Metamodeling or Profiling: a Practical Case in the Web Engineering Domain

Antonio Navarro and Humberto Cortés

Dpto. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid
28040 Madrid, Spain
anavarro@fdi.ucm.es, hjcortes@estumail.ucm.es

Abstract. Model Driven Development (MDD) provides several choices for the definition of modeling languages. The definition of a complete metamodel and the customization of a given metamodel using profiles are common approaches. In our opinion neither of these approaches is better than the other, and the choice should depend on the characteristics of each project. This paper describes our experiences defining a graphical notation for the characterization of web navigational maps based on a MOF metamodel and a UML profile. The advantages and drawbacks of both approaches are examined, as well as the solution selected for our project.

Keywords: MDA, MDD, NMM, navigational map, multitier architecture

1 Introduction

Model-Driven Architecture (MDA) is an approach defined by the *Object Management Group* (OMG) that promotes the development of software systems based on their models [1]. In MDA, different viewpoints of the software system are provided through models. To some extent, these models can be viewed as evolutions of the system from reality towards code. In this way, if reality means abstraction and code means realization, these models range from abstract to concrete representations of the application.

MDA defines three models that represent different views of the systems (from abstract to concrete):

- The *Computation Independent Model* (CIM), a domain model of the application.
- The *Platform Independent Model* (PIM), a model of the code that omits any detail that could tie the software model to any specific implementation platform.
- The *Platform Specific Model* (PSM), a PIM where specific implementation details have been included.

Because these models can be seen as an evolution of the others a mechanism for evolution from one model to another would be desirable. OMG provides *Query/View/Transformation* (QVT) a transformation language [2] which caters for the transition.

There are several choices for defining the transformation between models [1], but one of the most popular is defining transformation rules among the metamodels that describe the models that have to be related. *Metamodels* are just models that can be used to describe other models. Thus, we could use the *Entity-Relationship Model* (E-R Model) [3] to describe *UML class diagrams* [4]. Using E-R metamodeling language, we could define the entities `Class` and `Attribute`, and the relationship `classHasAttribute`, which relates the entity `Class` with the entity `Attribute`.

As we have done with the E-R model, we would like to use UML class diagrams as a metamodeling language. However, these diagrams have a very complex metamodel themselves, and, to make matters worse, the semantics of UML are object-oriented classes. Thus, if we provide a UML metamodel of the Entity-Relationship Model, and we define the class `Entity`, the class `Attribute` and an aggregation relationship between the class `Entity` and the class `Attribute`, we would be thinking of two Java (or C++) classes called `Entity` and `Relationship`, where the first has the second as an attribute.

To overcome these drawbacks OMG has defined a *UML Infrastructure* [5] a common modeling core that can be reused by UML and by MOF (*Meta-Object Facility*) [6]. MOF is a metamodeling language that basically provides a simplified version of UML class diagrams as a metamodeling language.

This core also defines four levels of metamodeling [5]:

- M3: Meta-metamodels (e.g. MOF).
- M2: Metamodels (e.g. the UML metamodel described as an instance of MOF).
- M1: Model (e.g. a UML model of a web shop described as an instance of UML).
- M0: Runtime instances (e.g. an object instance of a Java class depicted in the UML model of a web shop).

In addition, this core is extended with a profiling facility defined at M3 level. This facility allows any M2 MOF-defined metamodel to be enriched with stereotypes. A *stereotype* is an M2 metaclass that can be related to any M2 metaclass of the metamodel defined in terms of an instance of the M3 MOF meta-metaclass. Thus, using this relationship, we could include stereotypes in M1 as tags inside the instances of the M2 metaclasses. For example, at M2, the stereotype `input` could be related to the M2 UML metaclass `Action`. In this way, the tag `<<input>>` could be included in actions forming part of UML activity diagrams, as in [7].

MDA is the framework provided by OMG. An alternative approach that follows MDA principles, but independently of MDA standards, is *Model-Driven Development* (MDD). Thus, the terms MDA and MDD are often used interchangeably. MDD refers to the activity that is carried out by software developers. MDA is reserved for its formal OMG definition, which focuses more on creating a formal framework in which MDD can operate [8].

At this stage it should be evident that anyone interested in working with MDD will have to deal with metamodels. In some cases these metamodels can be predefined by an organization, such as the UML metamodel [4]. In other cases these metamodels will be defined with reference to a language, such as *Web Modeling Language* [9]. Finally, in other cases an existing metamodel can be tailored to be used in a specific domain, as in the case of *UML Web Application Extension* (UML WAE) [10].

Therefore, the rule of thumb for the use of metamodels in MDD is very simple:

1. If you have a valid metamodel, then reuse it.
2. If you don´t have a valid metamodel, then define one.
3. If you have a metamodel that would be valid if it had some specific tags in some elements, then customize it using a profile.

However there are cases where either a new metamodel can be defined or an existing metamodel can be customized using profiles. In these cases some rules have been defined [11] for choosing one or another approach. However, these rules mainly focus on the syntactical properties of the languages instead of the practical details of both approaches. Other approaches focus mainly on UML profiles [12].

This paper describes our practical experience during the definition of an MDD approach for describing the navigational maps of web applications. Section 2 briefly describes our notation. Section 3 describes the UML profile that defines it. Section 4 describes the alternative NMM profile. Section 5 analyzes some similar approaches in the web engineering domain. Finally Section 6 provides conclusions and details of future work.

2 NMM Notation

Navigation Maps Modeling (NMM) notation is an abstract notation for the description of navigation maps [13]. Navigation maps provide an overall view of a web application for an audience [14]. A navigation map describes the possible sequences of web pages displayed to a user, and is typically a part of the documentation of a web application [15]. At present, many web sites include navigation maps to help users during browsing, which makes their definition a key issue during the development of web applications [16]. Using navigation maps, developers can obtain an overall view of the whole application that can help them during the development process. In addition, the presence of navigation maps can help the users of web sites to find the information they want much more quickly.

NMM uses three diagrams to describe navigation maps:

- The *Page diagram* describes the structure of web pages and links in the application.
- The *Region diagram* describes the presentational regions of the user interface.
- The *Mixing diagram* assigns pages to regions.

Although NMM is formalized [13], it uses a graphical notation for describing page and region diagrams. The Mixing diagram defines the assignation of pages to regions using colors.

The elements of the NMM notation are formalized using first order logic and mathematical statements, although some transformation rules have been defined to translate abstract NMM diagrams to detailed designs expressed as UML WAE diagrams [13].

However, the NMM metamodel had not been formalized in terms of MOF (or Ecore [17]) elements, or the transformation rules had been defined using QVT (or ATL [18]) rules.

We have provided an MDD representation for NMM notation. The main question was what representation language to choose. MOF metamodels or UML profiles?

The following sections analyze both options as well as their advantages and drawbacks.

3 NMM MOF Metamodel

The first choice considered was to provide an MOF metamodel for NMM notation. For the sake of conciseness, this metamodel makes a package merge of UML `Core::Constructs` [5]. Fig. 1 depicts the NMM MOF metamodel for page diagrams.

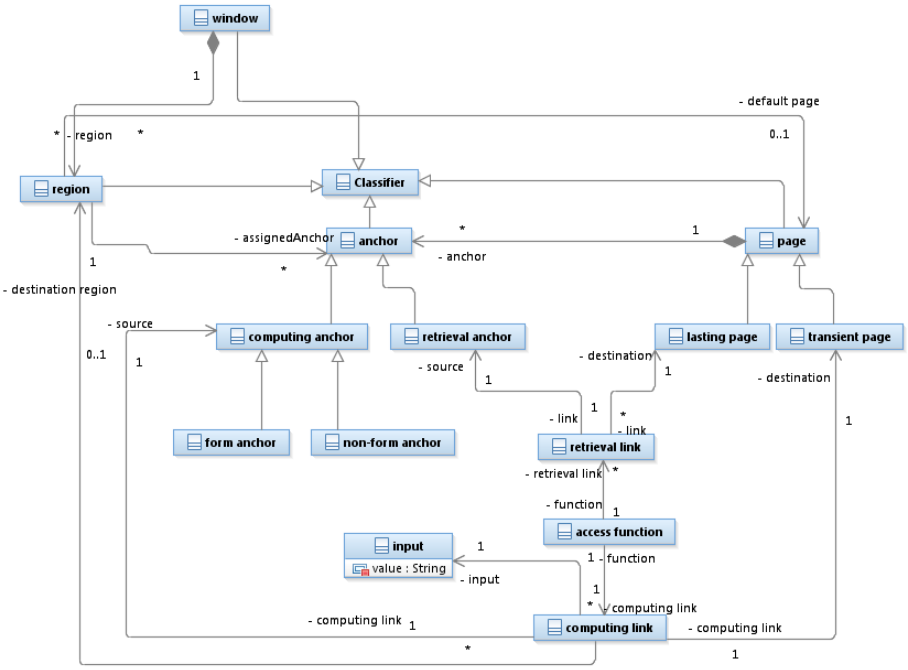

Fig. 1. MOF metamodel for NMM notation. For the sake of conciseness `Class` and `Association` are defined in UML `Core::Constructs` and merged in the NMM metamodel.

This metamodel provides a compact description of the abstract syntax of NMM notation. In order to use them in the context of a CASE tool, *Eclipse Graphical Modeling Framework* (GMF) [19] seems to be the most reasonable alternative.

In order to use such a framework, taking into account the Ecore representation of the domain model (i.e. the Ecore serialization of the MOF metamodels depicted in Fig. 1), three additional steps must be performed in order to define a CASE tool for NMM notation [20]:

- To provide a *graphical definition*, which defines the visual aspects of our generated editor.

- To provide a *tooling definition*, which comprises features related to editor palettes, menus, etc.
- To provide a *mapping definition*, which defines the mapping between the business logic (the Ecore domain model) and visual models (graphical and tooling definitions).

The NMM metamodel defined in Fig. 1 was generated using *IBM Rational Software Architect* CASE tool [21]. Another choice would have been to use *Eclipse Ecore Tools* [22] for the graphical definition of an Ecore NMM metamodel.

Regarding the advantages of this approach, we would emphasize:

- The descriptive power of the NMM MOF metamodel, which allows a custom definition of a new notation.

Regarding the drawbacks of this approach, we would mention the following:

- The need for knowledge of a metamodeling language (e.g. MOF or Ecore). Of course, this language is very close to UML class diagrams, but the use of UML class diagrams for modeling anything different that object-oriented code is not straightforward.
- The use of two different tools:
    - Eclipse Ecore Tools or another CASE tool.
    - Eclipse Graphical Modeling Framework.
- Dependence on two different frameworks that do not have integrated CASE tool support by any commercial brand:
    - Eclipse Modeling Framework.
    - Eclipse Graphical Modeling Framework.
- The lack of usability of these diagrams by a general-purpose UML CASE Tool.

4 NMM UML Profile

The other obvious choice was to define a UML profile for characterizing NMM notation. Although there are several CASE tools that support the definition of profiles for UML we chose *Borland Together* [23] because it includes an integrated environment for the definition of UML profiles, and the execution of QVT operational mapping transformations.

Fig. 2 depicts the complete UML profile for NMM notation. We have used the profile notation used by Borland Together instead of the graphical notation used in the UML Infrastructure. The idea is to show the expressive power of the notation used by the tool. Other notation used by other tools (e.g. IBM Rational Software Architect) is similar.

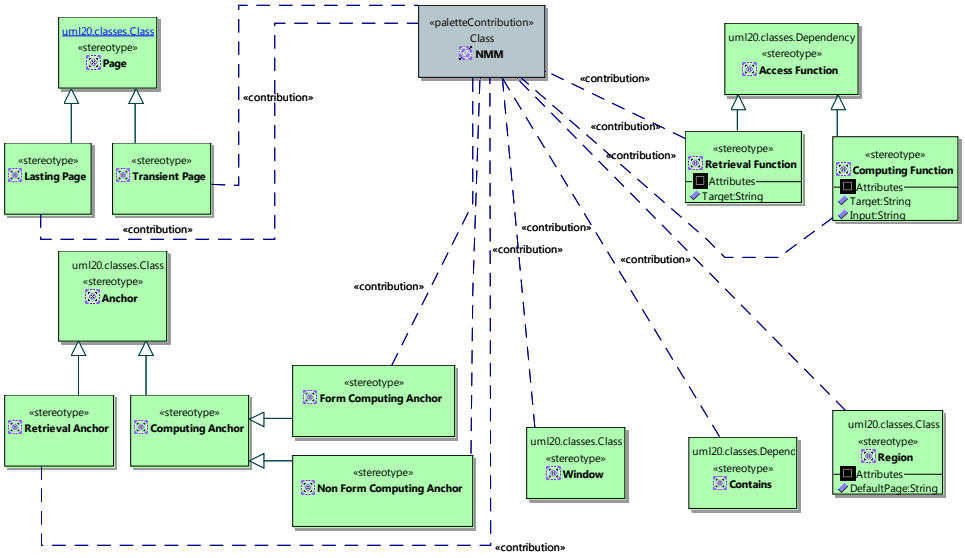

Fig. 2. UML profile for NMM page and region diagrams as defined in Borland Together.

No profile for the mixing diagram was needed because the assignation of pages to regions is made using the attribute `DefaultPage` of stereotype `Region` and the attribute `Target` of the stereotypes that specialize to the stereotype `Access Function`. The attribute has not been included in the stereotype `Access Function` itself for technical details related to the definition of QVT operational mapping transformations performed by the CASE tool.

Regarding the advantages of this approach we would emphasize:

- Deep knowledge of metamodeling languages is not needed to define a UML profile.
- The availability of a commercial CASE tool (Borland Together) that allows the definition of this profile and its use, together with QVT transformations, in the same tool. Of course, Eclipse could be used for metamodeling definition and for the execution of QVT transformations, but, in most cases, Eclipse plugins are not stable, being always under constant development. Therefore, a production environment based on the combination of different under-development projects it doesn't seem very reliable. In any case, a discussion about the advantages/disadvantages of commercial software vs. open-source is out of the scope of this paper.
- The integration of these profiled UML models with other UML models that can describe: (i) more detailed designs for the presentation tier using UML WAE; and (ii) the design of the remaining tiers using plain UML [13].

Finally, the main drawbacks of the profile-based approach are:

- The lack of descriptive power provided by the profile definition notation used by the CASE tool.
- The cost of the license of Borland Together.

5 Related Work

The use of MDD has found great acceptance in the web engineering community engaged in the design of web applications. This section analyzes the decision taken regarding the approaches studied in terms of the definition of a metamodel vs. the definition of the profile, and the CASE support available. Therefore, the analysis is not focused on the comparison of NMM with them [13].

UML-based Web Engineering (UWE) [24] has defined metamodels for the development of *Rich Internet Applications* (RIAs) [25], and in general for the use of MDD using UWE [26]. In most cases, these metamodels have a UML profile representation that makes them suitable for use in a UML CASE tool. However, UWE uses its proprietary CASE tool [27], or has developed its own Eclipse plugins [28].

Ubiquitous Web Application (UWA) uses an MOF-compliant metamodel [29]. UWE has an MDD approach that allows UWA models to be mapped to specific MVC-based designs using their own GMF-based CASE tools [30].

WebSA [31] is an MDD extension to different notations. WebSA proposes an MDD made up of a set of UML architectural models and QVT model transformations as a mechanism for (i) integrating the functional aspects of the current Web methodologies with the architectural models and for (ii) defining a set of transformations from the architectural models to platform-specific models. In particular, the approach has been tested with OO-H [32]. OOH4RIA [33] is an extension that depicts a process model for RIAs. OOH4RIA uses MOF metamodels and UML profiles as metamodeling facilities. For CASE support, OOH4RIA uses its proprietary tool.

Web Modeling Language (WebML) is a modeling language that has strong links with MDD [34, 35]. WebML has both an MOF metamodel and a UML Profile [9]. This allows it to be used with a standard CASE tool, or with its proprietary WebML CASE tool [36].

Finally, *UML Web Application Extension* (UML WAE) [10] was defined before the MDA era and, therefore, it does not provide an MDD approach. However, it is defined using a UML profile that makes it usable with any UML-compliant CASE tool that can understand UML profiles.

Table 1 summarizes this analysis.

Table 1. Practical MDA characteristics of notations analyzed.

Notation	Metamodel	UML profile	Proprietary CASE tool	UML CASE tool	Application Generation
NMM	yes	yes	no	yes	no
OOH4RIA	yes	yes	yes	no	yes
UML WAE	no	yes	no	yes	no
UWA	yes	no	yes	no	yes
UWE	yes	yes	yes	yes	yes
WebML	yes	yes	yes	yes	yes

6 Conclusions and Future Work

MDD is widely accepted by the web engineering community for the development of design notation for web applications. It seems to be very common to define a MOF-based metamodel to describe the notation and then to provide a UML profile to facilitate its use in a general purpose UML CASE tool.

However, in order to apply the transformation and generate code, most of the approaches use their own CASE tool, making the UML profile unnecessary. Thus, although MDD provides an opportunity for web notation to dispense with proprietary CASE tools, the lack of integrated support for MDD in most CASE tools, and the power of the code generation capabilities of these approaches, have hindered the use of standard CASE tools in web engineering.

In other cases web notations have developed their own Eclipse plugins, bringing their notations closer to CASE standards. However, the main drawback of this approach is the lack of a commercial integrated development environment.

The work presented in this paper has analyzed the advantages and drawbacks of defining a metamodel vs. defining a UML profile. We have chosen the profile solution because it allows the use of our notation with standard UML CASE tools. In addition, the use of Borland Together allows us to have a commercial integrated MDD environment. The major drawback is the cost of the tool and dependence on a specific tool. However the XMI [37] export facilities of the CASE tool and the use of QVT standards allow migration to Eclipse-based development. This migration is easy because the NMM approach has not the powerful application-generation capabilities that the other analyzed notations have.

Future work includes the definition of OCL constraints in order to guarantee the syntactical correctness of the NMM models, and the inclusion of additional abstract diagrams that permit the description of other tiers of the web application, as well as their transformation to UML detailed designs.

Acknowledgments. *El Ministerio de Ciencia e Innovación* (TIN2009-14317-C03-01) and *La Universidad Complutense de Madrid* (Group 921340) have supported this work.

7 References

1. OMG MDA Guide v1.0.1 (2003), http://www.omg.org/cgi-bin/doc?omg/03-06-01
2. OMG Meta Object Facility (MOF) 2.0 Query/View/Transformation V1.1 (2011), http://www.omg.org/spec/QVT/1.1
3. Chen, P.P.S.: The Entity-Relationship Model. Toward a Unified View of Data. ACM Transactions on Database Systems 1 (1), 1976.
4. OMG Unified Modeling Language (UML) Superstructure, V2.3 (2010), http://www.omg.org/spec/UML/2.3/Superstructure/PDF/
5. OMG Unified Modeling Language (UML) Infrastructure, V2.3 (2010), http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/

6. OMG Meta Object Facility (MOF) Specification, V2.0 (2006), http://www.omg.org/spec/MOF/2.0/PDF/
7. Navarro, A., Merino, J., Fernández-Valmayor, A. and Cristóbal, J.: Translating Workflow Diagrams into Web Designs. In SEKE 2008, pp. 667--672
8. Gardner, T., and Yusuf, L.: Explore model-driven development (MDD) and related approaches: A closer look at model-driven development and other industry initiatives (2006), http://www.ibm.com/developerworks/library/ar-mdd3/
9. Moreno, N., Fraternali, P. and Vallecillo, A.: WebML modelling in UML. IET Software 1, 67--80 (2007)
10. Conallen, J.: Modeling Web Application Architectures with UML. Communications of the ACM 42, 63--70 (1999)
11. Brucker, A.D. and Doser, J.: Metamodel-based UML Notations for Domain-Specific Languages. In: 4th International Workshop on Language Engineering, (2007)
12. Selic, B.: A Systematic Approach to Domain-Specific Language Design Using UML. In ISORC 2007 (2007)
13. Navarro, A., Fernández-Valmayor, A., Fernández-Manjón, B., and Sierra, J.L.: Characterizing navigation maps for web applications with the NMM approach. Science of Computer Programming 71, 1--16 (2008)
14. Abrahão, S.M., Olsina, L. and Pastor, O.: Towards the quality evaluation of functional aspects of operative web applications. In: Proc. IWCMQ 2002, pp, 325--328 (2002)
15. Han, M. and Hofmeister, C.: Separation of navigation routing code in J2EE web applications. In: ICWE 2005, pp. 221--231 (2005)
16. Navarro, A., Sierra, J.L, Fernández-Valmayor, A. and Fernández-Manjón, B.: Conceptualization of navigational maps for web applications. In MDWE 2005, pp. 80-88 (2005)
17. Budinsky, F., Steinberg, D., Merks, E. Ellersick, R. and Grose, T.J.: Eclipse Modeling Framework: A Developer's Guide, Addison-Wesley Professional (2003)
18. ATL Transformation Language (2011) http://www.eclipse.org/atl/
19. Eclipse Graphical Modeling Project (2011) http://www.eclipse.org/modeling/gmp/
20. Aniszczyk, C.: Learn Eclipse GMF in 15 minutes (2006) http://www.ibm.com/developerworks/opensource/library/os-ecl-gmf/
21. IBM Rational Software Architect CASE Tool (2011) http://www.ibm.com/developerworks/rational/products/rsa/
22. Eclipse Ecore Tools (2011) http://wiki.eclipse.org/index.php/Ecore_Tools
23. Borland Together CASE Tool (2011) http://www.borland.com/us/products/together/index.aspx
24. Hennicker, R., Koch, N.: Systematic Design of Web Applications with UML. Unified Modeling Language: Systems Analysis, Design and Development Issues, pp. 1--20 (2001)
25. Kroiss, C., Koch, N. and Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. In: ICWE 2009, pp. 493--496 (2009)
26. Kraus, A., Knapp, A. and Koch, N.: Model-Driven Generation of Web Applications in UWE. In MDWE 2007, (2007)
27. Knapp, A., Koch, N., and Zhang, G.: Modelling the Behaviour of Web Applications with ArgoUWE. In ICWE 2005, pp. 624--626 (2005)
28. Busch, M. and Koch N.: MagicUWE - A CASE Tool Plugin for Modeling Web Applications. In: ICWE 2009, 505--508 (2009)
29. Distante, D., Pedone, P., Rossi, G. and Canfora G.: Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces. In: ICWE 2007, 457--472 (2007)
30. Bernardi, M.L., Cimitile, M., Distante, D. and Mazzone, F.: Web applications design evolution with UWA. In WSE 2010, pp. 3--12 (2010)
31. Meliá, S. andGómez, J.: The WebSA Approach: Applying Model Driven Engineering to Web Applications. Journal of Web Engineering 5, 121--149 (2006)

32. Gómez, J., Cachero, C., and Pastor O.: Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia 8, 26--39 (2001)
33. Meliá, M., Gómez, J., Pérez, S. and Díaz, O.: Architectural and technological variability in Rich Internet Applications. IEEE Internet Computing Journal, 14, 24—32 (2010)
34. Ceri, S., Fraternali, P, and Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks 33, 137--157 (2000)
35. Ceri, S., Brambilla, M., and Fraternali P.: The History of WebML Lessons Learned from 10 Years of Model-Driven Development of Web Applications. Conceptual Modeling: Foundations and Applications, pp. 273-292 (2009)
36. Acerbis, R., Bongio, A., Brambilla, M., Butti, S., Ceri, S., and Fraternali, P.: Web Applications Design and Development with WebML and WebRatio 5.0. In TOOLS (46), pp. 392--411 (2008)
37. OMG MOF 2.0/XMI Mapping, Version 2.1.1 (2007), http://www.omg.org/spec/XMI/2.1.1/PDF/