HAL
open science

EMF Views: A View Mechanism for Integrating Heterogeneous Models

Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, Jordi Cabot

To cite this version:

Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, Jordi Cabot. EMF Views: A View Mechanism for Integrating Heterogeneous Models. 34th International Conference on Conceptual Modeling (ER 2015), Oct 2015, Stockholm, Sweden. 10.1007/978-3-319-25264-3_23 . hal-01159205

HAL Id: hal-01159205

https://inria.hal.science/hal-01159205v1

Submitted on 2 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

EMF Views: A View Mechanism for Integrating Heterogeneous Models

Hugo Bruneliere[1], Jokin Garcia Perez[1], Manuel Wimmer[2], and Jordi Cabot[3]

[1] AtlanModTeam (Inria, Mines Nantes & LINA)
{hugo.bruneliere, jokin.garcia-perez}@inria.fr
[2] Business Informatics Group (Vienna University of Technology)
wimmer@big.tuwien.ac.at
[3] ICREA - UOC
jordi.cabot@icrea.cat

Abstract. Modeling complex systems involves dealing with several heterogeneous and interrelated models defined using a variety of languages (UML, ER, BPMN, DSLs, etc.). These models must be frequently combined in different cross-domain perspectives to provide stakeholders the view of the system they need to best perform their tasks. Several model composition approaches have already been proposed addressing this problem. Nevertheless, they present some important limitations concerning efficiency, interoperability and synchronization between the base models and the composed ones. As an alternative we introduce EMF Views, an approach coming with a dedicated language and tooling for defining views on potentially heterogeneous models. Similarly to views in databases, model views are not materialized but instead redirect all model access and manipulation requests to the base models from which they are obtained. This is realized in a transparent way for both the modeler and the other modeling tools using the concerned (meta)models.

Keywords: Modeling, Viewpoint, View, Heterogeneity, Virtualization

1 Introduction

Software systems are becoming increasingly complex, making them more and more difficult to comprehend, develop and maintain. To handle this complexity, they are usually represented by sets of models at different abstraction levels and possibly conforming to different modeling languages [6]. Each one of these languages is specialized to provide the modeling constructs required to deal with a particular concern of the system. Typically, several models must be combined to generate the most adequate perspective of the system for each involved person (depending on his/her role). This is challenging, notably due to the heterogeneity of the models and to the various existing (implicit) relationships between them. Such a common issue is very likely to appear in almost any non-trivial project. For instance, the TOGAF Enterprise Architecture Platform (TEAP) project[4] was a joint industrial-academic collaboration to provide support for the governance of enterprise architectures (EAs). Its base platform was SmartEA[5] that integrates

[4] http://www.teap-project.org
[5] http://www.obeosmartea.com

TOGAF[6]. Use case providers wanted to customize SmartEA and also include both business process information defined using BPMN[7] and requirement specifications defined with ReqIf[8]. Besides integrating BPMN and ReqIf models as part of SmartEA, they wanted to be able to interconnect these models with the TOGAF ones and to provide partial views on the combined models depending on given profiles (e.g., with restricted access for security reasons). Thus, they needed to specify several viewpoints linking the TOGAF, ReqIF and BPMN metamodels altogether.

To tackle such a challenge, several approaches for model composition have already been developed so far, e.g., [2, 12, 15, 17, 18]. In short, mostly all of them propose the generation of a completely new composed model automatically populated with elements copied from the set of base models that participate in the composition. Thus, they present some important limitations in terms of performance (due to the time required to copy model elements into the composed model), synchronization (due to the lack of change propagation from the contributing models to the generated one, or the other way round), and/or interoperability (the composed model may have a different nature than the contributing ones and needs to be manipulated using specialized tools).

In this paper we propose EMF Views, an alternative solution based on adapting the concept of database views [1] for models. EMF Views enables the specification of model views grouping elements using concepts coming from different metamodels. Such views can be adapted to the needs of a specific user. From a user perspective (and from a modeling tool one as well), the view behaves as any other regular model. Views are not materialized but computed on-demand, which brings some important benefits compared to previous approaches. The expressivity of our view definition language is comparable to *select-project-join* queries in relational algebra. Our approach has been implemented and made available as an open source Eclipse component.

The remainder of this paper is structured as follows. Section 2 describes the related work. Section 3 presents EMF Views while Section 4 summarizes its implementation. Finally, Section 5 critically discusses EMF Views and outlines our next steps.

2 State-of-the-Art

Views have a long tradition since the introduction of relational databases, and have been proposed for object-oriented databases as well [1, 23]. View mechanisms have also been discussed in the context of modeling languages such as UML [4] and ER [9]. In a nutshell, most of these language-specific approaches rely on query languages to define virtual elements in intra-model views. Although this support is useful in a single model context, additional mechanisms are needed to provide inter-model views. In this respect, there are two major research lines: (i) multi-viewpoint modeling used to design different viewpoints based on a unified underlying model, and (ii) model composition to define correspondences between different models to build a common view.

There are several approaches for multi-viewpoint modeling (cf. [13] for a survey). Atkinson et al. [3] propose Orthographic Software Modeling (OSM), a projective multi-

[6] `http://www.opengroup.org/togaf`
[7] `http://www.bpmn.org`
[8] `http://www.omg.org/spec/ReqIF`

view methodology in which views are dynamically generated via model transformations from a single base model. Cicchetti et al. [10] propose to define viewpoints as subsets of a base metamodel. Integrating/linking different (meta)models does not seem to be a focus in these projective approaches. Burger [7] propose a mechanism to improve views in OSM [3], in which flexible read-only views can be defined at development time. Kramer et al. [19] propose a methodology to create an underlying metamodel from base ones by using structural mappings between metamodel elements. This approach has a mechanism to support extending viewpoints and synchronizing model views with the underlying model. Bork & Karagiannis [5] propose a graphical language to define viewpoints for multi-viewpoint modeling. Finally, Romero et al. [22] propose a complementary approach for specifying and realizing correspondences between viewpoints.

Concerning model composition, there are approaches that can be used to simulate views using different link types between models [12, 17, 18]. However, they do not explicitly focus on model views but rather provide general capabilities (often referred to as megamodels) needed to reason about connected models. For instance, languages such as the Epsilon Merging Language [18] may help during the composition process, e.g., by facilitating the identification of the elements to merge. Besides these model composition approaches, some dedicated model view approaches are emerging. In [15] the authors introduce, based on Triple Graph Grammars, the possibility to have non-materialized views by extending base metamodels using inheritance. In a successor work [2], they present an approach for materialized views without modifying the base metamodels, but requiring to explicitly populate the views via model transformations. EMF Facet[9] is another approach to define read-only non-materialized views. Finally, Hegedüs et al. [14] present a similar approach to EMF Facet for model interconnection by augmenting base metamodels with derived features.

To summarize, key challenges to be addressed are the following ones:

- **Genericity**: the view mechanism should be applicable for all modeling languages (i.e., to all metamodels and corresponding models);
- **Expressivity**: a *select-project-join*-like support should be at least provided;
- **Non-intrusiveness**: the view mechanism should be applicable without modifying (e.g., internally) the used modeling languages;
- **Interoperability**: a view should be a regular model from user and tool perspectives;
- **Modifiability**: a view should be changeable as a regular model is;
- **Synchronization**: changes in base models should be directly reflected in the views, and vice versa;
- **Scalability**: view creation and manipulation time should be sufficiently limited, as well as corresponding memory usage.

To the best of our knowledge, none of the available approaches fully satisfies all these characteristics. There is always a trade-off between the offered capabilities and some of these properties, such as scalability or synchronization more particularly. Moreover, no approach provides both inter-model view support and the expected expressivity in terms of view definition. EMF Views intends to tackle these challenges as explained in the following.

[9] `http://www.eclipse.org/modeling/emft/facet`

3 The EMF Views Solution

EMF Views is intended to answer to the concrete need for model views. This section provides an overview of its conceptual framework as well as the related SQL-like DSL it comes with for defining viewpoints.

Before introducing the overall approach itself, we define the terminology used in there. A **viewpoint** is the description of a partitioning and/or restriction of concerns from which systems can be observed. In our modeling context, it consists of a set of concepts coming from one or several metamodels, eventually complemented with some new interconnections between them. A **view** is a representation of a specific system from the perspective of a given viewpoint. In our modeling context, it consists of a set of elements coming from one or several models, eventually complemented with some new interconnections between them. A **virtual model** is a model whose (virtual) elements are just proxies to actual elements contained in other models. The same approach is also applicable at metamodel-level, i.e., a **virtual metamodel**. A **weaving model** is a model that describes links between elements coming from other different models. It conforms to a weaving metamodel that specifies the types of links that can be represented at weaving model-level.

3.1 Conceptual Framework

EMF Views proposes a generic approach allowing to build views on any set of interrelated models that conform to potentially different metamodels. It provides a two-step approach that explicitly separates the specification of viewpoints from the realization and handling of corresponding views. A model view consist of a set of proxy elements, which point to concrete elements from the base models referenced in the view, plus some newly added cross-model relationships between them.

In order to do so, EMF Views relies on a model virtualization approach that is deployed similarly at both metamodel- and model-levels. Thus, such views are actually virtual models that act transparently as regular models via proxies to these interrelated models, but do not duplicate any of the already available data. Each view conforms to a particular viewpoint, which has been previously specified from one or several corresponding metamodels (interconnected together) as a virtual metamodel. Interestingly, the fact that both viewpoints and views are actually virtual (meta)models behaving as normal (meta)models allows for easier viewpoint/view composition. An overview of the EMF Views approach is shown on Figure 1.

At *design time*, designers may specify a new viewpoint by choosing the concerned metamodel(s), listing the relations she/he wants to represent between them (as well as indicating how to eventually compute them at view-level, see hereafter), and identifying the concepts and properties to be selected. This required information is directly collected from the designer/architect, either manually or using our SQL-like DSL (cf. Section 3.2). This input data is stored in a weaving model that is then used by the virtualization mechanism to obtain the actual viewpoint. Therefore, the original metamodel(s) are not modified or polluted by the viewpoint definition. This results in a virtual metamodel, representing the viewpoint, that aggregates several different metamodels according to the given specification.

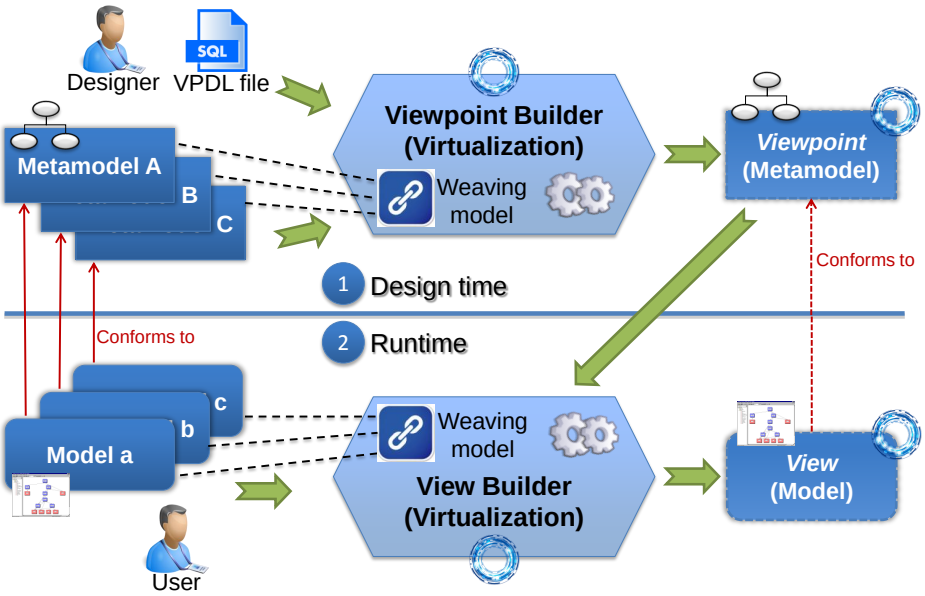

Fig. 1. Overview of the EMF Views approach

Similar to the *select-project-join* operations in relational algebra, the viewpoint definition specifies what classes/features from the contributing metamodels should be part of (or, conversely, filtered out from) the view (*projection*), what conditions model elements will need to satisfy in order to appear as a result in the view query (*selection*) and how the elements from different models should be linked when computing the actual view (*join*).

At *runtime*, once the viewpoint is specified, the user can work on querying and handling views that conform to it. To obtain such a view, she/he can choose the set of input models to be used as input data for the view (and that conform to their respective metamodels, themselves used to create the given viewpoint). With those models and the given viewpoint, EMF Views can build the corresponding view. As described before, the view is represented as a virtual model. In order to create the view, new links have to be established between the underlying models. These links are computed from the rules expressing the combination of the corresponding metamodels at the viewpoint-level (though a manual modification by the user is also possible when needed) by means of a matching engine (cf. Section 4). The links are stored in a separate weaving model associated with the view, without altering the original models neither.

3.2 A SQL-like DSL for Viewpoint Definition

In order to facilitate the definition of viewpoints more easily, EMF Views comes with a DSL strongly inspired from the very well-known SQL language. The choice of an SQL-like language has been quite natural since SQL has already proven its relevance to deal with similar problems in the database community. As said earlier, it also allows

expressing the main needed operations in our model view context, i.e., select, project, and join. Additionally, considering such a widespread language as a base language for our DSL intends to facilitate DSL adoption by potential future users.

Listing 1.1 presents a grammar excerpt of our textual DSL highlighting its four main language features:

- **Create view**: Defines the name of the view(point) as well as the contributing metamodels (and their respective names).
- **Select**: Lists the attributes and relations to be shown (corresponding classes are implicitly selected too). To show all, * is used.
- **From**: Specifies which concepts are going to be linked and provides a name for the new relation (that will be used in the view).
- **Where**: Expresses constraints to filter elements or to define connection between them, using simple mappings or more complex expressions.

Listing 1.1. Partial Grammar of VPDL (ViewPoint Definition Language)

```
Model: "create␣view" viewName=ID "on" metamodel+=Metamodel ("," metamodel+=
    ↪Metamodel)* expression+=Expression;
Metamodel: metamodelURL+=EString "as" metamodelName+=MetamodelName;
Expression: "select" select+=Select "from" from+=From "where" (condition+=
    ↪Condition)+;
Select: select+= "*" | selectFeature+=SelectFeature ("," selectFeature+=
    ↪SelectFeature)*;
SelectFeature: metamodel+=[MetamodelName] "." class+=Class "{" feature+=Feature ("
    ↪," feature+=Feature)* "}";
From: join+=Join ("," join+=Join)* ;
Join: joinLeft+=JoinPart "join" joinRight+=JoinPart "as" reference+=Reference;
JoinPart: metamodel+=[MetamodelName] "." class+=Class;
MetamodelName: name=ID; Class: name=ID; Feature: name=ID; Reference: name=ID;
Condition: ECLExpression;
```

To illustrate our language, Listing 1.2 shows a simple example of a viewpoint specification (from the TEAP scenario as introduced in Section 1). It selects and aggregates some elements from the original metamodels (*select* part) and establishes new relations between them (*from* and *where* parts).

Listing 1.2. Simple example of viewpoint specification in VPDL

```
create view myEnterpriseArchitectureViewpoint on
    "http://www.obeonetwork.org/dsl/togaf/contentfwk/9.0.0" as TOGAF,
    "http://www.omg.org/spec/BPMN/20100524/MODEL-XMI" as BPMN,
    "http://www.omg.org/spec/ReqIF/20110401/reqif.xsd" as REQIF
select TOGAF.Process{name}, BPMN.Process{processType, processCriticality},
       TOGAF.Requirement{rationale}, REQIF.SpecObject{longName}
from TOGAF.Process join BPMN.Process as detailedProcess,
     TOGAF.Requirement join REQIF.SpecObject as detailedRequirement
where TOGAF.Process.name = BPMN.Process.name and TOGAF.Process.isAutomated = false
      and REQIF.SpecObject.values->exists(v | v.theValue = TOGAF.Requirement.name)
```

4 Eclipse-based Tooling Support

EMF Views has been implemented on top of Eclipse and its well-known Eclipse Modeling Framework (EMF) providing general model creation and handling capabilities. It mainly consists of four main components which are notably adapting Virtual EMF [11] in a viewpoint/view context.

Viewpoint and View generators combine the two APIs mentioned thereafter and offer Eclipse GUI components, such as viewpoint- and view-specific creation wizards and

editors. A *Model View API* (deriving from the EMF model access API) supports virtualization to handle viewpoints and views transparently as any regular EMF models. A *Linking API* is managing inter-model links, and has been connected to the mapping engine of the Epsilon Comparison Language [17] to compute such links at view-level. VPDL (cf. previous section) has been developed from scratch with Xtext[10] and then integrated using model-to-text (in Xtend[11]) and model-to-model transformations (in ATL [16]). For user convenience, VPDL notably comes with a proper editor including syntax highlighting and content-assist displaying the applicable classes, attributes, and references. The open source tool and screencasts are available at GitHub[12].

5 Critical Discussions and Next Steps

EMF Views globally fulfils the expected properties introduced in Section 2. *Genericity* is ensured as any existing (meta)models may be considered to build viewpoints and views. The provided DSL and underlying support allow EMF Views to offer the expected *expressivity*. *Non-intrusiveness* is naturally achieved since the original metamodels and models are not modified at all. As mentioned earlier, *interoperability* is guaranteed as model views may be used wherever regular models can (mostly in a read-only mode currently). *Modifiability* is currently partially supported, as only changes to attribute values in views are propagated back to the original models. However, since views and actual models do share the same real instances via a proxy mechanism, *synchronization* at view-level is directly obtained.

To start evaluating *scalability*, we performed empirical experiments focusing on time and memory consumption. We compared EMF Views to a simple composition approach based on the execution of ATL model transformations [16] that copy/merge the elements from the contributing models to the composed model (representing the view). This acted as a representative example of typical behavior from existing composition approaches. We found out that our approach is faster at creating views, and shows only a small overhead during their manipulation (mainly due to implemented lazy loading strategy). Regarding memory usage, in our case the required memory is almost equal to the sum of the size of the contributing models since we do not duplicate model elements for the view but use lightweight proxies. In contrast, in traditional composition the fully duplicated composed model has to be kept in memory as well. We believe these results show that EMF Views scales up to large models and related views.

As further work, we plan to enrich our view mechanism with support for a limited set of aggregation operations, focusing on distributive aggregate functions [21] expressed with OCL extensions [8]. This will be linked to addressing the view update challenge [20], as EMF Views is currently limited to individual attribute updates. At tool-level, we plan to offer the option to persist a snapshot of the model view in case users want to export them to other (external) tools. Finally, we would like to improve usability by exploring alternative languages for view definition and manipulation (e.g., graphical ones) beyond our current SQL-based DSL.

[10] `https://eclipse.org/Xtext`
[11] `http://www.eclipse.org/xtend`
[12] `https://github.com/atlanmod/emfviewsSQL`

Acknowledgment: We thank Juan David Villa Calle and Caue Avila Clasen for their work on past EMF Views versions. This work has been co-funded by the Vienna Business Agency within the COSIMO project (grant number 967327), Christian Doppler Forschungsgesellschaft, and BMWFW, Austria.

References

1. S. Abiteboul and A. Bonner. Objects and views. *SIGMOD Record*, 20(2):238–247, 1991.
2. A. Anjorin, S. Rose, F. Deckwerth, and A. Schürr. Efficient Model Synchronization with View Triple Graph Grammars. In *ECMFA*, 2014.
3. C. Atkinson, D. Stoll, and P. Bostan. Orthographic Software Modeling: A Practical Approach to View-Based Development. In *ENASE*, 2010.
4. H. Balsters. Modelling Database Views with Derived Classes in the UML/OCL-Framework. In *UML*, 2003.
5. Domenik Bork and Dimitris Karagiannis. Model-driven Development of Multi-View Modelling Tools - The MUVIEMOT Approach. In *DATA*, 2014.
6. M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2012.
7. E. Burger. Flexible Views for Rapid Model-Driven Development. In *VAO Workshop*, 2013.
8. J. Cabot, J.-N. Mazón, J. Pardillo, and J. Trujillo. Specifying Aggregation Functions in Multidimensional Models with OCL. In *ER*, 2010.
9. S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks*, 33(1):137–157, 2000.
10. A. Cicchetti, F. Ciccozzi, and T. Leveque. A Hybrid Approach for Multi-View Modeling. *ECEASST*, 50, 2011.
11. C. Clasen, F. Jouault, and J. Cabot. VirtualEMF: A Model Virtualization Tool. In *ER Workshops*, 2011.
12. M. Didonet Del Fabro and P. Valduriez. Towards the Efficient Development of Model Transformations using Model Weaving and Matching Transformations. *Software and Systems Modeling*, 8:305–324, 2009.
13. T. Goldschmidt, S. Becker, and E. Burger. Towards a Tool-Oriented Taxonomy of View-Based Modelling. In *Modellierung*, 2012.
14. A. Hegedüs, A. Horváth, I. Ráth, and D. Varró. Query-Driven Soft Interconnection of EMF Models. In *MODELS*, 2012.
15. J. Jakob, A. Königs, and A.Schürr. Non-materialized Model View Specification with Triple Graph Grammars. In *ICGT*, 2006.
16. F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. ATL: A Model Transformation Tool. *Science of Computer Programming*, 72(1):31–39, 2008.
17. D. Kolovos. Establishing Correspondences between Models with the Epsilon Comparison Language. In *ECMFA*, 2009.
18. D. Kolovos, R. Paige, and F. Polack. Merging Models with the Epsilon Merging Language. In *MODELS*, 2006.
19. M. E. Kramer, E. Burger, and M. Langhammer. View-centric Engineering with Synchronized Heterogeneous Models. In *VAO Workshop*, 2013.
20. E. Mayol and E. Teniente. A Survey of Current Methods for Integrity Constraint Maintenance and View Updating. In *ER Workshops*, 1999.
21. T. Palpanas, R. Sidle, R. Cochrane, and H. Pirahesh. Incremental Maintenance for Non-Distributive Aggregate Functions. In *VLDB*, 2002.
22. J.R. Romero, J.I. Jaen, and A. Vallecillo. Realizing Correspondences in Multi-Viewpoint Specifications. In *EDOC*, 2009.
23. G. Wiederhold. Views, objects, and databases. *IEEE Computer*, 19(12):37–44, 1986.