Collaborative Repositories in Model Driven Engineering

Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, Alfonso Pierantonio
Dipartimento di Ingegneria e Scienze dell’Informazione e Matematica
Università degli Studi dell’Aquila Via Vetoio, L’Aquila, Italy {name.surname}@univaq.it

Abstract: Model-Driven Engineering (MDE) promotes the systematic use of models as first-class abstractions throughout the software development lifecycle. Over the last few years, many MDE techniques and platforms 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 are essential for the acceptance and the relevance of MDE in industrial contexts, e.g., the possibility to search and reuse already developed artifacts in any stage of the development processes. Recently, many repositories have been proposed in response to the need of the MDE community for advanced systems that support the reuse of modeling artifacts. This paper outlines the opportunities related to the adoption of model repositories and discuss the challenges that still have to be addressed.

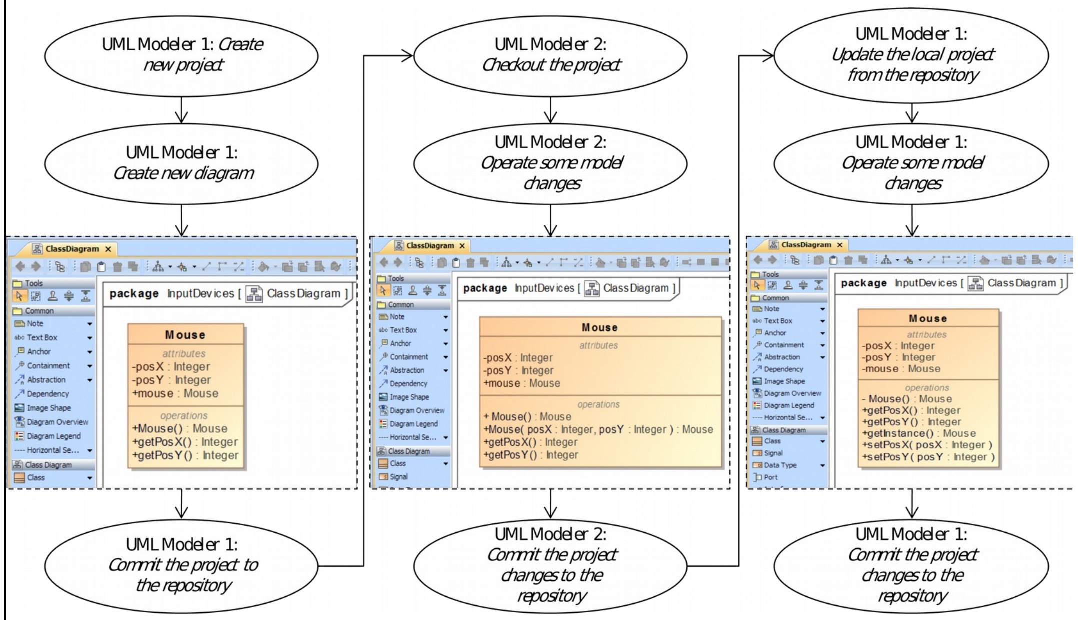
Model Driven Engineering

Model-Driven Engineering (MDE) promotes the systematic use of models as first-class abstractions throughout the software development lifecycle. This permits to leverage abstraction by shifting development focus from third generation programming languages code to design models expressed in domain-specific modelling languages. The objective is to increase productivity and reduce 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.
Several model-driven platforms and tools are available to simplify and automate many steps of MDE approaches. Commercial tools like Enterprise Architect, Visual Paradigm, MagicDraw, and Rhapsody enabled the adoption of MDE for developing complex software systems in key industrial domains including automotive, avionics, railways, and other embedded applications. According to recent studies [WHR13,WHR14] the adoption of MDE in industry can have a positive impact on productivity, maintainability, and traceability by relying on several artifacts that are typically produced when adopting MDE (e.g., domain specific modeling languages, model transformations, code generators, model for testing and simulation purposes or simply for team communications). Even though existing modeling tools provide developers and users with features able to simplify and automate many steps of model-based development processes, empirical studies show that barriers still exist making a wider adoption of MDE technologies difficult [DPP14]. In particular, to deal with the growing complexity of software systems, it is necessary to enforce consistent reuse and leverage the interconnection of the modeling artifacts that are produced and consumed during the different phases of the applied development processes. This aspect has been acknowledged by tool vendors that have been investing in the development of model repositories enabling collaborative modeling supports and permitting team members to checkout, commit, and update software models. This is similar to what happens in the case of source code development that is supported, especially for complex systems, by version control systems.

This article outlines different model repositories that have been proposed over the last years. Each tool has been proposed to support specific needs, e.g., collaborative modeling, the possibility to use different modeling tools in software lifecycle management, to enable tools interoperability, to increase the reuse of already developed models, and to enable the integration of heterogeneous models. It is worth mentioning that in this paper, aspects like model versioning, and conflict handling are not directly considered since we see them as problems and activities that occur at a higher-level of abstraction. To make a similitude, model repositories as considered in this paper are similar to cloud storage services like Dropbox and Google Drive. Such systems do not directly offer advanced conflict management features that instead additional or external services might provide by properly using and extending the public APIs of such public storage systems.

Sidebar

There are two modelers working in different sites, and that want to contribute to the specification of the same model. Modelling environments like MagicDraw permit modellers to locally download models that are stored in teamwork servers and then commit changes once the editing phases are completed. *UML Modeler 1(UM1)* starts first by creating an initial version of the model consisting of the class *Mouse* including its attributes and operations. Once completed the editing session, *UM1* commits the changes to the model repository used by the considered business unit. In this way, *UML Modeler 2 (UM2)* working in the same business unit of *UM1* and that has access to the repository can contribute to the specification of the model. To this hand *UM2* performs a checkout from the repository, and thus creates a local copy of the model. *UM2* operates changes to the model with the aim of introducing the singleton design pattern. After the editing section, *UM2* commits the changes to the repository. *UM1* refreshes the working copy of the model by merging changes that *UM2* has made and committed to the server. *UM1* changes the model by properly specifying the singleton design pattern for the class *Mouse* that was wrongly introduced by *UM2.* Once the changes have been performed, *UM1* commits the model to the repository.


UML Modeler 1: Create new project
UML Modeler 1: Create new diagram
UML Modeler 1: Commit the project to the repository
UML Modeler 2: Checkout the project
UML Modeler 2: Operate some model changes
UML Modeler 2: Commit the project changes to the repository
UML Modeler 1: Update the local project from the repository
UML Modeler 1: Operate some model changes
UML Modeler 1: Commit the project changes to the repository
package InputDevices [ClassDiagram]
Mouse
attributes
-posX : Integer
-posY : Integer
+mouse : Mouse
operations
+Mouse() : Mouse
+getPosX() : Integer
+getPosY() : Integer
+Mouse(posX : Integer, posY : Integer) : Mouse
- Mouse() : Mouse
+getInstance() : Mouse
+setPosX(posX : Integer)
+setPosY(posY : Integer)

Most of the modeling tools adopt a pessimistic scheme, i.e., modelers can lock specific model fragments in order to avoid that other modellers having access to the same parts of the considered model might operate conflicting model changes. Other systems like GenMyModel provide modellers with online collaborative features that are becoming familiar and widely used via popular software-as-a-service tools such as Google Drive and Apple iCloud.

Model Repositories in Model Driven Engineering: opportunities and challenges

Currently available support for discovering and reusing developed modelling artefacts is limited. As a result, similar metamodels, transformations, code generators, and other model management systems often need to be redeveloped from scratch, thus raising the upfront investment and compromising the productivity benefits of model-based processes. Consequently, the availability of advanced model repositories can be key to success and give the possibility to take advantage of many opportunities as summarized in the following.

Opportunities

The *opportunities* related to the availability of model repositories in MDE are manifold and it is possible to mention at least the following ones:

- *Community of users:* multitudes of users and developers can share developed models and tools and consequently their adoption and extension by other users are enabled. Thus, as typically happens for open source projects, more agile processes can be applied to develop models in collaborative ways.
- *Availability of certified models for verification and validation purposes:* especially in the domain of business-critical software, the need for trusted modeling artefacts is more pressing than ever. The intrinsic quality of new model-based tools is typically enhanced by starting the development from repositories of certified models, which can ease the difficult task of verification and validation.
- *Availability of modeling artifacts for learning purposes:* modeling can be a difficult task usually requiring very experienced modelers and domain experts. Already developed models represent valuable know-how since they capture domain insights and knowledge that might be conveyed to new modelers in order to let them become familiar with relevant modeling patterns and practices typically employed in the considered business contexts.
- *Tool interoperability/heterogeneity:* when developing and managing complex software systems different tools are used to create and manage models at different levels of abstraction and in heterogeneous formats. Such tool misalignment is a great obstacle to interoperability and can be solved by making use of common repositories to store and retrieve developed artifacts by means of standardized APIs.
- *Collaborative modeling:* several existing modeling environments inhibit collaborative modeling and hence prevent teams of technical and non-technical stakeholders from working together (in a distributed environment) to specify and evaluate alternative representations of their problem domain. This is mainly due to the fact that modeling environments are typical desktop-based and are able to manage modeling artifacts that are locally stored. The availability of online repositories can give modelers the possibility to collaboratively work on common modeling artifacts and share them.
- *Traceability:* having a common repository permits the creation and maintenance of easy-to-use cross-links relating a comprehensive range of heterogeneous artifacts produced at different stages of the software lifecycle. This is an influential aspect in adopting advanced repositories and weights a lot in assessing the significance of different genres of change impact.

While the benefits of adopting model repositories have been widely recognized, they have been not yet achieved because there are still a number of challenges to be met by existing tools as discussed in the next sections.

Challenges

In order to fully take advantage of the previously discussed opportunities, a number of challenges still have to be addressed. They can be distinguished among technical and non-technical ones as discussed in the following.

Technical challenges:

- *Management of different kinds of modeling artifacts:* most of the existing model repositories provide persistence options to models only. Consequently, reusability of other modeling artifacts and tools like modeling editors, model transformations, and code generators is not supported.
- *Advanced query mechanisms:* advanced query techniques are of crucial relevance to retrieve artifacts according to different criteria. For instance, models can be searched by considering the corresponding metamodels, domain type, the particular development phase, by exploiting some tagging mechanisms, or even querying the repository (or a part of it) by means of logical predicates (e.g., OCL).
- *Model management and analysis tools as service (MaaS):* 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). Since this can often be a burden, particularly for non-technical stakeholders with average IT skills, it is necessary to have cloud-based installations of model repositories and give the possibility to remotely use the stored modeling artifacts over the net.

- *Extension mechanisms:* model repositories should provide core functionalities and extension mechanisms, which permit to build new applications in order to manage new kinds of modeling artifacts and to provide additional services.
- *Federation of model repositories:* each organization might have its private repositories able to interact with public ones in order to retrieve modeling artifacts that are publicly available. Consequently, federation mechanisms have to be properly conceived in order to seamlessly aggregate modeling artifacts with respect to specific access and licensing policies and to ensure business entities that they would remain the owner of the produced modeling artifacts while still benefiting the use of other model repositories.

An aspect, which is orthogonal to all the previous challenge, is *scalability.* In particular, as MDE is increasingly applied to larger and more complex systems, current generation of model management technologies are reaching their limits and a new line of research is required in order to achieve scalability and to enable efficient management and persistence of models larger than hundreds of megabytes in size, thus to cope with industry-scale artifacts.

<u>Conceptual challenges:</u>
Further than the challenges previously mentioned, there are also non-technical ones that necessarily have to be met in order to fully benefit from the adoption of model repositories. In particular:

- *Incentives to share modeling artifacts:* there are several public model repositories around, however keeping them alive and solicit contributions from user communities is a hard task especially because each business entity might not see any benefit from sharing their own developed artifacts. Consequently, it is necessary to conceive rewarding mechanisms that can motivate users to share their artifacts.
- *Licensing related to the shared artifacts:* it is necessary to manage the intellectual property of the shared artifacts. Similarly to what occurs in the domain of open source software, it is necessary to identify, assess, enforce, and inferring licensing schemes under which modeling artifacts are uploaded and maintained in model repositories.
- *Guidelines to manage the sharing of artifacts and to keep their quality under control:* users that would like to reuse artifacts available in repositories should be somehow guaranteed that the available artifacts satisfy quality requirements that has to be defined for each kind of modeling artifacts. To this end, the sharing phase has to be moderated in a way that users upload their artifacts, which are analyzed and tested before making them available to other users.

An overview of existing model repositories

Over the last years several repositories have been proposed both from industry and academia. Establishing how many (active) repositories are currently available is not easy, especially because, as mentioned in the previous section, keeping a model repository growing and operative is a difficult task. In addition, it is not rare to store modeling artifacts in repositories which were originally used in adjacent domains like enterprise architecture. In the following some of the most representative repositories are overviewed and they are summarized in Table 1 with respect to the technical challenges presented in the previous section.

Rational Rhapsody Designer Manager is a collaborative design management software that helps teams and their stakeholders to share, trace, review and manage designs. The system includes a repository that teams can use to store, distribute, search and manage design models by means of a web-based access.
Enterprise Architect is a comprehensive UML analysis and design tool for UML, SysML, BPMN and many other technologies. Covering software development from requirements gathering through to the analysis stages, design models, testing and maintenance. EA is a multi-user, Windows-based, graphical tool and offers functionalities for sharing projects in team-based and distributed development environments.
Visual Paradigm is a UML CASE Tool supporting UML 2, SysML and Business Process Modeling Notation (BPMN) from the Object Management Group (OMG). In addition to collaborative modeling support, it includes a model repositories and mechanisms for conflict detection and resolution.

MagicDraw is a modeling tool supporting different stages of the development process and provides also modelers with teamwork support for pessimistic version control. It includes also reverse engineering mechanisms, as well as database schema modeling, and DDL generation.
Modelio (the open source modeling environment) is an open source modeling environment supporting different standard i.e. UML2, BPMN2, XMI, MDA, SysML, TOGAF, SoaML, UML Testing Profile. Modelio provides modelers with a centralized database for storing models, managing model consistency rules, and traceability links.
GenMyModel is an online system providing the possibility to edit UML diagrams on the cloud by means of an advanced online modeling tool. The modeling editor enables also the sharing of models with other users of the system or even on social networks.
CDO (Connected Data Objects) is based on a 3-tier architecture supporting EMF-based client applications, connecting to a central model repository server and leveraging different types of pluggable data storage back-ends like relational databases, object databases and file systems. It is also a model runtime environment with a focus on orthogonal aspects like model scalability, transactionality, persistence, distribution, queries and other services.
EMFStore is an operation-based version control system for models based on the Eclipse Modeling Framework. It provides change tracking, conflict detection, merging and versioning of models. Also it includes functionalities of repository mining for data extraction.
MORSE (Model-Aware Service) consists of a model repository that manages model-driven projects and artifacts, and model-aware services that interact with the repository for performing reflective queries on the models stored in the repository. The goal of MORSE is facilitate services to dynamically reflect on models, creating model-aware services (and components) for the SOA and support these with a model repository. Each model of the SOA is placed in the model repository. Moreover MORSE supports services and processes integrated with model-aware services, then the traceability information is managed and transmitted by the entire process.
MODELBUS is a model-driven tool integration framework, which permits to build seamlessly integrated tool environments for development processs. ModelBus allows the integration of heterogeneous tools vering the whole development lifecycle and comprising custom-made and proprietary tool. With the help of specific adapters it is possible to connect tools to the ModelBus allowing them to share data via models and functionality via services.
The main goal of *AMOR (Adaptable Model Versioning)* is the support of mechanisms to leverage version control techniques in the area of MDE. AMOR intends to achieve precise conflict detection and resolution, and it considers knowledge about the type of modifications the models have undergone in the course of their evolution and knowledge about the semantics of the modeling concepts.
ReMoDD (Repository for Model-Driven Development) is collecting documented MDD case studies, examples of models reflecting good and bad modeling practices, reference models (including metamodels) that can be used as the basis for comparing and evaluating MDD techniques, generic models and transformations reflecting reusable modeling experience, descriptions of modeling techniques, practices and experiences, and modeling exercises and problems that can be used to develop classroom assignments and projects. ReMoDD developers are working on an API to enable external tools to retrieve artifacts from the repository directly and a web interface will be provided too.
GME (Generic Modeling Environment) is a configurable toolset that supports the easy creation of domain-specific modeling and program synthesis environments. The metamodels specifying the modeling paradigm are used to automatically generate the target domain-specific environment. The generated domain-specific environment is then used to build domain models that are stored in a model database or in XML format. The thin storage layer includes components for the different storage formats. Currently, a fast proprietary binary file format and an XML format are supported.

Although a detailed comparison is out of the scope of this article, we wish to stress that recently different model repositories have been proposed both by academia and industry. Next section overviews an extensible repository, named MDEForge, which has been conceived at the Computer Science Department of the University of L'Aquila.

	Managed artifacts	**Query mechanism**	**MaaS**	**Extension mechanism**	**Federation mechanism**	**Collaborative modeling support**	**Tools interoperability support**	**Academic (A) / Industrial (I)**

Rational Rhapsody Designer Manager www.ibm.com	SysML, UML model	Advanced query engine	Yes	Yes	No	Yes	XMI-based	I
Enterprise Architect http://www.sparxsystems.com.au	SysML, UML model	No	Yes	Yes	No	Yes	XMI-based	I
Visual Paradigm http://www.visual-paradigm.com	UML model	No	Yes	Yes	No	Yes	XMI-based	I
MagicDraw http://www.nomagic.com	UML model	No	Yes	Yes	No	Yes	XMI-based	I
Modelio http://www.modelio.org	models	Based on a dedicated teamwork manager module	No	Yes	Yes	Yes	XMI-based	I
GenMyModel https://www.genmymodel.com	UML models	Web based	Yes	No	No	Yes	XMI-based	I
CDO http://www.eclipse.org/cdo	models metamodels	Keyword-based	Yes	Yes	No	Yes	EMF-based	I
EMFStore http://eclipse.org/emfstore	models metamodels	Keyword-based	Yes	No	No	Yes	EMF-based	I
MORSE http://www.infosys.tuwien.ac.at	models	Structural	Yes	Yes	No	Yes	Service-based	I
MODELBUS http://www.modelbus.org	models metamodels transformations	Browsing	Yes	Yes	No	Yes	Adapter-based	I
AMOR http://www.modelversioning.org	models metamodels	No	No	Yes	No	Yes	XMI-based	A
ReMoDD http://www.cs.colostate.edu/remodd	models metamodels transformations	Web based	Yes	No	No	No	No	A
GME http://www.isis.vanderbilt.edu/projects/gme	metamodels, models	Browsing	No	Yes	No	No	Based on the COM component model	Both
MDEForge http://www.mdeforge.org	models, metamodels, transformations, editors.	Tag based	Yes	Yes	Not yet	No	REST API based	A

Table 1: A sample of model repositories. This list is inevitably non-exhaustive and merely intended to reflect what kinds of systems are available

MDEForge: an extensible Web-based model repository

MDEForge [BDD14] has been conceived to deal with the challenges discussed in the previous sections. In particular MDEForge aims at:

- providing a community-based modeling repository, which underpins the development, 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. 1 the MDEForge platform consists 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.

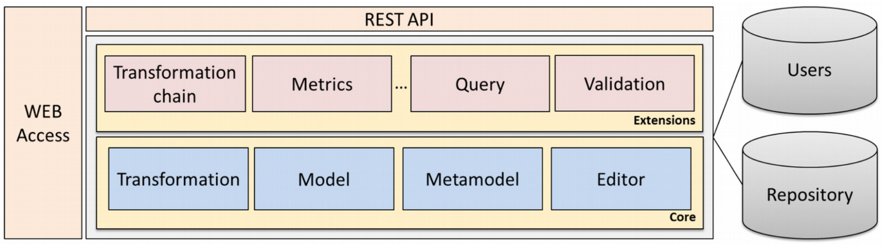


Figure 1: MDEForge overview

Conclusions

The productivity and quality benefits of MDE are testified by several industrial experiences in different application domains including automotive, space industries, embedded systems, and home automation. In order to deal with the complexity of large systems, and the consequent specification and manipulation of corresponding models, increasingly modeling teams have been started to adopt model repositories. They permit different modelers to collaboratively work to the same modeling activities, to use different tools by managing their interoperability, and to reuse and merge already developed model fragments. To completely take advantage of the opportunities related to the adoption of model repositories there are some issues that have to be still addressed. For instance, it is necessary to have advanced mechanisms to query repositories even in case of stored artifacts of megabytes in size. Importantly, it is necessary to define inter-repositories collaboration models, in order to define in a precise way how different repositories can be federated to collaborate and share their stored artefacts with each other.

References

[BDD14] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Amleto Di Salle, Ludovico Iovino and Alfonso Pierantonio, MDEForge: an extensible Web-based modeling platform, in: CloudMDE Workshop at MoDELS 2014, Valencia, Spain, http://ceur-ws.org/, 2014

[WHR13] Jon Whittle, John Hutchinson, Mark Rouncefield, Hkan Burden, and Rogardt Heldal. Industrial adoption of model-driven engineering: Are the tools really the problem? In Model-Driven Engineering Languages and Systems, volume 8107 of LNCS, pages 1–17. Springer Berlin Heidelberg, 2013.

[WHR14] Whittle, J.; Hutchinson, J.; Rouncefield, M., "The State of Practice in Model-Driven Engineering," Software, IEEE , vol.31, no.3, pp.79,85, May-June 2014