Decoupling Models and Visualisations for Practical EA Tooling

Steffen Kruse, Jan Stefan Addicks, Matthias Postina, and Ulrike Steffens

Software Engineering for Business Information Systems
OFFIS - Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
{steffen.kruse,jan.stefan.addicks,
matthias.postina,ulrike.steffens}@offis.de
http://www.st.offis.de

Abstract. Rigorous modelling techniques and specialised analysis methods support enterprise architects when embarking on enterprise architecture management (EAM). Yet, while customised modelling solutions provide scalability, adaptability and flexibility they are often in conflict with generic or reusable visualisations. We present an approach to augment customised modelling with the techniques of model transformations and higher-order transformations to provide flexible and adaptable visualisations with a minimum of requirements for the underlying enterprise models. We detail our approach with a proof-of-concept implementation and show how a decoupling can ease EAM approaches and provide appropriate tooling in practice.

1 Motivation

An adequate visualisation of interrelations is one important foundation of enterprise architecture management (EAM) [1] and often considered particularly useful for decision support by different stakeholders. However, there is still a number of obstacles to be overcome in order to support the flexible and effective creation of such visualisations.

EAM enterprise models may result from EA frameworks like described in [2,3] or more general in [4]. These frameworks provide initial samples of enterprise models which can be refined for specific enterprises or organisations. Changing markets, legal requirements, or changes in the organisational structure force enterprises to adapt accordingly. These changing business needs again have to be reflected by the underlying IT structure [5] and the respective enterprise model. The same is true for the managed evolution of enterprises' IT architectures as described e.g. in [6] which leads to new stakeholder concerns [7] and in turn call for an adjustment of enterprise models. Hence, enterprise models can be expected to be unique for each enterprise (at least at a certain degree of detail) and to mature over time. Following this line of argument, we conform to [8,9] who call for flexible and extensible enterprise models.

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 62–71, 2010.

The required flexibility of enterprise models has to be expanded to the types of analyses performed on them. Although common EA concerns can be identified [10], they still vary depending on an enterprise's specific characteristics and stage of development. Furthermore, EA research develops new concepts for further exploration of EA information for new possibilities of analyses to suit diversifying kinds of stakeholders [11,12] and more generic techniques for executing these kinds of analyses have been proposed [13,14,15].

Combining flexible enterprise models with flexible analyses is a challenging task, since each type of analysis has to rely on some basic assumptions with regard to the enterprise model it is used on. However, for EA visualisations we can mitigated this problem by decoupling models from visualisations and by enabling the design of new visualisations on detached models. This is achieved by the use of model tranformations and higher-order transformations(HOTs). These techniques allow a flexible way of visualising EA interrelations and provides adequate visualisations for different stakeholders and different concerns.

The remainder of the paper is structured as follows: In the next section, we give an overview of our approach, detailing the separate concerns in employing flexible visualisations. Section 3 covers the technical detail of prototyping our approach; starting with transforming models at run-time to produce visualisations from arbitrary enterprise models and going on to the use of HOTs for the configuration of visualisation cases. We conclude with a description of the next challenges at hand.[1]

2 Overview of Our Approach

EAM visualisations can be structured according to views and viewpoints in the software engineering technology space [16]. In this regard, a visualisation covers one or more stakeholder *concerns* by selecting suitable enterprise model elements and bundling this selection as a *viewpoint*. When the visualisation is displayed, being populated by live data (as instances of the selected enterprise model entities) a *view* conforming to the defined viewpoint is created.

Our approach can be structured according to four areas of interest, each covering different aspects of creating and utilising EAM visualisations. This separation of concerns is made possible by the decoupled use of models and viewpoints and potentially allows different experts and stakeholders to participate in the process. The areas are:

Enterprise Modelling. An enterprise model identifies domain specific and enterprise specific information objects and processes which are relevant for the EA activities of the different stakeholders. We build our prototype around the Eclipse Modeling Framework Project (EMF)[2]. We expect enterprise models to be unique for each enterprise to cater for individual characteristics. None the

[1] The research described in this paper was partially accomplished in the IF-ModE project funded by the German Federal Ministry of Education and Research.

[2] `http://www.eclipse.org/modeling/emf/`

less, modelling does not have to be started "from scratch" as enterprise models can be derived from domain reference models (e.g. [17], [18]) and by employing accepted EA frameworks like TOGAF [2] or the Zachman Framework [3].

We use an excerpt from an exemplary enterprise model to illustrate our approach (see figure 1). In essence it covers the relevant organisational units, processes, and software components and how these interact in a given enterprise. For this purpose, the *Support*-entity defines which component supports which organisational unit in which process. To order processes and organisational units, these can be organised in trees: subentities reference their parent entities via the respective *parent*-relation. When the corresponding information has been collected (i.e. an instance of this model is available), questions like "Which part of my organisation needs software X for which purpose?" can be answered. These answers are best given as suitable EA visualisations.

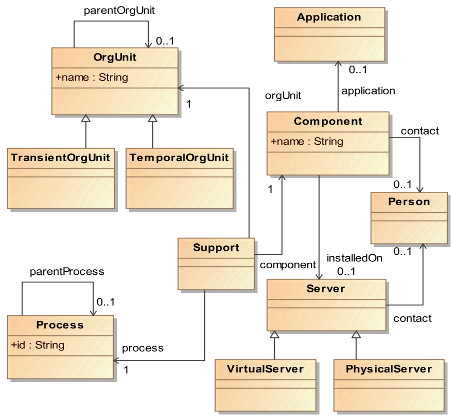

Fig. 1. Enterprise Model Example

EA Visualisation Design. For different kinds of information (and questions), appropriate forms of visualisation are needed. In general, the suitability of one type of visualisation over another does not only depend on the informational content but rather on the way the information is structured and how data entities relate to each other. A trivial example is the use of tables to display large amounts of sets of textual information with reoccurring attributes, while graph-like structures with textual annotations and directed edges are an accepted means to visualise processes. In essence, the design of the visualisation is independent of domain or enterprise context and only depends on the characteristics of the visual data. Our approach provides the means to specify visualisations in a domain-independent manner and thereby to allow visualisations to be bundled and reused in different contexts. In this way, visualisations can either be designed up-front by tool vendors and then applied to individual enterprise models or developed during an actual EA project and later reused.

Here we chose the matrix-map as an example of our work (see figure 2). The matrix-map is a suitable visualisation when looking at how one set of entities is distributed by relation over two other entity sets. The related entities make up the x- and y-axes and can be organised in sets of trees, while the entities of interest fill the matrix between the axes. Our example shows how software components of a fictional enterprise are distributed between processes and organisational units. This relationship is defined by the *Support*-entity in our exemplary enterprise model (see figure 1).

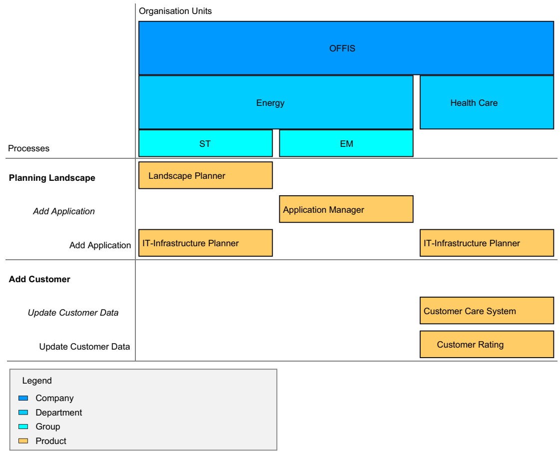

Fig. 2. Matrix-map

EA Visualisation Configuration. Once EA has been introduced within the enterprise and both, the enterprise model and the domain-independent visualisations are available; our approach provides the means to bring these two parts together. For this purpose, an enterprise architect selects a suitable type of visualisation and the model elements to be visualised. These are then mapped to configure a viewpoint of the required information to answer a specific question at hand. It is up to the enterprise architect to select the appropriate visualisation and informational content for the different stakeholder concerns.

EA-Information Procurement and Evaluation. Once viewpoints are configured, they can be filled with live data from a repository and updated at will to produce views. Stakeholders can use the resulting views (visualisations) for their informational needs. Depending on the kind of visualisation, further configuration mechanisms may be available at usage-time, such as filters or individual highlighting of information.

These four areas of interest allow us to employ our approach within different EA scenarios. For example, tool vendors can realise visualisation design as part of their tool implementation. An enterprise relying on the respective tool can then specify its individual enterprise model and configure the viewpoints to provide its stakeholders with views. Alternatively, an enterprise pursuing its own EA implementation can realise all four concerns by itself. This flexibility better enables enterprises to select the best strategy in fulfilling their EA concerns.

3 Overview of the Generation Process

As Buckl et al. [19] have shown, it is possible to use model to model transformations to extract the information needed for a specific viewpoint from an enterprise model and produce a model suited for processing by an algorithm to create the layout of an EA visualisation. This is quite obvious when models conformant to metamodels exist or can be created for both sides and can be related by a transformation. Yet, we deem the specification of model transformations to be a difficult task for people with little expertise in the area and inexpedient in practice, as a new transformation is needed for each viewpoint. We show how the transformations can be generated by higher order transformations (HOTs).

This eliminates the need for knowledge of model transformations for end-users and enables a high potential for the re-use of visualisations.

In essence, our approach relies on separating the semantics and structure of the information to be displayed from how it is displayed. This is achieved by defining the minimal set of characteristics[3] which a set of information objects is required to exhibit to be displayable as a given type of visualisation. For this purpose we create visualisation-specific models (ViMs) for each visualisation type (such as the matrix-map).

3.1 The Visualisation-specific Model (ViM)

The central entity of a visualisation is a visualisation-specific model (ViM). It defines the model elements and structure needed to configure a concrete viewpoint. Views (instances of this model) are to be processed by a layout engine to produce the final visualisation. The ViM describes valid models of a visual domain; it is independent of the semantics and architectural domain of the information to be visualised (enterprise model). In the case of the matrix-map (see the left-hand side of figure 4), the key entities are first the nodes making up the x- and y-axes, second the entities displayed in the matrix, and third the relationships between these. We refer to the entities shown on the matrix as *Items*. These have a title and a set of *Attachments*, which can provide further information on the properties *Items* posses. Each *Item* relates to exactly one *Node* on the x- and y-axes. The *Nodes* are organised in trees along the axes. We generated a Java object-model for the ViM using tooling of the Eclipse Modeling Framework Project (EMF) and developed a layouter to create a resulting diagram from the ViM as shown in figure 2.

3.2 The Transformation Process

To generate views at run-time, we make use of model transformations [20]. The model transformations describe the rules of producing views from a set of enterprise data in terms of the respective viewpoints. (Instances of the enterprise model on the left-hand side (LHS) are transformed into instances of the ViM on the right-hand side (RHS).) One transformation description covers exactly one viewpoint; it maps entities of the enterprise domain to those of the visual domain (see figure 3).

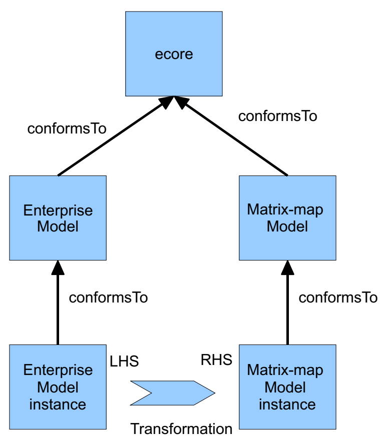

Fig. 3. Runtime Transformation

[3] We refrain from using the term "*pattern*" at this stage, as characteristics which are difficult to express in terms of patterns (like data dispersion or order) may become important in special types of visualisations. During the course of our future work on visualisation types we hope to sharpen the term.

In our prototype, we use QVT Relations (QVT-RL)[21] to describe transformations and the MediniQVT engine[4] for the execution. The enterprise metamodel is EMF ecore based. EMF ecore enjoys far reaching use in modelling in general and an appropriate model can be derived by transformation from other formats. We use Teneo of the Eclipse Modeling Framework Technology (EMFT) project[5] to produce an enterprise model from an EAM repository at runtime. Teneo provides automated persistency of EMF ecore based models. Since Teneo in turn utilises Hibernate[6], mappings to a wide variety of databases and data schemas are possible. Teneo has the added benefit of handling model queries transparently and loading entities only as they are required by the transformation process, which keeps data access to a minimum. In our example, the transformation maps *OrgUnits*, *Processes* and *Components* to *MatrixNodes* and *Items* and fills the corresponding titles. As each transformation covers one viewpoint, it means that a new transformation is needed when a different combination of entities is to be visualised (a different viewpoint is needed). For example, if we wanted to show in a matrix map who (*Person*) is responsible for which *Component* running on which *Server*, we would have to map these entities to the matrix map ViM again using a transformation description although the type of visualisation remains the same. Here we see a major drawback in relying solely on transformation descriptions for this task, as the user has to be adept at programming transformations. For this reason our approach generates transformations for viewpoints using HOTs.

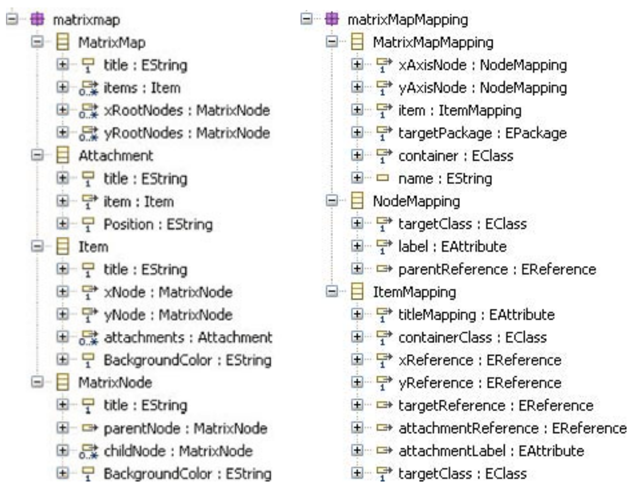

Fig. 4. Matrix-map and mapping model

[4] http://www.ikv.de/
[5] http://www.eclipse.org/modeling/emft/
[6] http://www.hibernate.org/

3.3 Generating Transformations: Higher-Order Transformations

To configure a viewpoint, we use a mapping model (MM), which maps elements of the enterprise model to elements of the ViM (see the right-hand side of figure 4). The MM is specific to the given visualisation (with constructs like x- and y-axis and nodes for the matrix-map), but specifies the required elements from the enterprise model in terms of ecore types. Thereby any enterprise model based on ecore (or any model based on ecore to be exact) can potentially serve as input and can be visualised using the given approach - without adaptation. The mapping models for the different visualisation types are augmented with a set of constraints detailing further characteristics the selected set of enterprise classes must fulfil to qualify for a viewpoint. As a simple example, consider that the EAttribute in a *NodeMapping* has to belong to the given *targetClass* EClass in order for the mapping to be correct (figure 4).

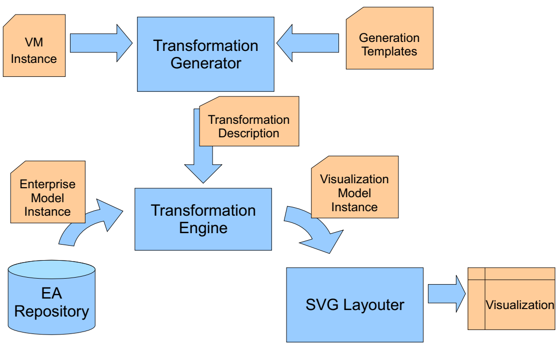

Fig. 5. Overview of the Component Interaction

To produce an executable transformation from a given mapping model instance, we make use of higher order transformations (HOTs). The overall interaction of the different components of our approach is sketched out in figure 5. The HOTs are expressed as oAW Xpand templates (see Listing 1) and create the QVT-RL transformation file required for the transformation process described above. The HOT templates are defined in terms of the mapping model, which in turn makes no assumptions on a specific enterprise model, beyond the minimal set of characteristics needed to visualise enterprise information in a visualisation type, and that the enterprise model is ecore based. In conclusion, no further requirements are imposed.

Listing 1 shows an excerpt of the Matrix-map HOT template where the QVT-RL transformation relation *XRootNodesRL* is created for the root *NodeMapping*-elements of the x-axis in the mapping model. Expressions in guillemets are part of the XPand language. The resulting transformation rule states that a *MatrixMap*-element must exist in the resulting matrix-map ViM instance, with a *MatrixNode*-element in the list of x-axis root nodes, when the class *c* of the *targetClass* type (the type of the enterprise entity to appear on the x-axis) exists, under the conditions:

- the given parent reference of the enterprise element is empty (it is a root node),
- the enterprise model element is mapped to the ViM instance by the rule *ModelToMap()* and

```
«DEFINE xRootNodes(Mapping mapping) FOR NodeMapping»
  top relation XRootNodesRL {
   checkonly domain tar model:
     «fullyQualifiedName(mapping.container)»{};
   checkonly domain tar c:
     «fullyQualifiedName(targetClass)»{};

    enforce domain mm map : matrixmap::MatrixMap
    {
      xRootNodes = node : matrixmap::MatrixNode{}
    };

  when {
   «IF parentReference!=null»
    c.«parentReference.name».oclIsUndefined();
   «ENDIF»
   ModelToMap(model, map);
   EClassToXNode(c, node);
  }
 }
«ENDDEFINE»
```

Listing 1. HOT Template excerpt

– an element of the enterprise model of the given type is mapped to a *MatrixNode* by the rule *EClassToXNode*.

When the rule is executed at run-time, all *MatrixNodes* with matching enterprise elements without parents are inserted into the *xRootNodes*-list.

The generated transformations correspond to viewpoints and can be stored for use by the intended stakeholders. When a stakeholder refreshes a visualisation, the transformation is fed with the up-to-date data from the EA repository and the desired view is produced.

We see a real benefit for the usability of EAM visualisations in providing a model (the ViM) for the configuration of viewpoints instead of having to write transformations, as the task is shifted from a programming to a modelling domain. The ViM captures all required information for producing views and the transformation generation is then fully automated (and transparent). Furthermore, by using modelling techniques instead of transformation programming, tool support for modelling becomes applicable to this task.

4 Conclusions and Future Work

We have shown how model transformations and higher order transformations can add a degree of flexibility to EA endeavours, while maintaining a high standard of rigorous modelling. The inherent benefit comes from catering for the separation of concerns by employing different techniques for different stages involved. We have constructed a first prototype of our approach and shown how visualisations can be developed so that they integrate well with customised and unique enterprise models.

We still see a knowledge-intensive and error prone task in the configuration of ViMs for viewpoints. Although well defined model constraints can reduce the impact of configuration errors, an enterprise architect still needs detailed information on the effects of all the elements of a ViM. We have so far developed a simple pattern-matching algorithm for the matrix-map visualisation, which finds suitable candidates in the enterprise model for the x- and y-axes when an entity is chosen as an *Item* and integrated it into a simple GUI. We plan on taking this idea further by providing an intuitive interface to the visualisation configuration by matching entities to user selections according to the model characteristics required by the given visualisation.

References

1. Lankes, J., Matthes, F., Wittenburg, A.: Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. In: Ferstl, O.K., Sinz, E.J., Eckert, S., Isselhorst, T. (eds.) Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety, 7. Internationale Tagung Wirtschaftsinformatik 2005, Bamberg, February 23-February 25, pp. 1443–1462. Physica-Verlag (2005)
2. The Open Group: TOGAF Version 9. Van Haren Publishing (February 2009)
3. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26(3) (1987)
4. Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. Ebookslib (2003)
5. Henderson, J.C., Venkatraman, N.: Strategic alignment: Leveraging information technology for transforming organizations. IBM Systems Journal 32(1), 4–16 (1993)
6. Hess, A., Humm, B., Voß, M., Engels, G.: Structuring software cities - A multidimensional approach. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), pp. 122–129. IEEE Computer Society, Los Alamitos (2007)
7. Postina, M., Sechyn, I., Steffens, U.: Gap analysis of application landscapes. In: The First Workshop on Service oriented Enterprise Architecture for Enterprise Engineering, Auckland, New Zealand, September 2. IEEE Computer Society, Los Alamitos (2009)
8. Kurpjuweit, S., Winter, R.: Viewpoint-based meta model engineering. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications (EMISA 2007), St. Goar, Germany, October 8-9. LNI, vol. P-119, pp. 143–161. GI (2007)
9. Kurpjuweit, S., Aier, S.: Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen. In: 9. Internationale Tagung Wirtschaftsinformatik, Wien, Österreichische Computer Gesellschaft, pp. 129–138 (February 2009)
10. TUM: Enterprise Architecture Management Pattern Catalog. Technische Universität München, Munich. Release 1.0 edn. (February 2008)
11. Gustafsson, P., Höök, D., Franke, U., Johnson, P.: Modeling the IT impact on organizational structure. In: 13th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2009), pp. 14–23. IEEE Computer Society, Los Alamitos (2009)

12. Närman, P., Johnson, P., Ekstedt, M., Chenine, M., König, J.: Enterprise architecture analysis for data accuracy assessments. In: 13th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2009), pp. 14–23. IEEE Computer Society, Los Alamitos (2009)
13. Frank, U., Heise, D., Kattenstroth, H., Schauer, H.: Designing and utilising business indicator systems within enterprise models - outline of a method. In: Loos, P., Nüttgens, M., Turowski, K., Werth, D. (eds.) Modellierung betrieblicher Informationssysteme (MobIS 2008) - Modellierung zwischen SOA und Compliance Management, Saarbrücken, Germany, November 27-28. LNI, vol. 141, pp. 89–105. GI (2008)
14. Johnson, P., Johansson, E., Sommestad, T., Ullberg, J.: A tool for enterprise architecture analysis. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), pp. 142–156. IEEE Computer Society, Los Alamitos (2007)
15. Addicks, J.S., Steffens, U.: Supporting landscape dependent evaluation of enterprise applications. In: Bichler, M., Hess, T., Krcmar, H., Lechner, U., Matthes, F., Picot, A., Speitkamp, B., Wolf, P. (eds.) Proceedings of Multikonferenz Wirtschaftsinformatik, MKWI 2008, München, February 25- February 28, GITO-Verlag, Berlin (2008)
16. Maier, M.W., Emery, D., Hilliard, R.: Systems and software engineering - recommended practice for architectural description of software-intensive systems. Technical report, IEEE Standards Association (2007)
17. Postina, M., González, J., Sechyn, I.: On the architecture development of utility enterprises with special respect to the gap analysis of application landscapes. In: Proceedings of the 3rd Workshop MDD, SOA, and IT Management (2009) (to appear)
18. TM Forum: Business process framework (eTOM), release 8.0. Technical Report GB921, TM Forum (February 2009)
19. Buckl, S., Ernst, A.M., Lankes, J., Schweda, C.M., Wittenburg, A.: Generating visualizations of enterprise architectures using model transformations. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications (EMISA 2007), St. Goar, Germany, October 8-9. LNI, vol. P-119, pp. 33–46. GI (2007)
20. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Syst. J. 45(3), 621–645 (2006)
21. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Object Management Group, Needham, MA. Version 1.0 formal/08-04-03 edn. (April 2008)