A FRAMEWORK FOR MODEL-DRIVEN PATTERN MATCHING

Ignacio García-Rodríguez de Guzmán, Macario Polo and Mario Piattini
ALARCOS Research Group
Information Systems and Technologies Department
UCLM-Soluziona Research and Development Institute
University of Castilla-La Mancha
Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain

Keywords: MDA, pattern-matching, QVT, transformation.

Abstract: Today, software technology is evolving to model engineering. Standards such as MOF and MDA and languages such as QVT and ATL are emerging to support this evolution from object paradigm to model engineering. At times, these standards and languages give rules and advices at a high level of abstraction, and concrete solutions and implementations are difficult to perform. As a consequence of this technological immaturity and the lack of documentation, many capabilities in this new field are not exploited. To this end, the authors in this paper propose a first step of providing a framework for performing *Model-Driven Pattern Matching* operations. Pattern matching based on models is an evolution of a traditional concept adapted to the model realm. In this respect, this kind of pattern matching seems to be promising not only for finding occurrences of given models in others, but also for giving meaning or sense to these patterns in order to undertake actions over the resulting matchings.

1 INTRODUCTION

Today, software engineering is going through a change of paradigm from object orientation to model driven development (Bézivin, 2006). Perhaps one of the reasons for this evolution is the growth of current platform complexities, which has evolved faster than the ability of general-purpose language to face it (Schmidt, 2006).

MDE brings many other standards such as MDA (OMG, 2003a), QVT (OMG, 2005a), UML2 (OMG, 2005b) among others.

One of the most ambitious bets is QVT, a model transformation language. QVT makes it possible to perform different kinds of operations over models such as query, transformation and views generation.

This paper focuses on QVT capabilities for performing pattern matching. QVT uses pattern matching to carry out most operations. Since both the pattern and the data are models, this pattern matching technique can be seen as a *Model-Driven Pattern Matching* (MDPEM from now on) process.

MDPEM has been conceived as an important element inside an MDA process intended to infer and to extract services from relational databases (García-Rodríguez de Guzmán et al., 2006a). In this process, MDPEM is also used to perform additional tasks over matchings.

This paper is organized as follows: Section 2 depicts the recent history of QVT; Section 3 describes the proposed framework using a working example; after the introduction of the approach, Section 4 outlines a possible use for matchings; Section 5 provides some conclusions and introduces future lines of work in this field.

2 STATE OF THE ART

Perhaps the soundest MDE-related technology is MDA (OMG, 2003a). Model transformation is an important part of MDA, and OMG proposes QVT to perform this operation.

OMG published the QVT RFP in 2002. In March 2003, the QVT-Partners published the "Initial submission for MOF 2.0 Query/Views/Transformations RFP" (OMG, 2003b).

In November 2003, QVT-Partners published the “Revised submission for MOF 2.0 Query / Views / Transformations RFP” (OMG, 2003c) (showing a more complete specification along with the declarative and imperative QVT´s languages). In November 2005, OMG published the “MOF QVT Final Adopted Specification” (OMG, 2005a).

On the other hand, up to now there is no available any QVT engine implementing the declarative QVT language. In order to solve this problem, some projects in the academic world (Queralt et al., 2006) are now underway.

3 MDPEM FRAMEWORK

3.1 An Overview to the Framework

This framework is used to provide information about: (1) where the patterns can be located in the MOF architecture; (2) how these patterns are represented; (3) the target model against which patterns are matched and (4) what the resulting matchings are and where they are located.

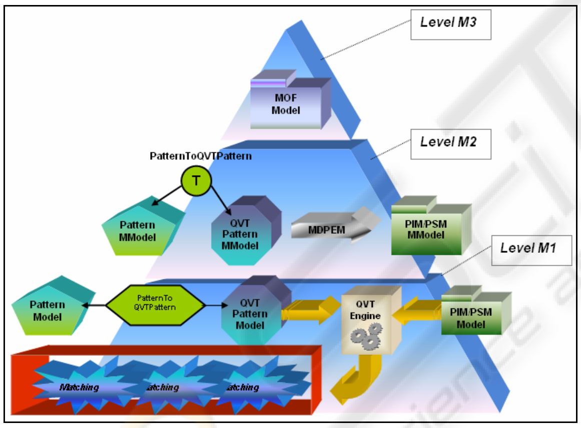


Figure 1: Generic framework for MDPEM.

Both patterns and models (against which patterns are matched) have their own metamodel. Thus *Level M2* (*MetaModel Level*) represents all these metamodels, as well as transformations among them to perform the MDPEM process.

Patterns, models and matchings make up a particular metamodel (from level 2). So *Level M1* level (*Model Level*) represents all the models involved in the MDPEM process.

3.2 Elements Involved in the Process

In order to make the framework clear, a description of all the elements is given:

- *Pattern MModel*: To generate valid patterns to do the MDPEM against a target model, a pattern metamodel is given together with the target metamodel. *Pattern Model* is the model actually describes the pattern to be found in the target model
- *QVT Pattern MModel*: QVT provides a metamodel (OMG, 2005a) to express any searching pattern (*template* in QVT terminology). The *QVT Pattern Model* is obtained applying the *PatternToQVTPattern* transformation over the *Pattern Model*.
- *PIM/PSM MModel*: This metamodel is used to represent the target model. *PIM/PSM Model* is actually the target model.
- *PatternToQVTPattern*: Because a *QVT Template* is required to perform the MDPEM process, a transformation between the pattern metamodel and the QVT template should be given for each pair <*pattern metamodel*, *QVT template*>. The transformation is defined in a metamodel level, but applied to models.
- *MDPEM*: represents that the QVT template metamodel is the basis to perform the MDPEM over the instance of the target model metamodel. The matching process is executed by a *QVT Engine*.
- *Matching*: Represents the result (if any) from the MDPEM execution using the specified *QVT template* over the given *target model*. These matchings are also models from the *target model*.

3.3 MDPEM Process

The MDPEM process is divided in the following steps:

1. Pattern model and target model are given.
2. QVT Template instance is obtained from Pattern Model.
3. MDPEM is carried out.
4. Matchings (sub-models) are returned to the invoker

According to (QVTP, 2003), “*The essential idea behind pattern matching is to allow the succinct* *expression of complex constraints on an input data type; data which matches the pattern is then picked out and returned to the invoker*". In our context, the *pattern model specifies the complex constraints*, and the *target model* represents *the data*.

In addition to these elements (and according to the framework shown in Figure 1), another element is required to perform the MDPEM process: the transformation between the *pattern model* and the *QVT template*.

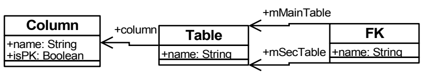


Figure 2: Very simple metamodel.

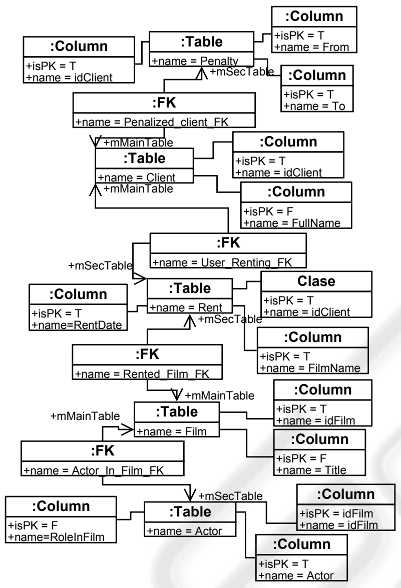


Figure 3: Working example.

As noted above, any model representing a searching pattern must be compatible in the context of the target model. Therefore, to specify a pattern, it may be useful to use the target model metamodel. Consequently, any pattern will be compatible with the target model and thus, MDPEM applicable. In this section, the process depicted in Section 3.3 will be explained using the working example of Figure 3. Both target model and pattern will conform to the metamodel in Figure 2.

3.3.1 Pattern Model and Target Model

As a first step in the MDPEM process, the definition of both the target model and the pattern is mandatory. Figure 3 contains the target model. This example represents a (very simple) database for a video store. This database keeps information about clients, films, rents, penalties and actors. This model conforms to the metamodel in Figure 2.

Once we have the target model, the pattern must be specified. Because the example in Figure 3 is a simple database, the pattern must specify the constraints in terms of tables, foreign keys, columns and so on. Figure 4 (a) represents the double foreign key pattern (DFK) (García-Rodríguez de Guzmán et al., 2006b). The DFK pattern relates three tables (*a*, *b* and M) by means of two foreign keys (fk_1 and fk_2). Figure 4 (b) represents the DFK pattern according to the metamodel in Figure 2.

3.3.2 QVT Template Generation for MDPEM Application

According to Figure 1, the pattern expressed in terms of the PIM/PSM metamodel is not the same as that used to perform the MDPEM. To this end, a *QVT Template* is obtained from the proposed pattern (Figure 4 (b)).

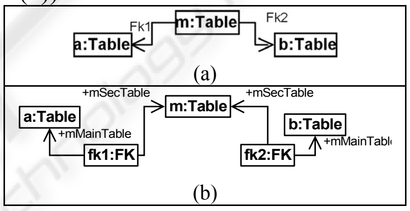
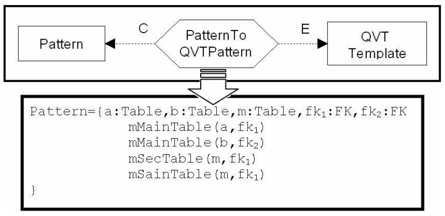


Figure 4: (a) DFK pattern and (b) Figure 2 metamodel-like representation.



```
Pattern={a:Table,b:Table,m:Table,fk1:FK,fk2:FK
            mMainTable(a,fk1)
            mMainTable(b,fk2)
            mSecTable(m,fk1)
            mSainTable(m,fk1)
}
```

Figure 5: QVT template creation.

There are two ways to obtain a *QVT Template*: (1) manually or (2) automatically. The first may be complex depending largely on the complexity of the pattern metamodel and the size of the pattern. The second can be carried out implementing a suitable transformation to obtain *QVT Template* from the source pattern (see Figure 5).

Due to the lack of space, the *PatternToQVTPattern* transformation for a working example is not shown, but rather the *QVT Template* textual representation. Once the *PatternToQVTPattern* transformation is written, any pattern conforming to the pattern metamodel can be transformed into the *QVT Template*.

This textual *QVT template* representation can be understood in the following way: "return all the matchings composed by three tables (*a*, *b* and *m*) and two foreign keys (fk_1 and fk_2). Those elements must hold the following conditions: *a* is related to fk_1, *b* is related to fk_2, *m* is related to fk_1 and *m* is related to fk_2".

In this pattern, the criteria to process the search are only based on the structure of the elements composing the pattern. In another situation, it may be useful to establish another kind of conditions.

3.3.3 MDPEM Application

Given the *QVT template* (representing the pattern) and the *target model* (Figure 3), the QVT engine looks for all the occurrences (matchings) of the template that exist in the model.

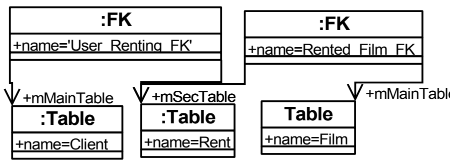


Figure 6: Matching obtained from the working example.

As a result, a set of sub-models from the target model, holding the constraints set by the pattern, are returned to the invoker.

Because all the matchings are "fragments" of the target model, all of them belong to the same level of the target model (Figure 1).

The only obtained matching (Figure 6) consists of a set of classes that hold the *a, b* and *c* tables and the foreign keys fk_1 and fk_2

4 PURPOSE OF MATCHINGS

Matchings may be useful when a particular purpose is bound to patterns. For example, a pattern such as DFK could be accompanied by abstract operations. Each *abstract operation* involves those tables included in the pattern, so, when a matching is found, this *abstract operation* can be applied to a real set of tables. The real result of this matching is a set of operations associated with the pattern.

The DFK pattern can be accompanied by the following operations: *getA_ForAGiven_B* (having B, obtain the associated A) and *getB_ForAGiven_A* (the opposite). The combination "*pattern+actions*" can be a powerful tool to deal with complex systems. Another possible use for MDPEM could also be the design pattern detection in large software systems, such as other authors do (Zhang et al., 2004).

5 CONCLUSIONS AND FUTURE WORK

In this paper, a framework for *Model-Driven Pattern Matching* has been presented. To perform MDPEM, both the pattern and target models must be known. To ensure compatibility among these models we propose using the metamodel of the target model (or at least a subset) to build the pattern. Thus any given pattern can be used to find matchings in a given target model.

Because each pattern must be translated into a *QVT Template*, a suitable transformation must be developed. An excerpt of a transformation to obtain *QVT Templates* from our patterns is presented.

ACKNOWLEDGEMENTS

This work has been partially supported by the ENIGMAS Project (PIB-05-058), the FAMOSO Project (2006: FIT-340000-2006-67); and the MECENAS project (PBI06-0024).

REFERENCES

Bézivin, J. (2006). Introduction to Model Engineering, ATLAS Group (INRIA & LINA), Nantes.

García-Rodríguez de Guzmán, I., M. Polo and M. Piattini (2006a). A Methodology for Database Reengineering to Web Services. European Conference on Model Driven Architecture - Foundations and Applications, Bilbao (Spain), Springer-Verlag Berlin Heidelberg.

García-Rodríguez de Guzmán, I., M. Polo and M. Piattini (2006b). Un primer paso para la obtención de servicios en bases de datos relacionales mediante patrones y MDA. ZOCO: Desarrollo y Mantenimiento Ágil de Aplicaciones Basadas en Servicios Web, Sitges (Barcelona).

OMG (2003a). MDA Guide Version 1.0.1., Object Management Group**:** 62.

OMG (2003b). QVT-Partners initial submission to qvt-rfp, Object Management Group.

OMG (2003c). Revised submission for MOF 2.0 Query/Views/Transformations RFP, Object Management Group.

OMG (2005a). MOF QVT Final Adopted Specification, Object Management Group.

OMG (2005b). Unified Modeling Language: Superstructure. Versión 2.0.
Queralt, P., L. Hoyos, A. Boronat, J. Á. Carsí and I. Ramos (2006). Un Motor de Transformación de Modelos con soporte para el Lenguaje QVT Relations. Desarrollo del Software Dirigida por Modelos y Aplicaciones 2006 (DSDM´06), Sitges, Barcelona (Spain).
QVTP (2003). Revised submission for MOF 2.0 Query / Views /Transformations RFP (Version 1.1), QVT-Partners (http://qvtp.org/).
Schmidt, D. C. (2006). "Model-Driven Engineering." IEEE Computer 39(2): 25-31.
Zhang, Z. and Q. Li (2004). Automated Detection of Design Patterns. Grid and Cooperative Computing, Springer Berlin / Heidelberg**:** pp 694-697.

SciTePress
Science and Technology Publications