MDEForge: an extensible Web-based modeling platform*

Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Amleto Di Salle, Ludovico Iovino, and Alfonso Pierantonio

DISIM - University of L'Aquila, Italy
{name.lastname}@univaq.it

Abstract. Model-Driven Engineering (MDE) refers to the systematic use of models as first class entities throughout the software development life cycle. Over the last few years, many MDE technologies have been conceived for developing domain specific modeling languages, and for supporting a wide range of model management activities. However, existing modeling platforms neglect a number of important features that if missed reduce the acceptance and the relevance of MDE in industrial contexts, e.g., the possibility to search and reuse already developed modeling artifacts, and to adopt model management tools as a service.
In this paper we propose MDEForge a novel extensible Web-based modeling platform specifically conceived to foster a community-based modeling repository, which underpins the development, analysis and reuse of modeling artifacts. Moreover, it enables the adoption of model management tools as software-as-a-service that can be remotely used without overwhelming the users with intricate and error-prone installation and configuration procedures.

1 Introduction

Model-Driven Engineering (MDE) refers to the systematic use of models as first class entities throughout the software development life cycle. Model-driven approaches shift development focus from code expressed in third generation programming languages to models expressed in domain-specific modeling languages [1]. MDE increases productivity and reduces time to market by enabling the development of complex systems using models defined with concepts that are much less bound to the underlying implementation technology and much closer to the problem domain.

Over the last few years, many MDE technologies have been conceived for developing domain specific modeling languages, and for supporting a wide range of model management activities. The relevance of MDE is evidenced also by the increasing interest in many scientific endeavours, active technology projects, and numerous industrial projects ranging from direct applications of MDE concepts and tools, to those developing its foundations [2]. Even though existing MDE technologies provide practitioners with facilities that can simplify and automate many steps of model-based development processes, empirical studies show that some barriers still exist for the wider adoption of MDE technologies [3]. Among the main issues that currently hamper a wide adoption of MDE there are at least the following:

* This research was supported by the EU through the Model-Based Social Learning for Public Administrations (Learn Pad) FP7 project (619583).

– the support for discovery and reuse of existing modeling artefacts is very limited. As a result, similar transformations and other model management tools often need to be developed from scratch, thus raising the upfront investment and compromising the productivity benefits of model-based processes. For instance, when modelers identify a need for a domain-specific modeling language, it is quite common to implement it from scratch instead of reusing already developed languages that might satisfy their requirements;
– modelling and model management tools are commonly distributed as software packages that need to be downloaded and installed on client machines, and often on top of complex software development IDEs (e.g. Eclipse).

In this paper we propose MDEForge, an extensible modeling framework specifically conceived to address the issues previously mentioned. In particular, MDEForge consists of a set of core services that permit to store and manage typical modeling artefacts and tools. Atop of such services it is possible to develop extensions adding new functionalities to the platform. All the services can be used by means of a Web access and by a REST API that permits to adopt the available model management tools as software-as-a-service.

The paper is structured as follows: Section 2 discusses the motivations of the paper by considering already existing works. Section 3 presents the architecture of MDEForge and makes and overview of its main components. Two extensions of the platform are presented in Section 4. Section 5 concludes the paper and discusses some research perspectives.

2 Background and Motivation

The artefacts and tools that are typically involved when applying MDE approaches are those shown in Fig. 1: *editors* are used to create *models* that in turn are manipulated to generate other models or even *code*. *Model repositories* are employed to enable the re-use of already specified models. The whole ecosystem is developed according to different kinds of relations (e.g., conformance) with corresponding *metamodels*.

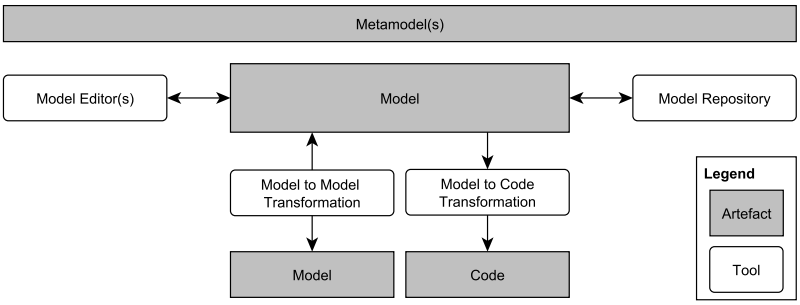

Fig. 1. Main MDE artefacts and tools

	Managed Artefact	**Main purpose**	**Typical deployment scenario**
AMOR [5]	Model	Model versioning	Desktop application
Bizycle [6]	Model	Integration of software components	Desktop application
CDO	Model	Storage	Client-Server application
EMFStore [7]	Model	Model versioning	Client-Server application
GME [8]	Model	Storage	Client-Server application
ModelBus [9]	Model	Model versioning	Client-Server application
Morse [10]	Model	Model versioning	Software-as-a-service
ReMoDD [11]	Any	Documentation	Web-based interaction

Table 1. Overview of existing MDE tools providing storage features

Even though existing modeling platforms (e.g., EMF [4]) provide developers and users with the tools shown in Fig. 1 they neglect a number of important features that if missed the acceptance and the relevance of MDE in industrial contexts might be reduced. In particular, in this paper we focus on the limited support for the re-use of already developed modeling artefacts[1] and for enabling the adoption of modeling tools as-a-service. To this end in this section we overview existing works that are related to MDEForge concerning the re-use of modeling artefacts (Sect. 2.1), and the possibility to use model management tools as a service (Sect. 2.2).

2.1 Reuse of modeling artefacts

In this section we discuss state-of-the-art approaches (see Tab. 1) for providing repositories of modeling artefacts, and outline outstanding research challenges for achieving a comprehensive solution to the problem of properly managing the persistence of models and the deployment and discovery of any kind of model management tools to enable their reuse and refinement.

AMOR - Adaptable Model Versioning [5]: it is an attempt to leverage version control systems in the area of MDE. AMOR supports model conflict detection, and focuses on intelligent conflict resolution by providing techniques for the representation of conflicting modifications as well as suggesting appropriate resolution strategies.

Bizycle [6]: it is a project aiming at supporting the automated integration of software components by means of model-driven techniques and tools. Among the different components of the project, a metadata repository is also provided in order to manage and store all the artefacts required for and generated during integration processes, i.e, external and internal documentation, models and metamodels, transformation rules, generated code, users and roles.

CDO[2]: it is a pure Java model repository for EMF models and meta models. CDO can also serve as a persistence and distribution framework for EMF-based application systems. CDO supports different kinds of deployments such as embedded repositories, offline clones and replicated clusters. However, the typical deployment scenario consists

[1] Hereafter with the terms *modeling artefacts* we include also *modeling tools*.

[2] `http://www.eclipse.org/cdo/`

of a server managing the persistence of the models by exploiting all kinds of database backends (like major relational databases or NoSQL databases), and an EMF client application.

EMFStore [7]: it is a software configuration management system tailored to the specific requirements of versioning models. It is based on the Eclipse Modeling Framework and it is an implementation of a generic operation-based version control system. EMFStore implements, in the modeling domain, the typical operations implemented by SVN, CVS, Git for text-based artefacts, i.e., change tracking, conflict detection, merging and versioning. It consists of a server and a client component. The server runs standalone and provides a repository for models including versioning, persistence and access control. The client component is usually integrated into an application and is responsible for tracking changes on the model, and for committing, updating and merging.

GME - Generic Modeling Environment [8]: it is a set of tools supporting the creation of domain specific modeling languages and code generation environments. A repository layer is also provided to store the developed models. Currently, MS Repository (an object oriented layer on top of MS SQL Server or MS Access) and a proprietary binary file format are supported.

ModelBus [9]: it consists of a central bus-like communication infrastructure, a number of core services and a set of additional management tools. Depending on the usage scenario at hand, different development tools can be connected to the bus via tool adapters. Once a tool has been successfully plugged in, its functionality immediately becomes available to others as a service. Alternatively, it can make use of services already present on the ModelBus. Among the available services, ModelBus also includes a built-in model repository, which is able to version models, supports the partial checkout of models and coordinates the merging of model versions and model fragments;

Morse - Model-Aware Repository and Service Environment [10]: it is a service-based environment for the storage and retrieval of models and model-instances at both design- and run-time. Models, and model elements are identified by Universally Unique Identifiers (UUID) and stored and managed in the Morse repository. The Morse repository provides versioning capabilities so that models can be manipulated at runtime and new and old versions of the models can be maintained in parallel;

ReMoDD - Repository for Model-Driven Development [11]: it is a repository of artefacts aiming at improving MDE research and industrial productivity, and learning experience of MDE students. By means of a Drupal Web application, users can contribute MDE case studies, examples of models, metamodels, model transformations, descriptions of modeling practices and experience, and modeling exercises and problems that can be used to develop classroom assignments and projects. Searching and browsing facilities are enabled by a Web-based user interface that also provides community-oriented features such as discussion groups and a forum.

According to Tab. 1 the majority of existing approaches focus on providing support for the persistence of models. Only ReMoDD supports other kinds of modeling artefacts, like transformations, and metamodels. However, the main goal of ReMoDD is to support learning activities by providing documentation for each stored artefact. Consequently, ReMoDD cannot be used to programmatically retrieve artefacts from the

repository or more generally cannot be adopted as software-as-a-service to search and reuse already existing modeling artefacts. Most of the discussed approaches require local installation and configuration. Only ReMoDD and Morse do not require to be installed locally. In particular, the modeling artefacts stored in ReMoDD can be searched and browsed through a Web-based application. Morse provides developers with the possibility to use it as a service.

2.2 Model management tools as service

The motivation or our work is shared also in [12] where authors propose the Modeling as a Service (MaaS) initiative as an approach to deploy and execute model-driven services over the Internet. This initiative is aligned with SaaS principles, since consumers do not manage the underlying cloud infrastructure and deal mostly with end-user systems. Interestingly, in [13] authors investigate the problem of transforming very large models in the Cloud by addressing two phases: i) model storage, and ii) model transformations execution in the Cloud. For both aspect authors identify a set of research questions, and possible solutions.

Even though there are different attempts [14] and projects (e.g., the EU MONDO project[3]) related to the adoption of Cloud infrastructures to apply MDE, the area is still mostly unexplored. In line with the MaaS initiative we want to contribute to this research area with an extensible modeling framework that enables the adoption of model management and analysis tools as service. As discussed later in the paper the framework is at its early stages and we have not addressed yet aspects like workload and capacity management that are typical in Cloud computing, however the results we have obtained so far are promising.

3 Overview of the MDEForge platform

In this section we present the MDEForge platform that has been conceived to overcome the issues discussed in the previous section. In particular MDEForge aims at:

- providing a community-based modeling repository, which underpins the devolpement, analysis and reuse of any kinds of modeling artifacts not limited to only models;
- supporting advanced mechanisms to query the repository and find the required modeling artifacts;
- enabling the adoption of model management tools as software-as-a-service;
- being modular and extendible;

As shown in Fig. 2 the MDEForge platform consists of a number of services that can be used by means of both a Web access and programmatic interfaces (API) that enable their adoption as software as a service. In particular, *core* services are provided to enable the management of modeling artifacts, namely transformations, models, metamodels, and editors. Atop of such core services, extensions can be developed to add new functionalities. For instance, by exploiting the transformation and metamodel services

[3] http://www.mondo-project.org/

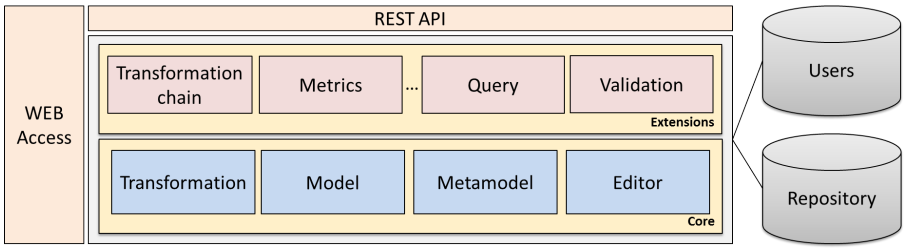

Fig. 2. Architecture of MDEForge

it is possible to extend the platform by adding a new service to enable the automated chaining of model transformations as discussed in Section 4.

We envision different kinds of users of the MDEForge platform and in our opinion they can be at least the following:

- *Developers of modeling artifacts:* as previously said we envision a community of users that might want to share their tools and enable their adoption and refinement by other users. To this end the platform provides the means to add new modeling artifacts to the MDEForge repository;
- *Developers of MDEForge extensions:* one of the requirements we identified when we started the development of MDEForge is about the modularity and extensibility of the platform. To this end we identified a set of core services that can be used to add new functionalities by means of platform extensions. In this respect, experienced users might contribute by proposing new extensions to be included in the platform;
- *End-users:* a Web application enables end-users to search and use (meta)models, transformations, and editors available in the MDEForge repository. Experienced users might use the REST API to exploit the functionalities provided by the platform in a programmatic way. For instance, tool vendors might exploit the functionalities provided by their tools by exploiting some of the transformations available in the MDEForge repository.

In the remainder of this section we give some details about the MDEForge repository (Section 3.1) and the available core services (Section 3.2).

3.1 The MDEForge Repository

The *Repository* component plays a key role in the MDEForge platform and it has been developed in order to store artifacts according to the metamodel shown in Fig. 3. In particular, the repository has been developed with the aim of managing any kinds of modeling artifacts (see the metaclass `Artifact` in Fig. 3). Each artifact refers to the corresponding type, e.g., model, transformation, metamodel, etc. The specification of the relation between a given artifact and the corresponding type is done by means of the `Relation` elements. In turn, each relation is typed by means of a corresponding `RelationType` element. By means of such modeling constructs it is possible e.g., to specify the *conformsTo* relation between a model *m1* and the corresponding metamodel *MM1* as shown in Fig. 4. Similarly, it is possible to specify any kinds of modeling

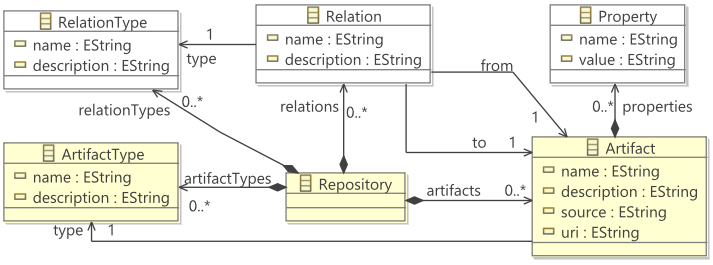

Fig. 3. Fragment of the MDEForge Repository Metamodel

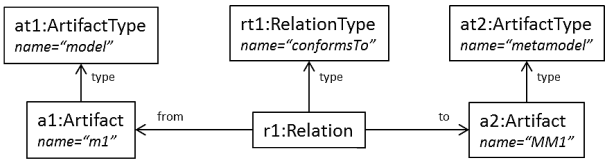

Fig. 4. Simple content of the Repository

elements together with their relations. For example, it is possible to represent also the execution engine of a given model transformation stored in the repository.

It is important to remark that all the artifacts stored and managed by the MDEForge platform are related to the users that created them. The system permits also to make artifacts public, private, or limit their visibility to specific users.

3.2 The MDEForge Core

In the following the core service previously mentioned and shown Fig. 5 are described. *Model Service:* it manages models in the repository, thus it permits users to upload, download, delete models, and even search models conforming to a specific metamodel; *Metamodel Service:* it provides the means to upload, download, delete metamodels, and find metamodels by a specific Universal Resource Identified (URI);
Transformation Service: it manages the transformations in the repository, i.e., it permits to upload, download, delete transformations. Moreover, it permits to remotely execute

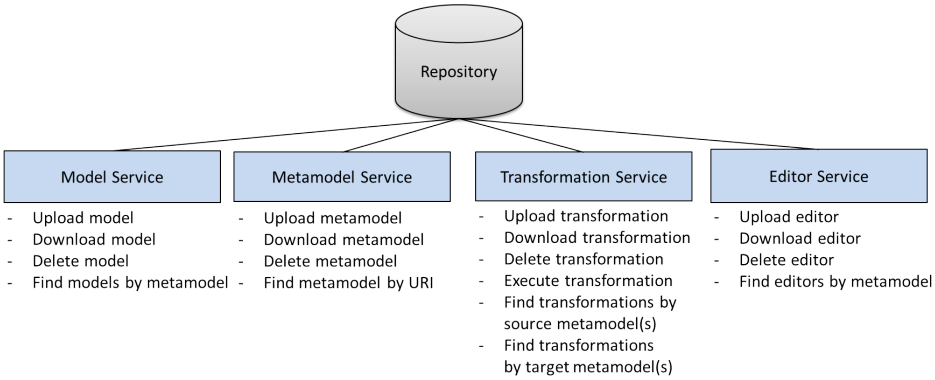

Fig. 5. Overview of the MDEForge Core services

transformations, or even find transformations by specifying the source and/or target metamodel(s);
Editor Service: it permits to upload, download, or delete editors from the repository. It is important to remark that at this stage of the project we manage the JARs containing the implementation of a given editor (e.g., developed by means of GMF [15]). Consequently, the editor service permits users to upload the implementation code of a given repository to enable other users to find, download, and in case install them locally. In other words, at this stage of the project we do not refer to on-line and collaborative editors even though this represents an important extension of the platform that we intend to investigate in the near future.

A prototypical implementation of the MDEForge has been developed[4] and even though it is still at an early stage, we managed to apply the platform to support two interesting problems: chaining model transformations, and measuring metamodels as discussed in the next section.

4 Examples of MDEForge extensions

In this section we present two examples of extensions we have developed atop of the core services of the platform and that have been successfully applied to deal with the problems of chaining of model transformations (Section 4.1) and to calculate metrics of metamodels to support the understanding of their characteristics (Section 4.2).

4.1 Automated chaining of model transformations

In [16] we have addressed the problem of automatically composing model transformations. In particular, we have proposed an approach to automatically discover and compose transformations: developers provide the system with the source models and specify the target metamodel. Then, by relying on the MDEForge repository, all the possible transformation chains are calculated. Importantly, in case of incompatible intermediate target and source metamodels, proper adapters are automatically generated in order to chain also transformations that otherwise would be discarded by limiting the reuse possibilities of available transformations.

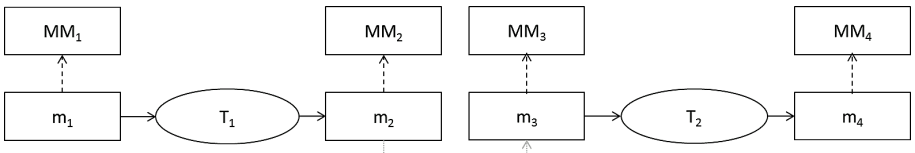

Fig. 6. Model transformation chain example

Figure 6 shows an explanatory model transformation chain. In particular, T_1 is a model transformation that generates models conforming to the target metamodel MM_2 from models conforming to MM_1. Additionally, T_2 is a model transformation that generates models conforming to MM_4 from models conforming to the source metamodel MM_3. In general, if the input metamodel of T_2 would be also the output metamodel of

[4] www.mdeforge.org

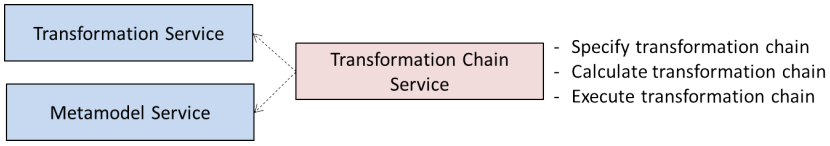

Fig. 7. Transformation Chain extension

T_1, then these two transformations could be chained. However, under certain conditions, two transformations can be chained even though the output metamodel of the first transformation does not correspond to the input metamodel of the second transformation.

To support the techniques presented [16] we have developed the extension shown in Fig. 7 consisting of the *Transformation Chain* service that makes use of the *Transformation* and *Metamodel* core services discussed in the previous section. In particular, users specify the model to be transformed, the target metamodel by means of the *Specify transformation chain* operation. Such an input is used to calculate the transformation chains that subsequently can be executed.

4.2 Measuring metamodels

In [17] we have developed an approach to measure metamodels with the aim of understanding their typical characteristics by investigating the correlations of different metrics applied on a corpus of more than 450 metamodels. In particular, we proposed an approach for *a)* measuring certain metamodeling aspects (e.g., abstraction, inheritance, and composition) that modelers typically use; and *b)* for revealing what are the common characteristics in metamodeling that can increase the complexity of metamodels hampering their adoption and evolution in modeling ecosystems.

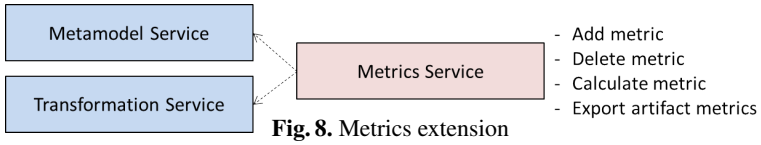

Fig. 8. Metrics extension

To perform such analysis we have developed an extension of the MDEForge platform to support the calculation of metrics as shown in Fig. 8. In [17] we discussed metrics calculated only on metamodels, however we are working also on the identification of possible correlations among transformation and metamodel characteristics. Finally, to enable the management of the calculated metrics, the added service permit to export CSV files encoding the values of all the calculated metrics. Generating CSV files enables the adoption of statistical tools like IBM SPSS, Microsoft Excel, and Libreoffice Calc for subsequent analysis of the generated data.

5 Conclusion and future works

In this paper we presented MDEForge, an extensible modeling framework supporting the creation of a community-based modeling repository, which underpins the development, analysis and reuse of modeling artifacts. The platform consists of core service that can be extended and all of them are remotely available as software as a service thus users are not overwhelmed with intricate and error-prone installation and configuration procedures. Two concrete extensions and applications of the platform have

been presented. As future work we intend to investigate issues that are typical in Cloud computing, e.g., scalability of the platform, and workload management. Moreover, we intend to implement further extensions for instance to support advanced queries on the repository. To this end we intent to investigate the integration of tools that have been recently proposed [18]. Moreover, we plan to extend the platform by adding services enabling collaborative modeling activities.

References

1. Schmidt, D.C.: Guest NOOPeditor's Introduction: Model-Driven Engineering. Computer **39** (2006) 25–31
2. Di Ruscio, D., Paige, R.F., Pierantonio, A.: Guest editorial to the special issue on Success Stories in Model Driven Engineering. Science of Computer Progr. (2014)
3. Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R.: Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In: MODELS. Volume 8107 of LNCS. Springer Berlin Heidelberg (2013) 1–17
4. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.J.: Eclipse Modeling Framework. Addison Wesley (2003)
5. Brosch, P., Langer, P., Seidl, M., Wimmer, M.: Towards End-user Adaptable Model Versioning: The By-Example Operation Recorder. In: Procs.of CVSM '09, Washington, DC, USA, IEEE Computer Society (2009) 55–60
6. Kutsche, R., Milanovic, N., Bauhoff, G., Baum, T., Cartsburg, M., Kumpe, D., Widiker, J.: BIZYCLE: Model-based Interoperability Platform for Software and Data Integration. In: Procs.of the MDTPI at ECMDA. (2008)
7. Koegel, M., Helming, J.: EMFStore: a model repository for EMF models. In: Software Engineering, 2010 ACM/IEEE 32nd Int. Conf. on. Volume 2. (2010) 307–308
8. Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment. In: Workshop on Intelligent Signal Processing. (2001)
9. Hein, C., Ritter, T., Wagner, M.: Model-driven tool integration with ModelBus. Workshop Future Trends of Model-Driven (2009)
10. Holmes, T., Zdun, U., Dustdar, S.: Automating the Management and Versioning of Service Models at Runtime to Support Service Monitoring. In: EDOC. (2012) 211–218
11. France, R., Bieman, J., Cheng, B.: Repository for Model Driven Development (ReMoDD). In: Models in Software Engineering. Volume 4364 of LNCS. Springer Berlin Heidelberg (2007) 311–317
12. Brunelière, H., Cabot, J., Jouault, F.: Combining Model-Driven Engineering and Cloud Computing. In: MDA4ServiceCloud'10 Workshop co-located with ECMFA. (2010)
13. Clasen, C., Didonet Del Fabro, M., Tisi, M.: Transforming Very Large Models in the Cloud: a Research Roadmap. In: First International Workshop on Model-Driven Engineering on and for the Cloud, Copenhagen, Danemark, Springer (2012)
14. Paige, R., Cabot, J., Brambilla, M., Chechik, M., Mohagheghi, P.: Procs. of CloudMDE - First Workshop on MDE for and in the Cloud. (2012)
15. Eclipse: Graphical Modeling Framework. `http://www.eclipse.org/gmf/` (2014)
16. Basciani, F., Di Ruscio, D., Iovino, L., Pierantonio, A.: Automated Chaining of Model Transformations with Incompatible Metamodels. In: Procs. MODELS 2014 Accepted. (2014)
17. Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: Mining metrics for understanding metamodel characteristics. In: MiSE 2014 - ICSE Workshop. (2014)
18. Szárnyas, G., Izsó, B., Ráth, I., Harmath, D., Bergmann, G., Varró, D.: IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud. In: Procs. MODELS 2014, Valencia, Spain, Springer, Springer (2014) Accepted.