*, 2010.

[BHR+10b] Gábor Bergmann, Ákos Horváth, István Ráth, Dániel Varró, András Balogh, Zoltán Balogh, and András Ökrös. Incremental Evaluation of Model Queries over EMF Models. In *Model Driven Engineering Languages and Systems, 13th International Conference, MODELS 2010*. Springer, Springer, 2010.

[BMM+11] Sabine Buckl, Florian Matthes, Ivan Monahov, Sascha Roth, Christopher Schulz, and Christian M. Schweda. Enterprise Architecture Management Patterns for Enterprise-wide Access Views on Business Objects. In *European Conference on Pattern Languages of Programs (EuroPLoP) 2011*, Irsee Monastery, Bavaria, Germany, 2011.

[BMNS09] Sabine Buckl, Florian Matthes, Christian Neubert, and Christian M. Schweda. A Wiki-based Approach to Enterprise Architecture Documentation and Analysis. In *The 17th European Conference on Information Systems (ECIS) – Information Systems in a Globalizing World: Challenges, Ethics and Practices, 8.–10. June 2009, Verona, Italy*, pages 2192–2204, Verona, Italy, 2009.

[BMR+10] Sabine Buckl, Florian Matthes, Sascha Roth, Christopher Schulz, and Christian M. Schweda. A Method for Constructing Enterprise-wide Access Views on Business Objects. In Klaus-Peter Fähnrich and Bogdan Franczyk, editors, *GI Jahrestagung (2)*, volume 176 of *LNI*, pages 279–284. GI, 2010.

[BURV11] Gábor Bergmann, Zoltán Ujhelyi, István Ráth, and Dániel Varró. A Graph Query Language for EMF models. In Jordi Cabot and Eelco Visser, editors, *Theory and Practice of Model Transformations, Fourth International Conference, ICMT 2011, Zurich, Switzerland, June 27-28, 2011. Proceedings*, volume 6707 of *Lecture Notes in Computer Science*, pages 167–182. Springer, Springer, 2011.

[CG02] Alesandro Cecconi and Martin Galanda. Adaptive zooming in web cartography. In *Computer Graphics Forum*, volume 21, pages 787–799. Wiley Online Library, 2002.

[DQvS08] Remco M. Dijkman, Dick A.C. Quartel, and Marten J. van Sinderen. Consistency in multi-viewpoint design of enterprise information systems. *Information and Software Technology*, 50(7–8):737 – 752, 2008.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[IL08] Hannakaisa Isomäki and Katja Liimatainen. Challenges of Government Enterprise Architecture Work – Stakeholders' Views. In Maria Wimmer, Hans Jochen Scholl, and Enrico Ferro, editors, *Electronic Government, 7th International Conference*, pages 364–374, Turin, Italy, 2008. Springer.

[Int07] International Organization for Standardization. ISO/IEC 42010:2007 Systems and software engineering – Recommended practice for architectural description of software-intensive systems, 2007.

[Lee99] Y.T. Lee. Information modeling: From design to implementation. In *Proceedings of the Second World Manufacturing Congress*, pages 315–321. Citeseer, 1999.

[Mat08] Florian Matthes. Softwarekartographie. *Informatik Spektrum*, 31(6), 2008.

[MBF+11] Stephan Murer, Bruno Bonati, Frank J. Furrer, Stephan Murer, Bruno Bonati, and Frank J. Furrer. *Managed Evolution*. Springer Berlin Heidelberg, 2011.

[MBLS08] Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M. Schweda. *Enterprise Architecture Management Tool Survey 2008*. Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany, 2008.

[MWF08] Stephan Murer, Carl Worms, and Frank J. Furrer. Managed Evolution. *Informatik Spektrum*, 31(6):537–547, 2008.

[Nie94] Jakob Nielsen. *Usability Engineering*. Elsevier LTD, Oxford, 1994.

[Ros03] Jeanne W. Ross. Creating a Strategic IT Architecture Competency: Learning in Stages. *MIS Quarterly Executive*, 2(1), 2003.

[RWC11] Dan Rubel, Jaime Wren, and Eric Clayberg. *The Eclipse Graphical Editing Framweork (GEF)*. Addison-Wesley, 2011.

[Sch11] Christian M Schweda. *Development of Organization-Specific Enterprise Architecture Modeling Languages Using Building Blocks*. PhD thesis, TU München, 2011.

[WEK02] R. Wiese, M. Eiglsperger, and M. Kaufmann. yfiles: Visualization and automatic layout of graphs. In *Graph drawing: 9th international symposium, GD 2001, Vienna, Austria, September 23-26, 2001: revised papers*, volume 129, page 453. Springer Verlag, 2002.

[Wit07] André Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. PhD thesis, Fakultät für Informatik, Technische Universität München, Germany, 2007.

[WR09] P. Weill and J.W. Ross. *IT Savvy: What Top Executives Must Know to Go from Pain to Gain*. Harvard Business Press, 2009.