

The Model Discovery (MoDisco) Component: A Proposal for a New Eclipse/GMT Component



By Jean Bézivin, Hugo Brunelière, Mikaël Barbero – INRIA (ATLAS Group)

Version 1.0

Introduction

MoDisco (for Model Discovery) is an Eclipse GMT component for model-driven reverse engineering. The objective is to allow practical extractions of models from legacy systems. Because of the widely different nature and technological heterogeneity of legacy systems, there are several different ways to extract models from such systems. MoDisco proposes a generic and extensible metamodel-driven approach to model discovery. A basic framework and a set of guidelines are provided to the Eclipse contributors to bring their own solutions to discover models in various kinds of legacy.

Due to the highly diversified nature of the considered legacy, MoDisco is a collaborative component involving many organizations. Each of them will bring its own expertise in a given area. A common infrastructure, inspired by the OMG KDM recommendation, will allow integrating all these contributions.

As a GMT component, MoDisco will make good use of other GMT components or solutions available in the Eclipse Modeling Project (EMF, M2M, GMF, TMF, etc), and more generally of any plugin available in the Eclipse environment.

Background

Systems are becoming more and more complex. Developing and managing such complex systems already is a main issue. The next important effort is about reverse engineering complex legacy systems in order to be able to migrate them, make them interoperable, or simply understand them.

The proposed MoDisco component is mainly about providing an extensible and generic framework under the Eclipse GMT project, part of the top-level Eclipse Modeling Project. The Eclipse GMT project acts as a research incubator for MDE prototypes. Thus, the MoDisco component aims at providing a base framework for model driven reverse engineering tasks. We will discuss the framework's composition further in this document.

One of the key to success of this extensible framework will be its adoption by leading industrials and the development of a wide variety of extensions and a wide user community.

Discovery Principles

Principles of model discovery are based on a metamodel-driven approach (see Figure 1). It means that every step is guided by a metamodel. Thus, the very first step of a model discovery process is always to define the metamodel corresponding to the models you want to discover. This step is common to all kinds of systems.

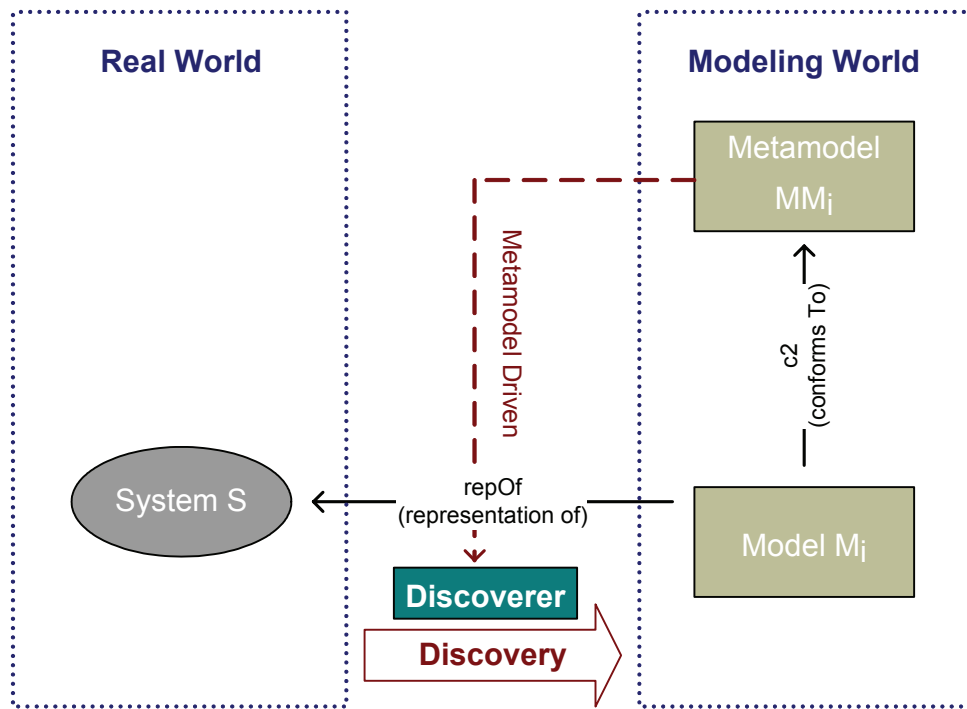


Figure 1 Metamodel driven model discovery

Then, the second step is about creating one or many discovering tools that will be called “discoverers” in this document. These tools extract necessary information from the system in order to build a model conforming to the previously defined metamodel. The way to create these discoverers is often manual but can also be semi-automatic. The output of a discoverer is a model, in XMI format for instance.

Motivating Examples

As a first motivating example, we will study a Unix system from two different points of view. The first one will consider the discovery of the file system’s structure, i.e. the snapshot of the file system at time t . The second one will take in account users’ login and users’ logout operations during a given duration.

File system discovery

In this first example, we would like to discover the file organization of a UNIX system (see Figure 2). We define a simple metamodel to this purpose. An abstract entity file (as

everything is a file linked to a user in a UNIX system) and two specialized file types: folder and simple file. A folder is a special file type that may contain any type of files.

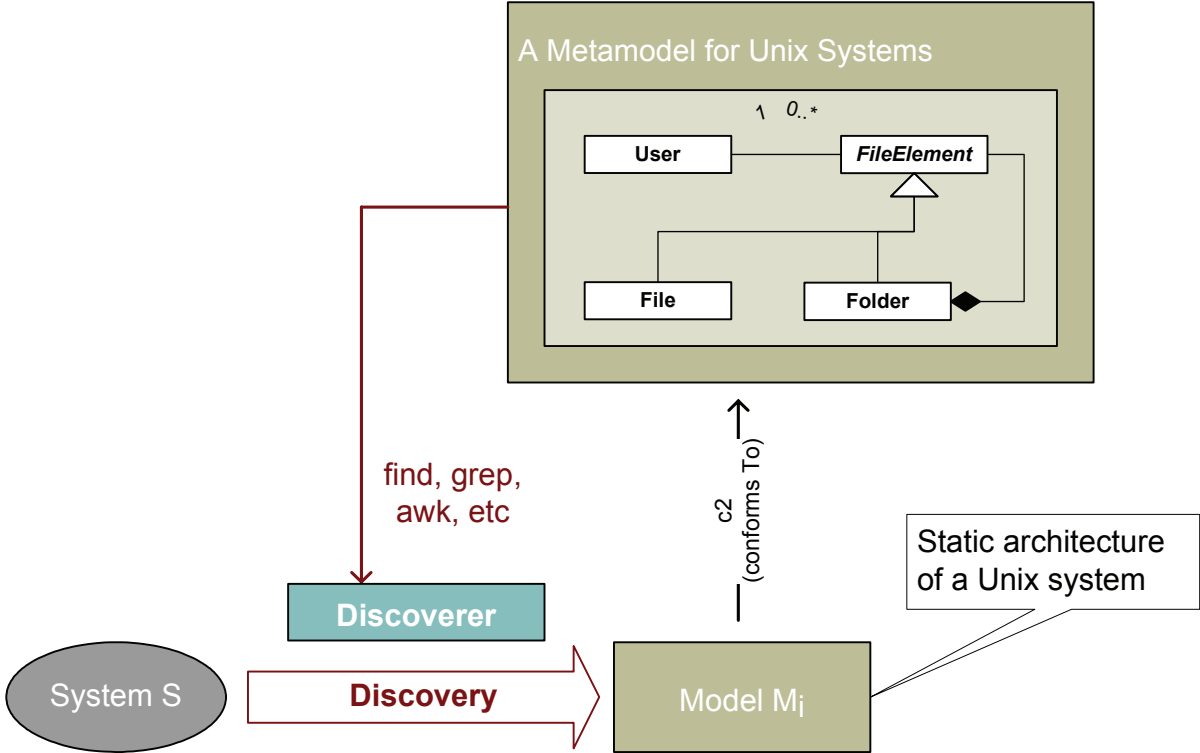


Figure 2 Unix file system discovery

Then, we define a set of shell scripts (manually or semi-automatically) using find, grep, awk, etc. commands. The semi-automatic definition of this discoverer should be possible by adding decorations on metamodel’s entities. These decorations can be assimilated to metadata on metamodel’s entities but the definition of such a method is out of the scope of this document.

The discoverer is then applied and generates a model of the file system conforming to the metamodel. This model can be in a concrete syntax of the metamodel or in an exchange format as XML. Finally, the discoverer may access a JMI-like programming interface for instantiating models element. In that case, the model would automatically be created in memory.

Login and logout trace

In this second example, we would like to discover the sequence of users’ login and logout events in a given Unix system (see Figure 3). Thus, we define similarly the metamodel. Each of the login and logout events is linked to a specific user.

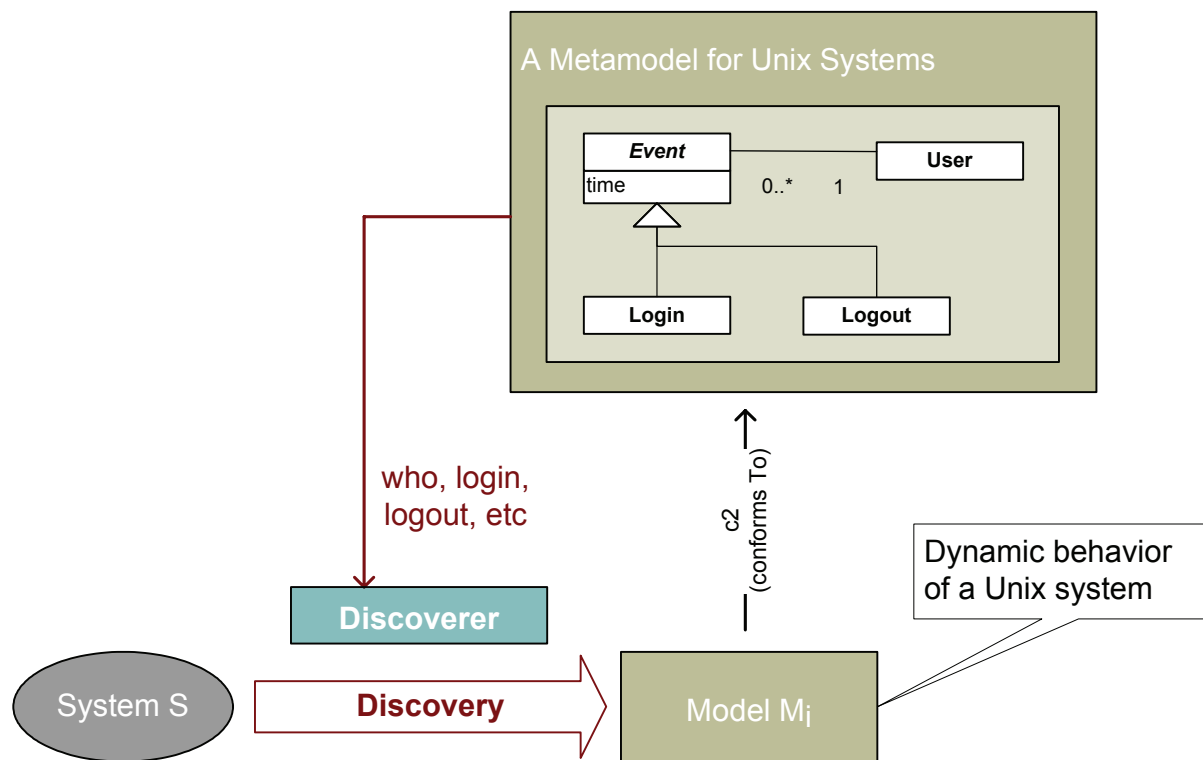


Figure 3 Unix users' login and logout trace discovery

Then we create the discoverer from the metamodel definition. For instance, this discoverer can observe system's events to dynamically create the output model. Another solution would be to analyse a log history to do it.

Here, we face a very interesting issue. We study the dynamic dimension of a system. The model represents an execution trace of the system. It is a snapshot of the system evolution between two moments whereas the preceding example was a snapshot of the system at time t . We always consider that the observation has an end but it could never finish: in this special case, we face an infinite model.

Conclusions about these two examples

Considering the two previous examples, we can conclude that the same general process is applied in both cases. But we also note that different kinds of systems involve different kinds of discoverers and probably different MoDisco framework facilities.

Nevertheless, we reckon that the very first investigation to do is about classifying systems. This classification may not be an absolute one, but a decision tree to help developing discoverer. Moreover, we have to keep in mind that a same system can be classified in two different categories, depending on the point of view we would like to observe it (as the previous Unix system). The research interest behind this is about linking the two discovered models from the two points of view of the same system.

We propose in Figure 4 a very first initial sketch of such a classification that will be developed as more examples are considered.

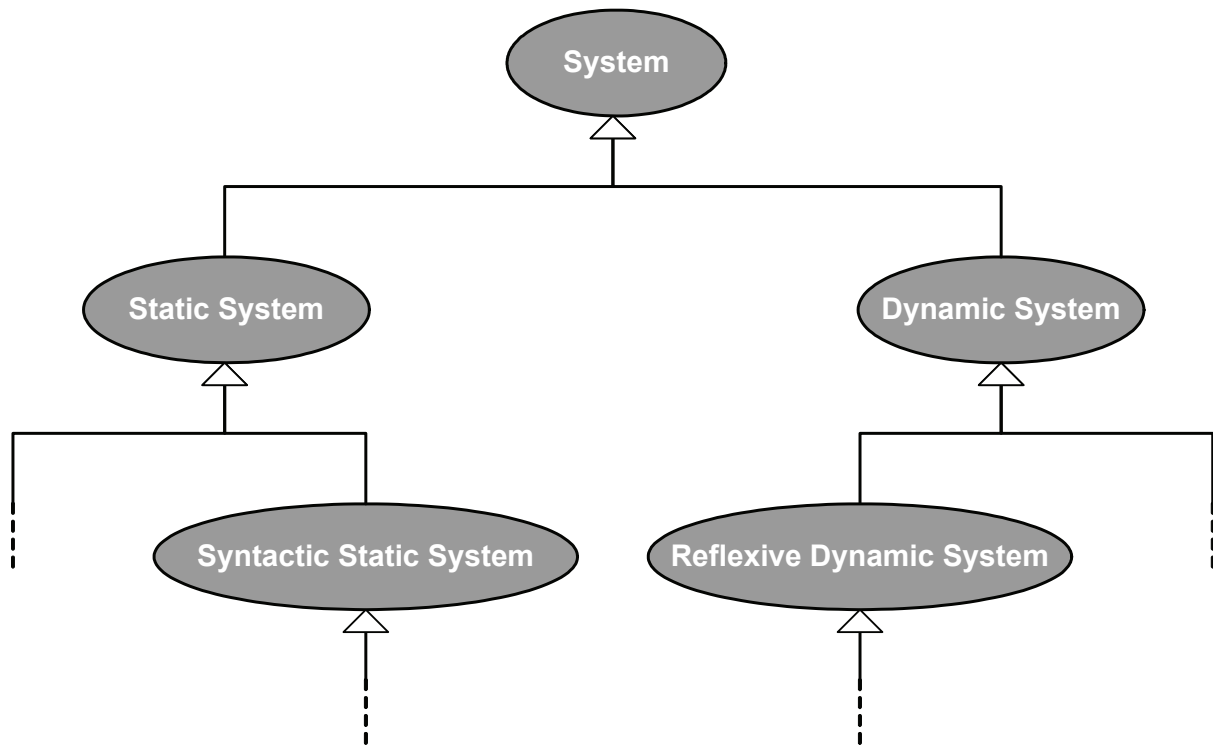


Figure 4 A very first systems classification example

Finally, we would like to discuss an interesting use case: a complete reflective system equipped with introspection and intercession capabilities. These kinds of systems allow us to define discoverer in the same language as the system is. Then, as they are fully reflective, we are able to discover everything in those systems. So reflective systems and associated discoverer may be the best ones in term of ease and completeness of discovery.

Component Description

The MoDisco framework is a generic framework that provides a basis for extension. It offers a minimum tool set to allow model discovery. The first component is a base metamodel. It is based on the Knowledge Discovery Metamodel (KDM) from the OMG. Actually, it is a minimal subset of KDM allowing end users to define (by extension) some KDM compliant metamodels. The framework offers facilities to manipulate models which metamodels are extensions of the base metamodel.

As we already said, the framework is designed to be extended. A MoDisco extension is a couple of base metamodel's extension and plugin. This couple is called a "blade". Those blades have a specific domain of discovery, for instances Unix files system discovery, Cobol system cartography, 4GL source code cartography, etc. The global extension mechanism will be based on model extension concepts applied to metamodel (to be defined) and Eclipse plugins mechanism. The couple of extension points provided by the MoDisco framework will build the blades' sleeve.

A graphical overview of the MoDisco component is provided in Figure 5.

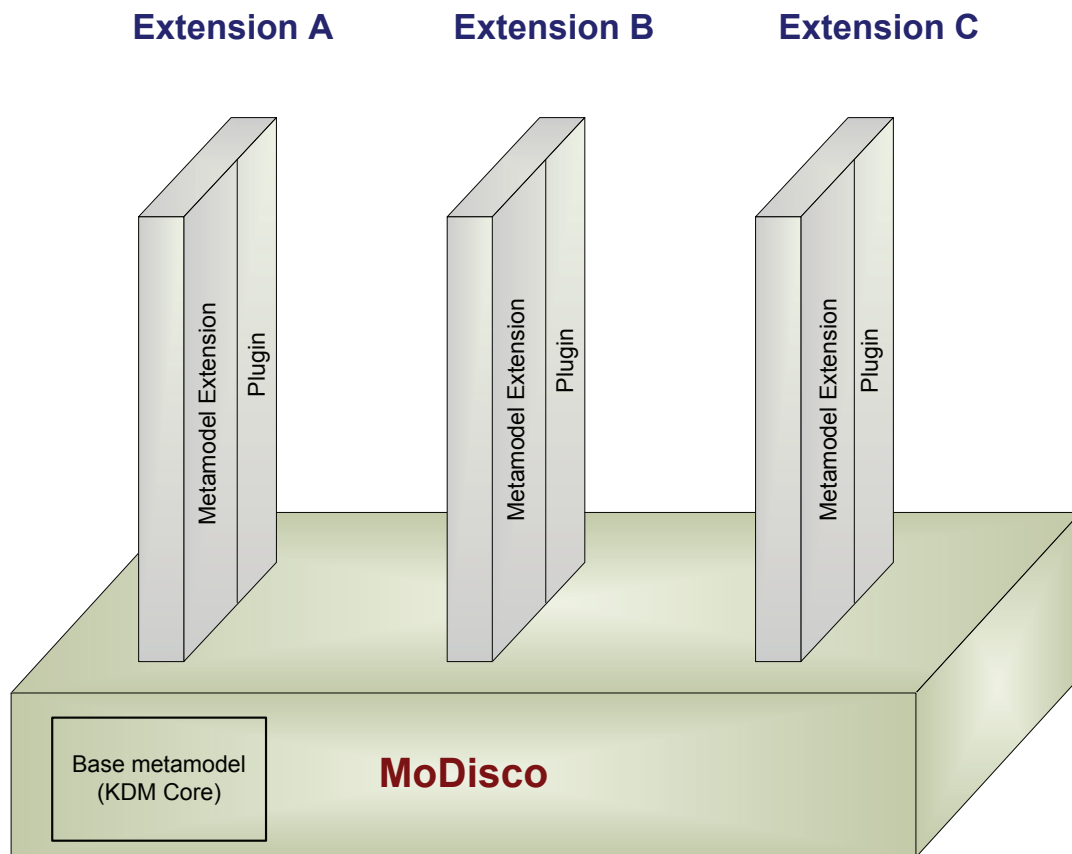


Figure 5 Overview of the MoDisco component

In addition, MoDisco will not only be a programming framework. It will define a methodology to help users to build extensions. For instance, the fact of having a system classification and a methodology for each system type would be crucial for its adoption.

Benefits of This Approach

What are the benefits of the MoDisco approach compared to already existing reverse engineering tools?

First, MoDisco proposes a unified approach to model-driven reverse engineering and a metamodel driven methodology. This way, we are able to work in the modeling world, coming from a heterogeneous world to a homogeneous one. The target model engineering space already proved its adaptability and scalability by several experiments to match requirements for data integration, tools interoperability and platform migration.

Moreover, the well structured modeling world allows easy manipulation of many different concepts in a unified way. For instance, every model can be transformed, weaved, extracted with the same tool set. As those operations are defined upon models' metamodels, they are reusable for different use cases.

Organization

We propose that the MoDisco component be undertaken as part of Eclipse Modeling Project (EMP)/Generative Modeling Technologies (GMT).

The initial list of committers, contributors and interested parties will be completed and provided later.

MoDisco is supported by the ModelPlex European Integrated Project (FP6-IP #034081).

A first prototype implementation of the basic framework is currently under development and will be contributed as soon as possible. Several white papers will also be provided illustrating various possible applications of the MoDisco tools.

Revision History

- First draft, October 23, 2006
- First revised version, October 27, 2006
- Last revised version, October 31, 2006