A Generative Approach for Creating Stakeholder-specific Enterprise Architecture Views

Sabine Buckl, Florian Matthes, and Christian M. Schweda

Technische Universität München, Institute for Informatics,
Boltzmannstr. 3, 85748 Garching, Germany
{sabine.buckl,matthes,schweda}@in.tum.de
http://www.systemcartography.info

Abstract. Understanding the architectures of complex system, e.g. enterprises, is greatly facilitated by using graphical *views* thereof. These views result from the application of an underlying *viewpoint* to a comprehensive architectural description. The viewpoint thereby describes, which architectural concepts should be visualized in which way. The creation of views that consistently represent the enterprise architecture (EA) from a specific viewpoint, is a challenge of ongoing interest in EA management. In this paper, we present a technique that can be used to create views consistent to arbitrary architecture viewpoints and show, how this technique is realized in a prototypic tool. Central constituents of the tool are a model providing the graphical primitives for describing visualizations (called *visualization model*) and a model-to-model transformation reifying an architectural viewpoint.

Key words: model transformation, viewpoints, views, EA management

1 Motivation

A major task of enterprise architecture (EA) management is to provide transparency concerning the overall architecture, i.e. a common goal of EA management is to foster the communication between the different stakeholders with business and/or IT background [1]. Commonly, visualizations (views) are regarded as important means to facilitate communication between the stakeholders from business and IT. In order to be useful these visualizations should not be arbitrary "drawings" of the architecture, but should correspond to selected architecture *viewpoints*[1] on dedicated parts of the overall architecture (cf. [4, 9]). What up to this points reads quite similar to the challenge of architecture visualization in the context of software engineering, shows at the second look some subtle complexities. In contrast to the situation in software engineering, where widely-accepted conceptualizations of software systems, e.g. via "classes", "components", and "interfaces", exist, the field of EA management is lacking such well-defined and

[1] The term *viewpoint* is used in this context according to it's definition in [7].

grounded terminology. This might be ascribed to the novelty of the field, but quite a few researchers [1, 4] give a different explanation and challenge the idea that a "one-size-fits-it-all" conceptualization of EAs exists. These researchers in contrast expect the EA conceptualization to be organization-specific, such that every enterprise has its specific EA *information model* that incorporates only the information necessary for the dedicated EA management approach. Complementing the variety of information models, different organizations also employ different visualization standards, i.e. use viewpoints differing in respect to the used symbols as well as to the employed rules underlying the layout. Against the aforementioned background of largely differing information models and visualization principles, it becomes clear that the creation of a consistent EA view (for an example see Figure 1) is no simple task.

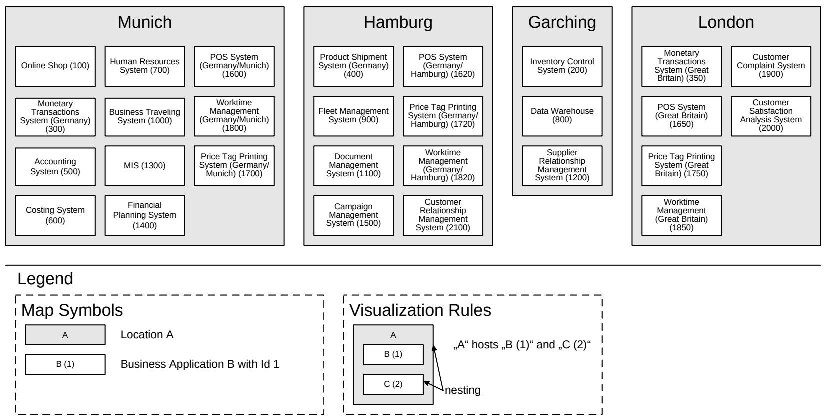
Munich
Online Shop (100)
Human Resources System (700)
POS System (Germany/Munich) (1600)
Monetary Transactions System (Germany) (300)
Business Traveling System (1000)
Worktime Management (Germany/Munich) (1800)
Accounting System (500)
MIS (1300)
Price Tag Printing System (Germany/Munich) (1700)
Costing System (600)
Financial Planning System (1400)
Hamburg
Product Shipment System (Germany) (400)
POS System (Germany/Hamburg) (1620)
Fleet Management System (900)
Price Tag Printing System (Germany/Hamburg) (1720)
Document Management System (1100)
Worktime Management (Germany/Hamburg) (1820)
Campaign Management System (1500)
Customer Relationship Management System (2100)
Garching
Inventory Control System (200)
Data Warehouse (800)
Supplier Relationship Management System (1200)
London
Monetary Transactions System (Great Britain) (350)
Customer Complaint System (1900)
POS System (Great Britain) (1650)
Customer Satisfaction Analysis System (2000)
Price Tag Printing System (Great Britain) (1750)
Worktime Management (Great Britain) (1850)
Legend
Map Symbols
A
Location A
B (1)
Business Application B with Id 1
Visualization Rules
A
B (1)
C (2)
„A" hosts „B (1)" and „C (2)"
nesting

Fig. 1. Exemplary architectural view

A technique suitable for generating views consistent with underlying viewpoints based on arbitrary EA information models has to address different challenges. Matthes et al. give in [10] a detailed list of these challenges, of which we – due to reasons of brevity – only provide a summary:

Coupling of information and visualization meaning that a link should be maintained relating graphical elements in the visualization with the underlying architectural elements. The tool's mechanisms, thereby account for visualization correctness. If the graphical elements were in contrast decoupled from the underlying data, the views would degenerate to mere drawings, which need to be manually maintained if the underlying information changes.

Flexibility of information schema meaning that architectural models conforming to an arbitrary schema should be accessible to the tool. This reflects the fact that the information on the EA is gathered according to the specific demands of an organization.

Adaptability of viewpoint meaning that an enterprise architect should be able to adapt the viewpoint to its specific needs. For example, the architect should be able to adapt the colors according to the corporate identity of the enterprise. Adaptability thereby reflects the fact that visualizations, as mean of communication, heavily rely on stakeholder acceptance.

Composition and modularization of viewpoints meaning that an enterprise architect should be able to define *viewpoint modules*, i.e. reusable parts for creating visualizations. Thereby, especially the widely-used mechanism of *layering* visualizations is accounted for.

In Section 2 we outline a technique suitable for addressing the aforementioned requirements. The technique is based on the technology of model-to-model transformations, which is applied to connect information models on the one hand with a model of visual concepts, namely the *visualization model*, on the other hand. Complementing the description of the technique, we describe the prototypical realization thereof in a tool. Related tools as well as related techniques from nearby fields are presented in Section 3. Section 4 concludes the article with a critical reflection of the presented approach and an outlook on future work.

2 Generating visualizations via model transformation

Model transformation approaches in the context of generating visualizations can be used to maintain the strict separation between information and visualization, while also ensuring consistency between both concepts. The model transformation, called *syca transformation* in our approach, links different types of models, namely the model of the information to be visualized – the *semantic model* – and the model describing the visualization – the *symbolic model*. The transformation is thereby described relying on concepts of the models' respective meta models. These are the *information model* describing the concepts used for modeling information and the *visualization model* defining the graphical concepts for describing visualizations. Figure 2 illustrates the basic constituents of the approach and further introduces the concept of the *transformation meta model*, which provides the basis for specifying syca transformations, as well as the concept of a common meta model for both information and visualization model.

Different analyses of the information models used in EA management have been undertaken, e.g. by Buckl in [3], leading to the finding that the majority of currently used models follows the paradigm of object orientation. While the information models of many approaches do not explicitly account for the underlying meta-model, the analysis of Buckl further showed that mostly only core concepts of object-oriented modeling, e.g. *classes*, *properties*, and *associations* are used. This advocates for the utilization of a simplistic common meta-model with the OMG's Meta Object Facility (MOF) [11], more precisely the essential part thereof (EMOF) being a prominent candidate. The EMOF provides the meta-model of choice for the technique presented in this paper, especially as an implementation of the modeling facility is ready at hand as part of the *Eclipse Modeling Framework* (EMF). This framework was chosen, as its metamodel, the *ECore*-metamodel, can be considered to be very similar to the EMOF-metamodel[2]. Additionally, the EMF provides serialization and editing related functionalities, as well as an active user and developer community.

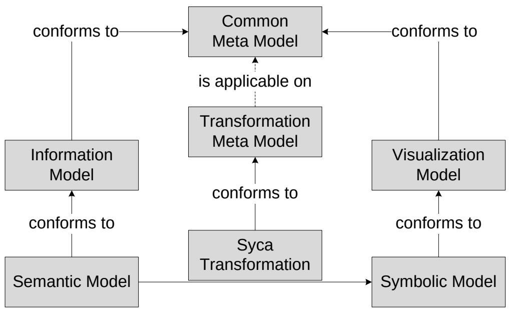

Fig. 2. Basic idea of the model transformation approach

2.1 Semantic model and information model – the *left* side

The information model sets up the language for describing the modeling subject, i.e. it introduces the core concepts, which are used to create a model of the subject's reality. In the context of EA management, the information model contains concepts like business processes, locations, etc., which are represented irrespective a visualization. Instance data documented in accordance to the information model is part of the semantic model, which contains so called *information objects*. In this sense, the information model (for an example see Figure 3) acts a meta-model for the semantic model, cf. Figure 4.

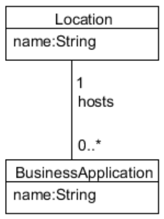

Fig. 3. Information model

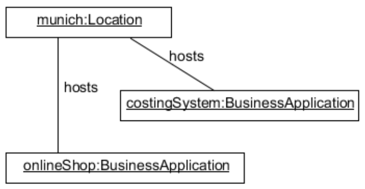
munich:Location
hosts
hosts
costingSystem:BusinessApplication
onlineShop:BusinessApplication

Fig. 4. Cutout of the semantic model

2.2 Symbolic model and visualization model – the *right* side

The visualization model contains elements, which represent graphical concepts, namely *map symbols*, e.g. "rectangle", or *visualization rules*, such as "nesting". These rules do not represent visible concepts, but they exert distinct demands on the positioning, size, or overall appearance of the symbol instances. For example, instances of the "nesting" rule demand that "inner" map symbol instances are grouped into the "outer" map symbol. Figure 5 introduces the map symbol

[2] Only minor differences concerning naming and the usage of references exist.

and visualization rule that are needed for the exemplary visualization. Figure 6 displays the symbolic model describing that rectangles representing business applications are nested in the rectangle representing the hosting location.

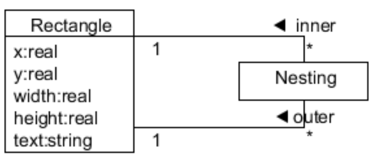

Fig. 5. Visualization model

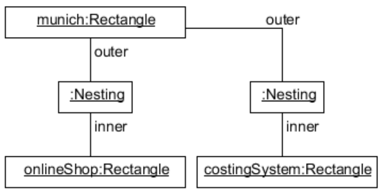

Fig. 6. Cutout of the symbolic model

The syca transformation creates a symbolic model based on the corresponding semantic model, while the map symbol instances in the symbolic model are not yet supplied with absolute positions. These positions are in a second step calculated by a layouter, which is capable to interpret the visualization rule instances and to compute appropriate positioning and sizing. Sketching the mathematical formalism incorporated in the layouter, we give examples of the layouting constraints that apply on the rectangle instance[3]:

```
munich.x - munich.width/2 < onlineShop.x - onlineShop.width/2
munich.x + munich.width/2 > onlineShop.x + onlineShop.width/2
munich.y - munich.height/2 < onlineShop.y - onlineShop.height/2
munich.y + munich.height/2 Y onlineShop.y + onlineShop.height/2
```

2.3 The syca transformation and its meta model – the *middle*

The syca transformation establishes the link between the data and its visualization and thereby enable the automatic generation of corresponding architectural views. Different types of model-to-model transformation languages may be used to implement the syca transformation.

The transformer component of the tool interprets the transformation rules and generates a symbolic model from the corresponding semantic model. Over the last years, we have successfully applied different model-to-model transformation languages for defining architectural viewpoints in an executable manner. Wiegelmann showed in [13] that the Atlas Transformation Language (ATL) [2] can be used to describe the necessary transformations. With ATL, architectural viewpoints can be defined in a highly declarative manner, although especially the appropriate instantiation of visualization rules becomes fairly complex, e.g. when matrix-like views should be created. Example ATL-like code generating the architectural visualization from Figure 1 is given below.

[3] According to the visualization model of Ernst et al. [6], the symbols' `x` and `y`-coordinates are anchored at the symbols' centers.

```
rule OrgUnit2Rectangle {
  from infoObject : Semantic.OrganizationalUnit
  to symbol : Symbolic.Rectangle (text = infoObject.name)
)

rule BusinessApp2Rectangle {
  from infoObject : Semantic.BusinessApplication
  to
    symbol : Symbolic.Rectangle (text = infoObject.name),
    rule : Symbolic.Nesting (
      inner = symbol,
      outer = transforming (infoObject.hostedAt)
    )
)
```

Complementing the model-to-model transformation languages, we further applied java to realize the syca transformations. Java-based transformations do – due to the maximum expressiveness – not run into the difficulties that the declarative model transformation languages have to deal with, but are even less intuitively to develop. Targeting an increased level of usability by rising the level of abstraction, Ramacher designed in [12] a java-based introspective framework consisting of highly-reusable transformation primitives that can be composed to syca transformations. First practical applications of this framework are still to be undertaken, but the first experiences are very promising.

3 Related approaches and tools

Domokos and Varro present in [5] an approach to ensure consistency between data and the corresponding visualization. The approach provides "open visualization framework applicable to metamodel based modeling languages", which is further developed towards executability. With no dedicated visualization model, the approach aims at transforming arbitrary information models to arbitrary visual languages, e.g. SVG, as far as both (information and visualization) can be described with *XML*. Consequently, the generation of a visualization in fact is an XSLT-based transformation between two XML-documents. In this respect, the approach does not reach a high level of abstraction, but calls for very basic transformation procedures. Further, the article does not encompass a visual language suitable for expressing the aspects of relative positioning, as the application concerns petri-nets and their representation as nodes-and-edges.

Kruse et al. present in [8] a component-oriented tool for supporting EA management. Central to their approach is a strong 'componentization' in the way, that different types of viewpoints are implemented as different components of the tool. Each viewpoint in this respect brings along a dedicated information model, that describes the information necessary for creating the corresponding visualization. A company willing to use the corresponding tool for visualizing their EA or parts thereof, selects the appropriate visualization components and supplies a model-to-model transformation transforming parts of the organization-specific EA information model to the information model associated with a specific viewpoint. Put in other words, the approach of Kruse transforms and projects instances of one information model to instances of a different information model, which conversely is directly fed into a layout and visualization mechanism.

Multiple commercial tools provide support for EA management. Matthes et al. analyzed in [10] nine prominent of these tools, coming to the result that most of the tools fall for two types of visualization-related problems. On the one hand, the *process-* or *methodology-driven* tools bring along a fixed EA information model that underlies a set of predefined viewpoints. Visualizations corresponding to these viewpoints can mostly be generated automatically, and may use a rich visual language, including relative positioning of symbols to convey information. On the other hand, *metamodel-driven* tools can flexibly adapt their information model to the specific needs of the using organization, but are mostly limited to visualizations of the nodes-and-edges type. More complex visualizations, using e.g. relative positioning, have to be programmed in these tools utilizing scripting languages with no visualization- and layout-specific concepts.

4 Critical reflection and outlook

This article presented a technique to create visualizations (views) from arbitrary EA descriptions. The generated views thereby are consistent with a previously defined viewpoint and are created using a model-to-model transformation. Complementing the presented technique also a prototypic tool implementing the technique was described. This tool has been used in different practice cases in the past and showed the applicability of the technique on various different organization-specific EA information models. Different languages were utilized to realize the model-to-model transformation, namely the basic programming language java (cf. [12]) as well as the model transformation language ATL (cf. [13]). While both languages were sufficient to generate visualizations, they provide a rather low level of abstraction, when it comes to describing architectural viewpoints. More precisely, the complexity of the model transformations expressed in these languages is often beyond the level, that an enterprise architect can cope with.

Increasing the level of abstraction in defining viewpoints and thereby facilitating the creation of end-user defined viewpoints are the challenges to be addressed next. Two different strategies to achieve this can be pursued:

- Rise the level of abstraction in the visualization model, i.e. replace the fine grained visualization rules with more coarse ones. As an example, one could think of a *cluster*-rule for visualizations like the one in Figure 1.
- Rise the level of abstraction in the transformation language, i.e. provide domain specific concepts for specifying a syca transformation going beyond basic *query model*- and *transform model*-concepts.

While both ideas may be useful to achieve the goals, especially the latter one seems more appealing. In consequent continuation of the prefabrics of Ramacher (cf. [12]), a tailored transformation language could allow to stay with the basic visualization model concepts that have a clear and unambiguous semantics.

Acknowledgements

The development of the herein presented technique and prototypic tool is supported and partially funded by Siemens IT Solutions and Services (SIS).

References

1. S. Aier, S. Kurpjuweit, C. Riege, and J. Saat. Stakeholderorientierte dokumentation und analyse der unternehmensarchitektur. In H.-G. Hegering, A. Lehmann, H. J. Ohlbach, and C. Scheideler, editors, *GI Jahrestagung (2)*, volume 134 of *LNI*, pages 559–565, Bonn, Germany, 2008. Gesellschaft für Informatik.
2. ATLAS group at LINA & INRIA. Atl: Atlas transformation language, 2006.
3. S. Buckl. *Modell-basierte Transformationen von Informationsmodellen zum Management von Anwendungslandschaften.* Diploma thesis, Fakultät für Informatik, Technische Universität München, 2005.
4. S. Buckl, A. M. Ernst, J. Lankes, K. Schneider, and C. M. Schweda. A pattern based approach for constructing enterprise architecture management information models. In *Wirtschaftsinformatik 2007*, pages 145–162, Karlsruhe, Germany, 2007. Universitätsverlag Karlsruhe.
5. P. Domokos and Varró, Dániel. An open visualization framework for metamodel-based modeling languages. *Electronic Notes in Theoretical Computer Science*, 72(2), 2002.
6. A. M. Ernst, J. Lankes, C. M. Schweda, and A. Wittenburg. Using model transformation for generating visualizations from repository contents – an application to software cartography. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), Munich, Germany, 2006.
7. International Organization for Standardization. Iso/iec 42010:2007 systems and software engineering – recommended practice for architectural description of software-intensive systems, 2007.
8. S. Kruse, J. S. Addicks, M. Postina, and U. Steffens. Decoupling models and visualisations for practical ea tooling. In *Pre-Proceedings of the 4th Workshop on Trends in Enterprise Architecture Research, November 23rd, 2009, Stockholm, Sweden*, pages 85–98), Stockholm, Sweden, 2009.
9. M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis.* Springer, Berlin, Heidelberg, Germany, 2005.
10. F. Matthes, S. Buckl, J. Leitel, and C. M. Schweda. *Enterprise Architecture Management Tool Survey 2008.* Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany, 2008.
11. OMG. Meta object facility (mof) core specification, version 2.0 (formal/06-01-01), 2006.
12. R. Ramacher. *Entwurf und Realisierung einer Viewpoint Definition Language (VDL) für die Systemkartographie.* Diplomarbeit, Fakultät für Informatik, Technische Universität München, 2009.
13. J. Wiegelmann. *Analysis and Application of Model Transformation Languages for Generating Software Maps.* Bachelor thesis, Fakultät für Informatik, Technische Universität München, 2008.