

MoDisco in a Nutshell! How to Deal with your IT Legacy? Reverse Engineering using Models...

Hugo Bruneliere

▶ **To cite this version:**

Hugo Bruneliere. MoDisco in a Nutshell! How to Deal with your IT Legacy? Reverse Engineering using Models.... JavaTech Journal, 2011, 10, p. 21-24. hal-00615269

HAL Id: hal-00615269
https://hal.science/hal-00615269

Submitted on 11 Feb 2016

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.

Eclipse Indigo Simultaneous Release

MoDisco in a nutshell!

How to deal with your IT Legacy? Reverse Engineering using models…

For the second time, the MoDisco project is integrated in the yearly Eclipse Simultaneous Release. After Helios last year with its version 0.8.0, MoDisco version 0.9.0 is now officially released as part of Indigo. In addition to the already existing reverse engineering features (notably for Java), upgraded for the occasion, MoDisco comes this year with some interesting brand new components and dedicated JEE technology support. Of course, still model-driven…

By Hugo Bruneliere

Looking to the large diversity of technologies available nowadays to build software systems can be quite disturbing, and the number and complexity of these systems is still considerably increasing! Adding the fact that the amount of more "old-fashioned" existing systems still relevant and used today by the industry is also huge, the problem of dealing with all these legacy systems can be naturally considered as a main challenge in the Software industry. Having a better understanding of the IT legacy in order to document, maintain, improve or migrate them is not an easy task. The MoDisco project [1] intends to facilitate the elaboration of such reverse engineering solutions by offering a reusable and extensible model-based framework. Relying on different Modeling technologies provided in Eclipse, all based on the Eclipse Modeling Framework (EMF) [2], it provides concrete components to tackle this important issue.

A Short History

The MoDisco initiative actually started back in 2006 when the AtlanMod Team (INRIA & Ecole des Mines de Nantes) [3] and Mia-Software (Sodifrance group) [4] were working together in the context of the IST-FP6 MODELPLEX European project [5]. As part of this collaborative project, and one of its tasks dedicated to (model driven) reverse engineering, AtlanMod created the MoDisco project in Eclipse-GMT (Generative Modeling Technologies), which acted at the time as an incubator for modeling prototypes. Several experimental components were then developed by AtlanMod and contributed to this newly created project. Mia-Software rapidly joined the project and started committing actively to it. In 2008, Mia-Software was officially part of the project and increased their involvement accordingly. Since this period, the MoDisco community grew progressively and the project has been regularly presented in the major Eclipse events (EclipseCon and Eclipse Summit Europe, now EclipseCon Europe), among others. In the meantime the MoDisco base infrastructure and set of components were settled and, consequently, the project has been promoted naturally to Eclipse-MDT (Model Development Tools).

The Eclipse Project

MoDisco is thus an official Eclipse-MDT project dedicated to Model Driven Reverse Engineering (MDRE) from IT legacy systems. It is officially integrated within the yearly Eclipse Simultaneous Releases and part of the "Eclipse Modeling Tools" bundles (last year's Helios, this year's Indigo, next year's Juno…).

It provides a set of extensible and reusable components, which form a generic framework allowing the elaboration of concrete solutions to support the specificities of different reverse engineering scenarios.

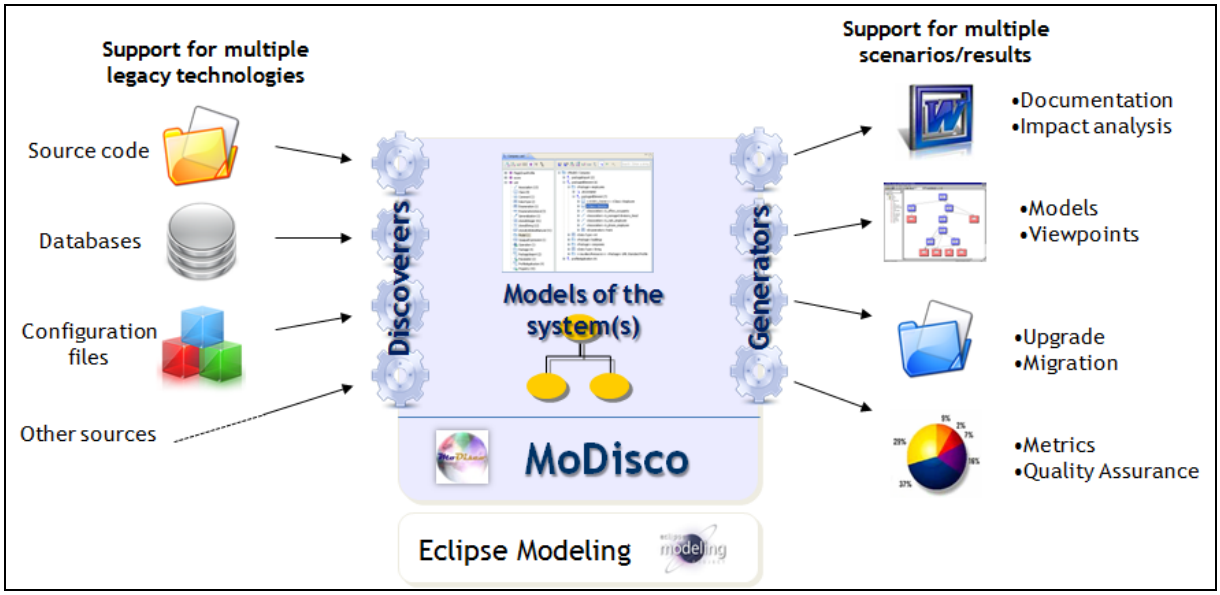


Figure 1 MoDisco Overview

As shown from Figure 1, it can be effectively used as part of modernization processes (technology migration, software refactoring, architecture restructuring, etc), but also for retro-documentation, quality assurance or basically any kind of use case requiring a better understanding of existing systems.

MoDisco is considered by the OMG Architecture Driven Modernization (ADM) Task Force [6] as the reference provider for real implementations of several of its standards: Knowledge Discovery Metamodel (KDM) [7], Software Measurement Metamodel (SMM) [8] and Abstract Syntax Tree Metamodel (ASTM) [9]. In addition to this, it already provides an advanced support for the Java, JEE and XML technologies, including complete metamodels, corresponding discoverers and generators, customizations, query libraries, etc. Now let's see in a bit more details what's in there…

Model-based Features

MoDisco offers a set of model-based components allowing elaborating on reverse engineering solutions. The concerned legacy systems (to be "reverse engineered") can be of varied and various natures. In addition to its technology-independent components, MoDisco natively provides an extensive support for Java legacy systems. From now on, let's consider Java as our legacy technology example to show the different features available in the framework.

The support for a given technology starts by providing the corresponding metamodel(s)[1]. Depending on the selected technology, the detail level of such metamodels may vary. In the case of Java, the metamodel implemented in MoDisco and directly usable covers the full Java language and constructs: it specifies everything a Java program can contain from the global structure (i.e. packages, classes, dependencies) to the internal one (i.e. detailed instructions and statements inside the methods). Thus, it is possible to completely represent an existing program as a (Java) model!

[1] A metamodel can be generally defined as a model specifying the set of concepts, relations between them and constraints required for modeling a given domain or range of problems.

In order to obtain such a model, so-called “discoverers” have to be used. All the available discoverers can be plugged to the framework, and then used accordingly, by registering them via the Discoverer Manager. Their goal is to take as input legacy artifacts and generate as output one (or several) models. Concerning the specific Java Discoverer available from MoDisco, it automatically produces a Java model out of selected Eclipse plug-in(s). As stated before, the content of the obtained Java models exactly reflects the content of the corresponding source code. There is no loss of information here, and this is fundamental as it then allows performing the following reverse engineering actions directly on the models (instead of on the source code).

In a large majority of the possible scenarios (cf. Figure 1) and related processes, it is necessary to be able to navigate efficiently such models of legacy systems. However, one of the main characteristics of these models is their important size and complexity. To directly address this, MoDisco provides as part of its framework a customizable generic Model Browser. It can be used to more easily browse all EMF models (independently from their metamodels) and provides advanced features such as full exploring, infinite navigation, filtering, sorting, searching, etc. As shown on Figure 2, a dedicated Java customization is directly provided and comes with the standard Eclipse icons and formatting.

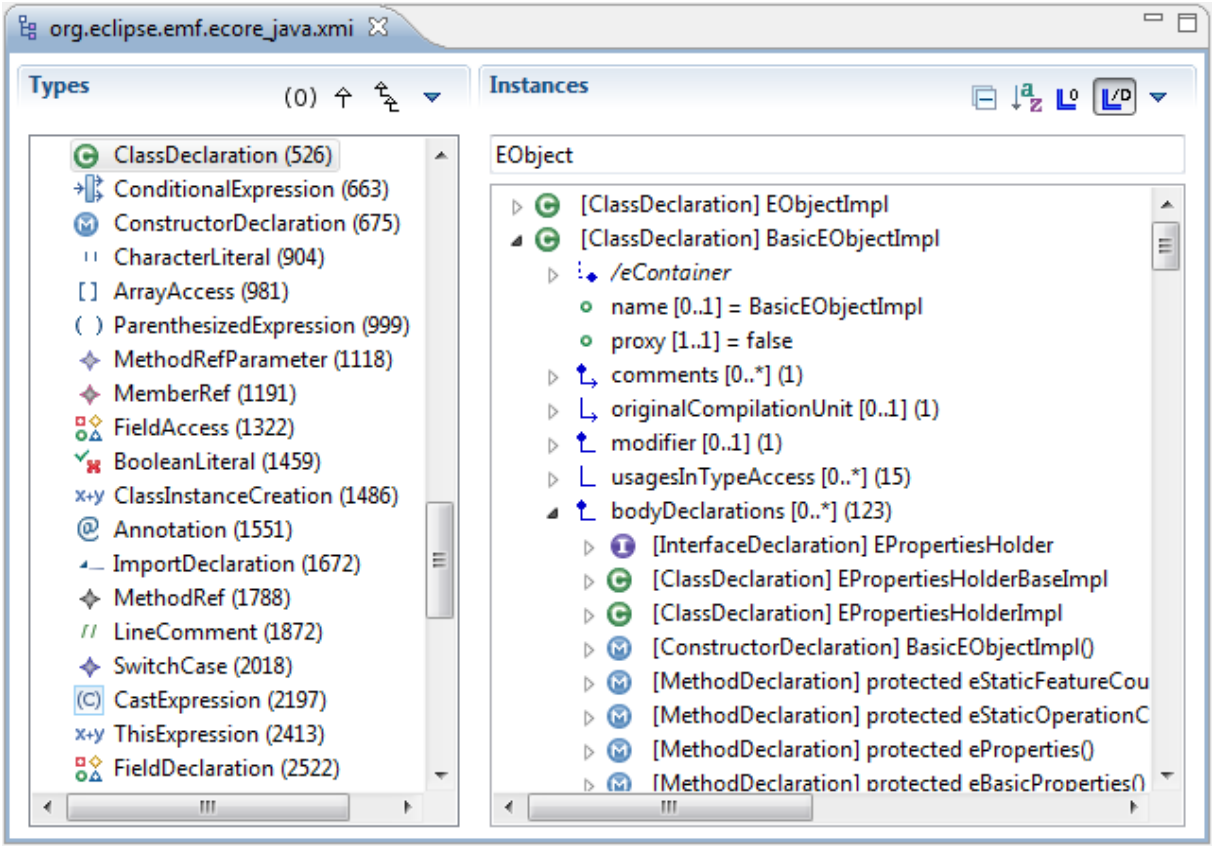


Figure 2 Discovered Java model opened with MoDisco Model Browser

In addition to the Model Browser, MoDisco has an integrated mechanism to define and execute queries over the handled models. Such queries, implemented in Java, OCL [10], ATL [11] or any other language pluggable to the framework via the provided extension points (cf. the Query Manager component), are particularly relevant when some specific information needs to be extracted out of huge models. This is done most of the time in order to have a better understanding of their actual content. The results of the running of these queries can also be reused as part of the so-called “Facet” mechanism, which allows dynamically extending existing models with

data computed at runtime (via the queries). Originally developed as part of MoDisco, this mechanism has now its own life within the context of the dedicated Eclipse-EMFT EMF Facet project [12]. But of course, facets are still integrated and used intensively in MoDisco for reverse engineering purposes.

Similarly to “discoverers”, MoDisco also comes with “generators” corresponding to the supported technology metamodels. Thus, Java has its specific generator which allows producing the Java source code from the handled Java models. Moreover, in order to be able to obtain other kinds of artifacts out of these models (as shown on Figure 3), transformations can also be defined and plugged within the framework via an integrated Discovery Workflow. Some transformations from Java to various Object Management Group (OMG) standards are already provided such as notably KDM [7] and UML2 [13].

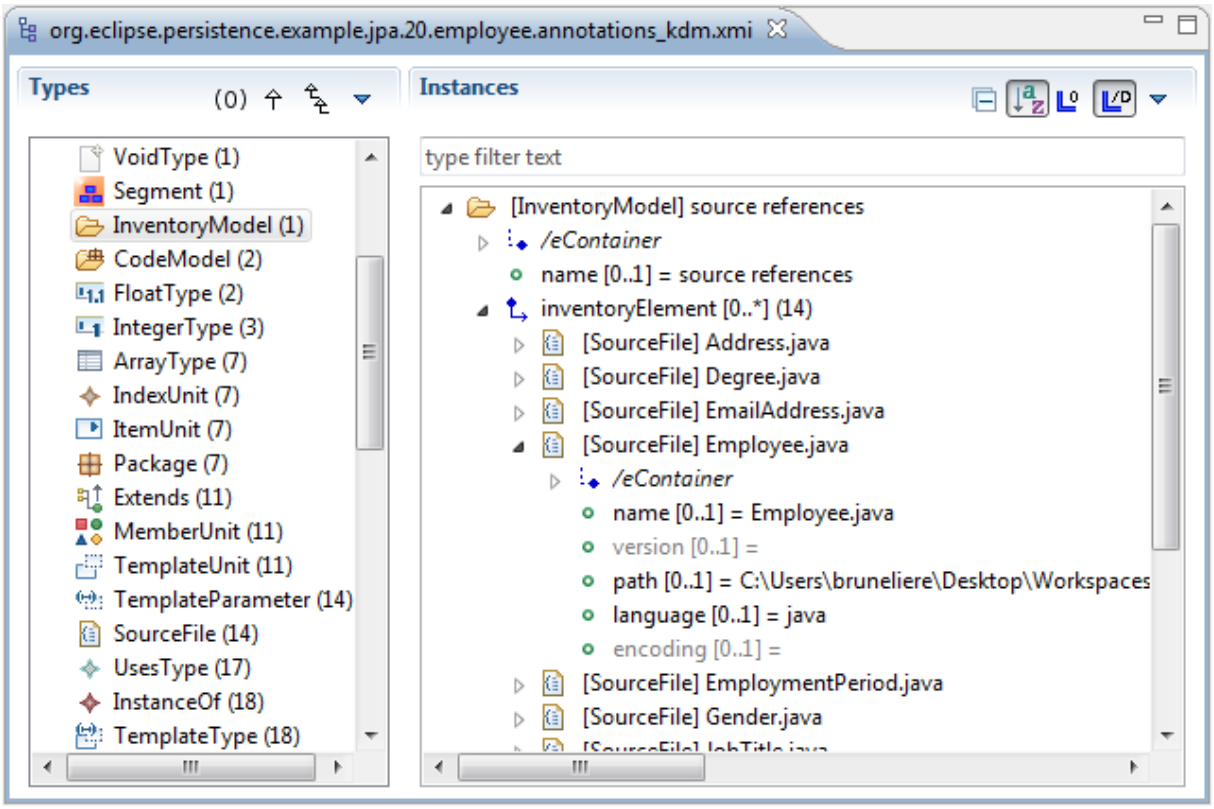


Figure 3 KDM model automatically generated by transformation from an initial Java one

All the presented features and Java-dedicated support are very useful when dealing with reverse engineering from Java legacy systems. Note that, as complementary components but still very interesting, a generic XML metamodel (based on the related W3C specification) and corresponding discoverer are also given for free. Let’s now take a look to the complete novelties brought by the Indigo release of MoDisco…

Reusability and Extensibility

In the two previous sections of this article, the different available MoDisco features have been largely presented, with a strong focus on the extensive Java support natively offered by the framework. Let’s now try to have a more global vision of the different MoDisco components and see the benefits brought by the underlying architecture.

As shown on Figure 4, the MoDisco framework is actually composed of three distinct but complementary layers.

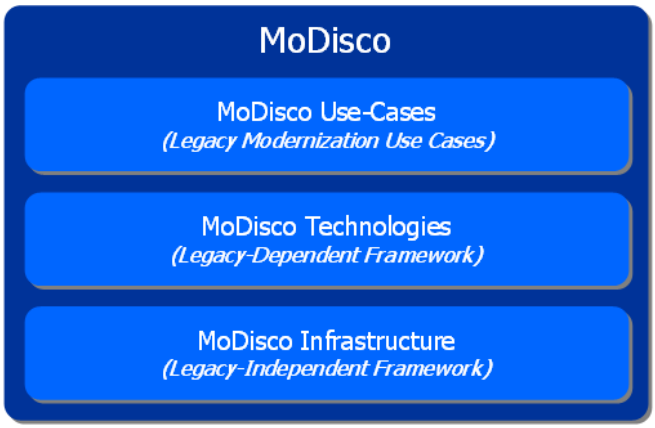


Figure 4 MoDisco framework architecture

The “Infrastructure” layer is the base of the framework. It provides the set of generic components (in the sense of legacy technology-independence) that can be reused in various and varied reverse engineering scenarios. It includes fundamental components such as the powerful Model Browser, the Discovery Manager & Workflow or the Query Manager. It also contains the reference implementations of generic OMG ADM standard metamodels, as presented before in this paper, as well as the corresponding tooling (related transformations, customizations, etc).

The “Technologies” layer is made to be used on top of components from the “Infrastructure”. It implements the legacy technology-specific support via the corresponding reference metamodels, discoverers, generators, transformations, etc. As stated before, the Indigo MoDisco version notably offers an advanced support for the Java and JEE technologies. Additionally, a base support for XML is also provided. As relying on generic components from the “Infrastructure”, the set of components available from this “Technologies” layer can be naturally extended with the support for other technologies, and the related components then plugged in the framework accordingly for future uses

Finally, the “Use Cases”.layer is just here to present a couple of sample scenarios showing how the different components, from the two previously described layers, can be integrated together via the framework in order to achieve a particular reverse engineering goal. Thus, two simple scenarios of this kind are currently provided to quickly demonstrate what can be done with MoDisco when starting from available Java artifacts in the Eclipse environment.

These three layers, and the natural complementarity between their contained generic and technology-specific components, give to the MoDisco framework all the extensibility and reusability capabilities required to be able to elaborate on concrete reverse engineering solutions.

What’s new in the Indigo version?

Different significant improvements have been performed on most of the existing MoDisco components (since the previous Helios version). Obviously, many of them relate to the provided support for the Java technology as described before. The upgrades mainly concern regular bug fixing, feature enhancement requests, but also performance increases according to these items.

However, as part of MoDisco v0.9.0, brand new metamodels and discoverers can be now found out. On one hand, what's totally new in this year's Indigo version of the project is the extension of the Java support with a set of JEE-dedicated components. They include notably:

- A JSP metamodel and corresponding discoverer to automatically obtain models from existing JEE Web applications written in JSP (cf. example from Figure 5);
- A specific framework to facilitate the development and use of discoverers for JEE descriptor files (e.g. for the different JEE framework configuration files);
- Related to this, the already implemented support for two of these descriptor file types: "web.xml" and "ejb-jar.xml";
- And a set of useful queries and facets for more easily extracting or presenting JEE-specific information from previously discovered Java models.

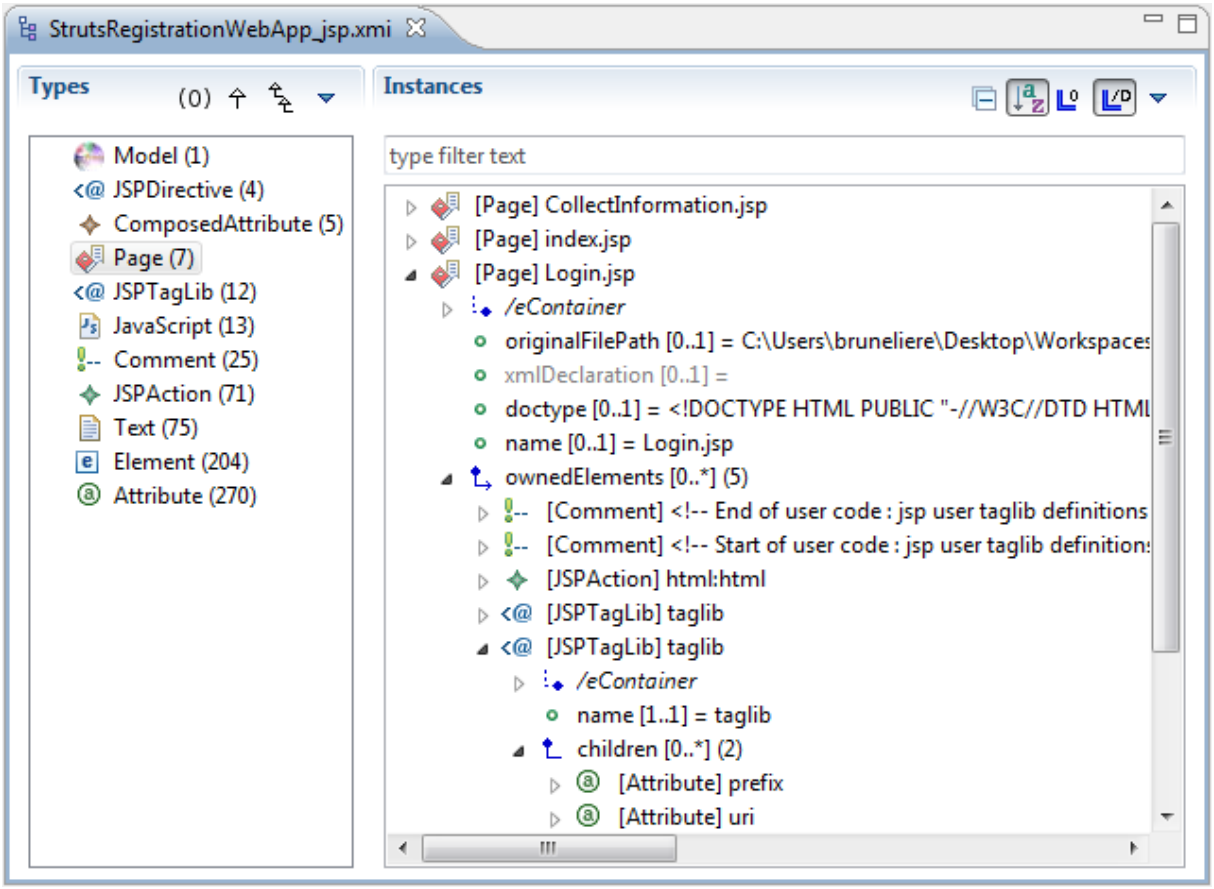


Figure 5 Discovered JSP model opened with MoDisco Model Browser

On the other hand, continuing collaborating with the OMG ADM Task Force, MoDisco now offers a reference implementation (in EMF Ecore) of another of their standards called Abstract Syntax Tree Metamodel (ASTM) [9], in addition to the already provided KDM [7] and SMM [8] ones.

Conclusion

In its latest version for Indigo, the MoDisco project and its underlying framework are now providing a significant support for the Java technology and related JEE ones, such as JSP for instance. They also provide

more reference implementations of relevant OMG ADM standards. Based on this set of components and offered features, either upgraded since the previous Helios version or completely new, complex (model driven) reverse engineering solutions can be designed, then built and finally executed effectively over different legacy systems. Of course, all these available resources are heavily supported by an active development team and continuously growing user community.

Moreover, the release of MoDisco v0.9.0 also opens a new exciting era in the history of the project. Next version is planned to be v1.0.0, which will imply the project to "graduate" and get rid of the "incubation" status to officially become a "mature" project (according to the Eclipse Foundation terminology). The required effort will be directly supported by the MoDisco team via its extension with additional dedicated resources. This is fully part of the ongoing project's industrialization process, driven in collaboration with AtlanMod and Mia-Software, which will be obviously continued and even strengthened in the coming months.

Reverse engineering and the understanding of legacy systems remain complex fields with some still open very interesting challenges. One of those which are particularly relevant in the context of MoDisco is scalability: how to deal efficiently with very big legacy systems and the huge amount of data (and metadata) this implies in corresponding reverse engineering processes? Let's focus a bit more on this during the coming year… So stay tuned, and of course see you next year for the MoDisco Juno version!

Hugo Bruneliere is an R&D engineer working in the field of Model-Driven Engineering (MDE) for the AtlanMod Team (INRIA, EMN & LINA), located in Nantes (west of France). He has worked during 3 years and a half as the INRIA responsible for the MODELPLEX IST European project 34081 in which AtlanMod was involved. His work is now focusing on the methodologies and concrete uses of Model-Driven Engineering (MDE) for model-driven reverse engineering (or MDRE), global model management (or GMM) and tool interoperability (using model transformation tools such as Eclipse-M2M ATL). He is the leader of the Eclipse-MDT MoDisco project and also committer in the Eclipse-EMFT EMF Facet project.

References

[1] Eclipse-MDT MoDisco project: http://www.eclipse.org/MoDisco/
[2] Eclipse Modeling Framework (EMF): http://www.eclipse.org/modeling/emf/
[3] AtlanMod Team (INRIA, Ecole des Mines de Nantes & LINA): www.emn.fr/x-info/atlanmod
[4] Mia-Software (Sodifrance group): http://www.mia-software.com/
[5] IST FP6 MODELPLEX European Project: www.modelplex.org
[6] OMG Architecture Drive Modernization (ADM) Task Force: http://adm.omg.org/
[7] OMG Knowledge Discovery Metamodel (KDM): http://www.omg.org/spec/KDM/
[8] OMG Software Metrics Metamodel (SMM): http://www.omg.org/spec/SMM/
[9] OMG Abstract Syntax Tree Metamodel (ASTM): http://www.omg.org/spec/ASTM/
[10] Eclipse-MDT Object Constraint Language (OCL) project: http://www.eclipse.org/modeling/mdt/ocl
[11] Eclipse-M2M ATL Transformation Language (ATL) project: http://www.eclipse.org/atl/
[12] Eclipse-EMFT EMF Facet project: http://www.eclipse.org/modeling/emft/facet/
[13] OMG Unified Modeling Language (UML): http://www.omg.org/spec/UML/