A MODEL-DRIVEN HYPERMEDIA DEVELOPMENT METHOD

Carlos Solís, José H. Canós, María C. Penadés, Manuel Llavador
Department of Computer Science (DSIC), Technical University of Valencia
Camino de Vera s/n, 46022, Valencia, España.
E-mail {csolis, jhcanos, mpenades, mllavador}@dsic.upv.es

ABSTRACT

This paper introduces the Model-Driven Hypermedia Development Method (MDHDM). It is a software process model that applies Model-Driven Engineering (MDE) to the generation of hypermedia applications starting from both process models and object oriented conceptual models. The method provides complete guidance for all the phases of the software development lifecycle, from the analysis phase to the implementation phase. MDHDM benefits from the automation of the software development process provided by MDE. Formal transformation rules have been defined to automatically generate navigational elements from process and conceptual models. MDHDM starting point is the process model. It is suitable for modelling Business Process and Safety Oriented Systems. In addition, MDHDM provides template patterns for generating the final hypermedia application from the navigational model, as well as a tool that support the method.

KEYWORDS

Hypermedia, Web Engineering, Model-Driven Engineering

1. INTRODUCTION

Hypermedia design methods (HDMs) such as HDM (Garzotto, 1993) and RMM (Isakowitz, 1995) emerged in the 1990's to face new challenges that traditional software engineering methods were not able to deal with. Easy navigation over complex information spaces, the diversity of potential users (especially with the advent of the World Wide Web), and session control are the most relevant features to be supported by HDM methods and tools. HDMs are structured around a set of stages in the development lifecycle. Each stage produces artefacts that are obtained from other artefacts generated during earlier stages. The artefacts are produced by applying a number of heuristics, which are not formally defined nor automated. The lack of formalization generates ambiguous interpretations of these heuristics, whereas the lack of automation forces the development teams to deal with many different technologies to deploy an application. Moreover, from the developer's point of view, it is unclear how to translate the models generated during the lifecycle to a technological platform.

In this paper, we show how these drawbacks are solved using a Model-Driven Engineering (MDE) approach (Kent, 2002). We provide a well-defined process that generates a hypermedia application by performing transformations over a set of models following precise rules. Our proposal is called the Model-Driven Hypermedia Development Method (MDHDM). Specifically, MDHDM is a software engineering method for the analysis and design of hypermedia and web systems. The main difference between MDHDM and the other methods that follow a similar approach is that the transformations between software artefacts are performed using a metamodel transformation language. The metamodels are defined following the Meta Object Facility (MOF) framework (OMG, 2004), and the transformations are defined in the Query-View-Transformations (QVT) language (OMG, 2005). We have defined a Conceptual Metamodel, a Process Metamodel, a Navigational Metamodel (which includes the presentation to the user), and a Concrete Hypermedia Metamodel as well as their transformation functions. All of these form part of a hypermedia development method following a MDE approach. Another difference between MDHDM and other HDM methods is that the starting point is the Process Model. Few hypermedia applications consider only the navigational aspect. Navigation is performed in order to achieve a specific goal, therefore, navigation is

performed to complete a process. The Process Model is more important than the structure provided by the Conceptual Model, and, in domains such as business processes, and safety-oriented systems it is essential.

The rest of this paper is organized as follows. Section 2 describes some classical hypertext design methods found in the literature. Section 3 gives an overview of Model-Driven Engineering (MDE) and Model-Driven Architecture (MDA) (OMG, 2003, Mellor, 2004) and their relationship. Section 4 presents the MDHDM and describes both the PIM and the PSM artefacts, and the transformations between the models mentioned above. Finally, Section 5 presents some conclusions and future work.

2. RELATED WORK

According to classical Software Engineering (SE) principles, considering hypermedia as software means, among other things, that its development must be performed following a given process model that consists of a number of stages which are followed either sequentially or iteratively. Proposals like HDM (Garzotto, 1993), RMM (Isakowitz, 1995), OOHDM (Schwabe, 1996), WebML (Ceri, 2000), and UWE (Koch, 2000) include different stages, but in most of these methods the following steps are included (by order of application): conceptual modelling, navigational or access structure design, abstract user interface design, and implementation. After each stage, one or more artefacts are generated; the evaluation of these artefacts can produce feedback to the preceding stages. In the following paragraphs, we outline the purpose, inputs, and outputs of these stages.

Conceptual Modeling. As in classical SE, Conceptual modelling/Analysis is the first stage in the development process. The output is a representation of a real problem called the Conceptual Model. It is sometimes expressed as an entity-relationship diagram, as in RMM, HDM and WebML, as it is sometimes expressed as a class diagram in an object-oriented (OO) model, as in OOHDM and UWE.

Navigational Design. The navigational design starts when a designer chooses which classes and associations of the Conceptual Model can be navigated by a set of users. Navigation paths are normally derived from the relationships between the elements in the Conceptual Model (e.g. associations in OO models). The Navigational Model is completed by defining access structures such as indexes, menus, tours, and search forms to allow users to have selective access to information. The type of access structure suitable for a given relationship in the Conceptual Model can be derived from the cardinalities in the associations (see e.g. Isakowitz, 1995).

Presentation Design. It is part of the hypermedia design and represents the final interfaces that the users interact with when they navigate. In this stage, most methods design abstract user interfaces (AUIs), that is, the way the user will interact and perceive the data presentation; the components of an AUI are text fields, buttons, images, etc.

Implementation. It is the final stage and is dependent on the target technology; it produces the actual hypermedia document that will be deployed according to the architecture defined for the system.

The Process Modelling is not considered in the stages described above, although it is necessary for guiding the user navigation. The process dimension is far more important than the structure, and most of the knowledge about a domain is determined by the tasks performed by the users.

In addition, in these methods the transformations between the models are described in a heuristic way and lack formalization, which results in ambiguous interpretations by the developers. Furthermore, there are not guidelines to the developers during the implementation, and the transformation patterns from abstract hypermedia models to a concrete implementation are not provided. All of these problems keep the developers from using models successfully.

3. PRINCIPLES OF MDE

MDE (Kent, 2002) is an approach to software development based on models, metamodels, and transformations. A model is an abstraction of the world described in a well-defined language (Kleppe, 2003). A metamodel is an ontology designed to build kinds of models, i.e., it is a model of a modeling language (Kleppe, 2003). Mapping functions or transformations are applied to the models as instances of the metamodels. A transformation is composed by a set of rules that is defined using the elements of the source

and target metamodels. Thus, a model-driven software development process is a chain of model transformations (a composition of transformation functions), which covers all the development phases. In MDE, software is developed from abstract models that are refined to more concrete models until a point is reached, where it is easy to generate source code. The refining steps should be as automated as possible. A MDE method must contain all the elements that are usually present in any software development method. Nonetheless, there are some differences: all the artefacts are represented using a well-defined modeling language or metamodel, and the frameworks that implement processes of this kind use specialized model transformation tools.

The Model-Driven Architecture or MDA (OMG, 2003) is the MDE approach proposed by the Object Management Group (OMG). The key elements in MDA are the Platform Independent Models (PIMs) and the Platform Specific Models (PSMs). A PIM is used to describe the problem domain with the user's vocabulary. A PSM describes a system using a specific technology. The basic goal of MDA is to describe a system using PIMs, transform them into PSMs, and generate a deployable application from the PSMs. The interfaces between PSMs are called bridge models. MDA is a subset of a MDE process because it only takes into account PIM to PSM transformations. However, in MDE, there are PIM to PIM, and PSM to PSM transformations.

OMG has defined a model management framework called Meta-Object Facility (MOF) (OMG, 2004). Its basic function is to define the PIMs and the PSMs in a standard way. MOF is divided into four metalevels (Table 1).

Table 1. MOF metalevels.

Metalevel	Function	Example
M0	Represents the data level	Instance of Person: "John Smith"
M1	A set of model instances of the metamodels in M2	A UML model with Class Person, etc.
M2	A set of metamodel instances defined using MOF	UML metamodel with the UML elements: Class, Association, Specialization, etc.
M3	Defines the MOF metamodel	A MOF Class, Association, etc.

The MOF metamodel is in M3, which is the top of the reflective tower. MOF is used to define the metamodels in layer M2. The metamodels in M2 are used to define the models in M1, and the models in M1 are used to generate the system's code in M0. MOF also defines a standard interface to a MOF model repository and a XML schema to exchange models (XML Metadata Interchange XMI). Once models and metamodels are formalized in MOF, it is possible to define the transformations among them. Even though MDA does not state how to transform a PIM into a PSM, it states that transformations can be done automatically by programs. Later, the OMG defined a standard transformation language: Query/Views/Transformations (QVT) (OMG, 2005), which defines two languages to transform models: QVT Core and QVT Relations.

OOHDMDA (Schmid, 2005) transforms a hypermedia PIM model to a PSM servlet model. The transformation is based on the XMI representation of the models. This kind of transformation is built on the persistent representation of the models. In other words, it is based on syntactical elements instead of metamodel elements and metamodel transformation languages, which work on a semantic level.

4. OVERVIEW OF MDHDM

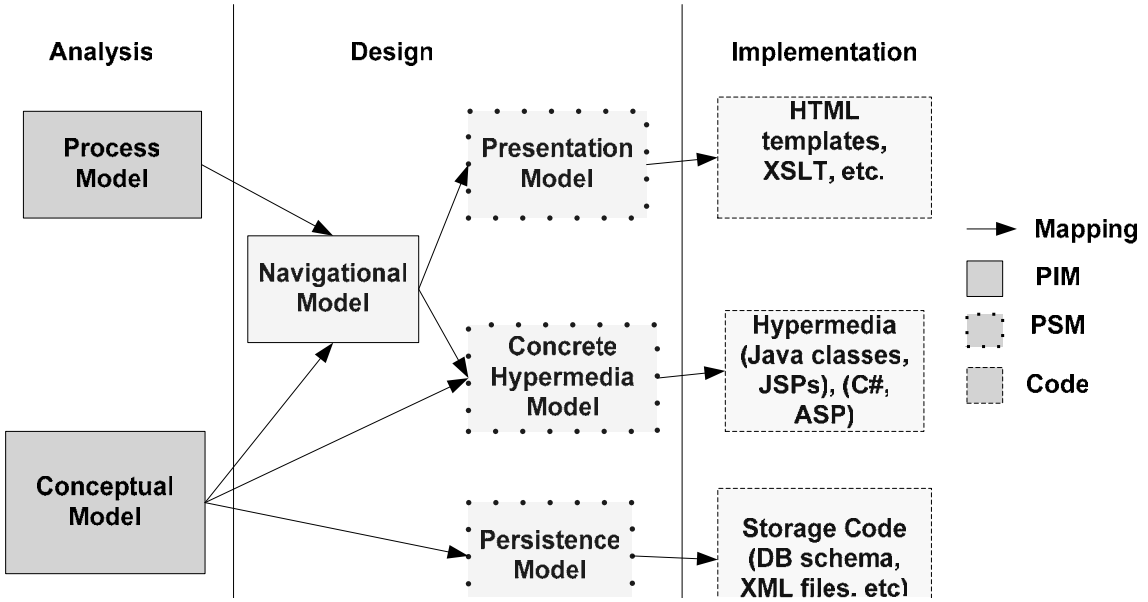
The MDHDM software process is summarized in Figure 1, where the different models used, and the transformations are shown. PIM and PSM models are artefacts created at some phase in the software process. The Process Model and the Conceptual Model are created at the analysis phase, whereas the Navigational Model, the Presentation Model, the Concrete Hypermedia Model and the Persistence Model are built during the design phase. In the following, PIMs, PSMs, transformations, and the supporting tool are introduced.

4.1 Platform Independent Models

In MDHDM, the two starting PIMs are the Process Model (PM) and the Conceptual Model (CM), which are built in the analysis phase of MDHDM.

The Process Model (PM) represents the problem domain from a dynamic point of view; it is a typical workflow model (with activities linked by control and data flows, and actors responsible for the execution of the activities). The PM is the basic model in MDHDM, and is included in some but not all of the methods. The PM is the starting point in MDHDM. The process model is used because the navigation paths are, mainly, determined by the control flow of the activities that compose the process. The MDHDM notation is the Business Process Modeling Notation (BPMN) (OMG, 2006). We can model the control and data flows using BPMN. The elements used are activities (manual and automatic), or-joins, or-splits, and-splits, and-joins, and subprocesses.

Figure 1. The MDHDM Software Process.



The Conceptual Model (CM) in our proposal follows OO principles. The CM describes the structure of the system, which is made up of a set of classes and the relationships among them (generalization/specialization, association and/or aggregation) (Rumbaugh, 1991).

The Navigational Model (NM) is a PIM obtained from the PM and CM. As in other hypermedia development methods, the NM represents the navigational elements (classes, indexes, tours, links, redirections, task views, and others) in the application. There are two types of navigational elements: the structural navigation elements, which are derived from the classes and relations in the CM; and the process navigation elements, which are obtained from the control and data flows in the PM. The structural navigational model is similar to the UWE navigational model (Koch, 2000).

The navigational model is obtained using a PIM to PIM transformation function applied to the Process Model, and the Conceptual Model. The MDHDM navigational metamodel has two types of links: process links and structural links. The structural links are derived from the associations in the CM. The process links are derived from the PM, and can activate automatic tasks and advance the process state. The Navigational Model is obtained in the design stage of MDHDM. The derived NM can be enriched with more navigational paths and structures. We have defined the metamodels of PM, CM, and NM as MOF metamodels.

4.2 Platform Specific Models

The NM is the source from which two PSMs are generated: the Presentation Model (PreM) and the Concrete Hypermedia Model (CHM). The Persistence Model (SM), another PSM, is created from the CM.

The final step, the code generation or implementation of the hypermedia application is performed using code generation patterns from these PSMs.

The PreM represents the user interface model. Typically, it is composed of data presentation elements such as text fields, buttons, images, etc. The PreM is built from a set of sample HTMLs and XSLT transformations. The XSLT transformations are derived semi-automatically, using the XSMapper (Llavador, 2006), the XML representation of the model instances, and the XML schemas of the samples.

The SM is a representation of the persistence medium of the conceptual model. This model is not present in the other methods. The SM is necessary because the Conceptual Model instances are stored in an information medium. The SM elements are retrieved by the CHM instances that perform the Navigational Model. The transformations from OO Conceptual Models to persistence technologies like RDBMS, OODBMS, or XMLDBMS are well known (Abiteboul, 2000, Rumbaugh, 1991).

The CHM represents a dynamic Hypermedia Technology, that despite being neutral, is close to web technology because it is the most widely used target platform for hypermedia. The CHM is conformed by a set of behavioral patterns of the hypermedia elements, which describes the interaction of the navigational elements with the other PSM models, namely the PreM and the SM. The elements in the CHM have a specific behavior that is expressed with UML sequence diagrams, and provide a guideline for the developers. A CHM is a generic PSM, which allows us to generate JSP, PHP and other languages. The CHM element behavior is a sequence of invocations to functions in the bridge models. For example, the behavior of a navigational class is as follows: the navigational class should receive an Oid (object id) in its request. Then, using the bridge it should ask the SM, for the object corresponding to the Oid, get the XML representation of the object, and apply a XSLT transformation to obtain the final presentation. The most relevant point is that the interface functions defined by the CHM permit the exchange of the SM and the presentation without secondary effects in the CHM model and its implementation.

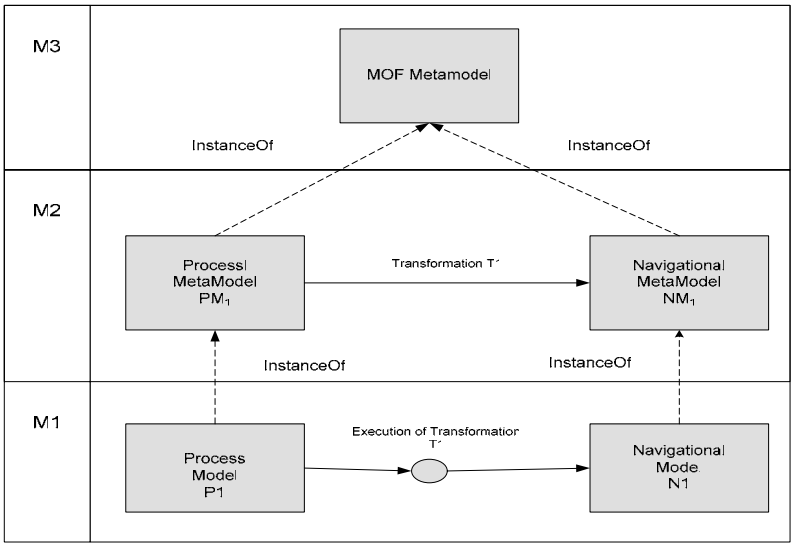
4.3 Transformations.

In a MDE method, the generation of artefacts from stage to stage is seen as a model transformation. In order to define model transformations, a transformation language is needed. The natural candidate when using MDA is Query-View-Transform (QVT) (OMG, 2005). However, currently there is no implementation available. Therefore, we selected ATL (ATLAS group, 2005) to describe the transformations, as it achieves the best balance between practicality and conformance to MOF/QVT. ATL is a high-level language that uses OCL queries for model transformations. It also provides programming and debugging tools. We have formally defined an ATL transformations from the PM and CM to the NM, another transformation from NM to the CHM, as well as the code generation patterns. The only model that is not transformed according to a metamodel transformation is the PreM.

The transformations are based on metamodels and are applied to models. Figure 2 shows a MOF transformation. The MOF metamodel is defined in metalevel M3. The PM and CM metamodels are defined using MOF elements in the metalevel M2. According to MDA, the metamodels in M2 are PIM or PSM metamodels. In this example PM1 and NM1 are PIM metamodels. In metalevel M2, there is one transformation function: T1, which uses the elements defined by PM1 and produces a Navigational Model that is defined according to the elements in NM1. In metalevel M1, there is a Process Model P1, which is an instance of Metamodel PM1. When the transformation T1 is applied to P1, then a Navigational Model N1 is obtained.

The PM transformation uses the data input and output of the activities and the process control flow to derive a NM. The first step is to project the swim lane of each actor in the process. A swim lane is a view of the process where only the activities of a single actor are shown.

Figure 2. A MDHDM transformation using the MOF framework.



Manual task. A manual task is transformed into a Task View in the navigational model. The manual tasks are classified into two categories: simple and composed. The simple manual tasks are: the data entry task, the selection task, the view/edition of process variables, and confirmation task. A data entry task is transformed in a data entry form. A selection task can select an object or a set of objects, the selection an object can be performed in the hypermedia node that represents or points to the same kind of object either: a navigational object, an index or a tour. The selection of object collections can be only performed using indexes. A manual task to edit or view a process variable is mapped to a navigational class that represents the process variable. A confirmation task indicates that an external task must be done or a command to execute an automatic task is waited. A confirmation task is mapped into a Task View with an advance link.

Automatic Task. Automatic tasks do not have a mapping in the hypermedia world, but they have to be performed if a preconditioning process link is clicked by the user.

Or-split. Or-splits are transformed into redirection elements if they are automatic; however a manual or-split (which is called deferred or-split in the business process terminology) is transformed into a menu of process links in the hypermedia model.

And-split. And-splits are transformed into windows, so that the user can perform the navigation in parallel windows.

Or-join. Or-joins do not have a mapping in the hypermedia model. The Or-joins disappear in the Navigational model, and the input nodes of an Or-join must be connected to its output node.

And-join. And-joins are mapped to the event of closing a window. However, in the navigational model, the And-joins disappear, and the input nodes of an And-join must be connected to its output node.

Process links. The process links are shown if a hypermedia node matches the data needed for a process activity. If a hypermedia node contains a set of information that can be mapped to the activity input, then a process link has to be shown in the node. For example, in an e-commerce process for buying products, the "add to shopping cart" activity receives a product as input. During the "article selection" activity, a process link to the "add" activity has to be shown in the access structures that point to products, and/or in the product nodes.

Subprocess. A subprocess is mapped using the mappings of its individual elements.

The transformation rules from the CM to the NM are similar to the heuristics shown in (Isakowitz, 1995). These rules are based on the creation of access structures from the relationships. We have extended the set of rules with derived associations, star associations, and home page creation. Some of the typical transformations from the CM to the NM are:

Class Partition. The set of class attributes can be divided in several groups. The groups are formed for two reasons: the attributes in a group share a specific characteristic or a Human Computer Interaction (HCI)

rule is applied. A main group is selected and the class associations are related to this group. The mark used is the tagged value partition and the group name; the attributes without a partition remain in the main group.

Class Fusion. The fusion of classes can be applied to a relation with cardinality 1-1. The mark Fusion is indicated in one of the association-ends. As a result, there is only one class in the navigational model which shows the attributes and associations of the other class.

Association 1-1 to Navigable Association 1-1. An association with cardinality 1-1 is transformed into a 1-1 navigable association (a simple hyperlink).

Association 1-N to Access Structure. An association with cardinality 1-N is transformed into a navigable association to an access structure, and into another association from the access structure to the associated navigable class. The default access structure is an Index; however if the marking value Access is used, then any available access structure in the Navigation Metamodel can be indicated.

Association M-N to Access Structure. The transformation for an association 1-N is applied twice.

The only model that is not transformed according to a metamodel transformation is the PreM. However, the PreM still follows a model transformation approach. Each instance element of the navigational model satisfies a XML schema that is generated from the metainformation. The XML data of each navigational instance is called Nij (instance i of model element j) and satisfies the schema XSDj. For each element in the model, a graphic designer creates a sample instance in HTML. The HTMLhi satisfies the schema XSDhi and is a sample instance of the model element i. The XSDj is extracted automatically using a XSD inferrer (Mono-Project). Finally, a semiautomatic XSLT transformation, using XSMapper (Llavador, 2006), is created between the XSDi and the XSDhi. The XSLT transformation is built from the Xpaths of the elements and attributes of the schemas of the navigational model instances.

The code generation is done using also a metamodelling approach. The models are consulted using OCL queries, and the results of the queries are used to fill the code generation template patterns.

4.4 Implementation of the method

The MDHDM support tool has been developed in Java, and is currently being improved. A beta version can be provided upon request. The MDHDM editor uses the Eclipse Modeling Framework (EMF) implementation of MOF. EMF provides the Ecore Metamodel, which is a subset of the MOF metamodel. The metamodels of the process, conceptual, and navigational models have been represented using Ecore metamodel. EMF generates the Java classes to create models reflectively.

The editor is composed of a set of graphical interfaces to draw process and conceptual models. The MDHDM editor allows the user to build model instances using the EMF's reflective Application Programmer Interface (API). The Models drawn are the input for the ATL transformations. The transformations are performed by the ATL transformation engine. The ATL engine works as an Eclipse plug-in, but it can also be embedded in Java applications, therefore, the ATL engine is embedded in the MDHDM editor. The source code generation can be done using the EMF template engine JET (Java Emitter Templates) or using ATL queries.

5. CONCLUSION

We have described how MDE can be applied to hypermedia development in order to obtain a full model-driven hypermedia development method (MDHDM). The PIM and PSM models needed for hypermedia development have been identified. As an original contribution, we have introduced the Persistence and the Concrete Hypermedia PSMs. The CHM describes the behavior of the Navigational Model elements from a dynamic point of view. It shows how they interact with the Persistence and Presentation Models. We have provided a transformation from a Process Model to a Navigational Model.

The transformations among the models are formalized in ATL, which helps to automate the process and guide software engineers in accomplishing their tasks. The main difference with other methods is that the transformations are defined declaratively and follow a complete MDE paradigm. The entire process is driven by the models and transformations that are defined at the metamodel level. A tool supporting the method is

also available. With MDE, we can automate the generation of hypermedia applications and facilitate the programming of development tools.
We are currently completing the catalogue of transformations and improving the implementation of a support tool. We also plan to automate the transformation of BPMN to WSBPEL (Web Services Business Process Execution Language) (OASIS, 2006) in order to support SOA (Service Oriented Architecture) development.

ACKNOWLEDGEMENT

This work has been supported by the CICYT España under grant DYNAMICA (TIC 2003-07804-C05-01) and the CONACYT México under grant 178867/193415.

REFERENCES

Abiteboul, S. et al, 2000. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, San Francisco, USA.

Atlas Group. 2005. Atlas User Manual. http://www.sciences.univ-nantes.fr/lina/atl/.

Ceri, S. et al, 2000. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks*, Vol. 33 (1-6), pp 137-157.

EMF, Eclipse Modelling Framework. http://www.elipse.org/emf/.

Garzotto, F., et al., 1993. HDM- A Model Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, Vol 11, No.1, pp 1-26.

Isakowitz, T., Stohr, E. A. and Balasubramanian, P., 1995. RMM: A Methodology for Structured Hypermedia Design. *Communications of ACM*, Vol. 38, No. 8, pp 34-44.

Kent, S., 2002. Model-Driven Engineering. *Proceedings of IFM2002*, LNCS 2335, Springer, 2002.

Kleppe, A. et al, 2003. *MDA Explained*. Addison Wesley. USA.

Koch, N., et al, 2000. Extending UML to Model Navigation and Presentation. *Modeling Web Applications, Workshop in UML2000*.

Mellor, S.J., et al., 2004. *MDA Distilled*. Addison Wesley. USA.

Mono-Project. http://www.mono-project.com/XML_Schema_Inference.

Llavador, M. and Canós, J.H., 2006. XSMapper: a Service-oriented Utility for XML Schema Transformation.

ERCIM News, Num. 64, January.

OASIS, 2006, Web Services Business Process Execution Language. http://www.oasis-open.org/

OMG, 2003. MDA Guide, http://www.omg.org/docs/omg/03-06-01.pdf

OMG, 2004. Meta Object Facility 2.0. http://www.omg.org/docs/ptc/04-10-15.pdf

OMG, 2005. MOF 2.0 Query/Views/Transformations. http://www.omg.org/docs/ptc/05-08-01.pdf

OMG, 2006. Business Process Managament Notation (BPMN). http://www.bpmn.org/

Schwabe, D. et al, 1996. Systematic Hypermedia Design with OOHDM. *Proceedings of International Conference on Hypertext*. Washington DC, USA, pp. 116-128.

Schmid, H.A. and Donnerhak, O., 2005. OOHDMDA-An MDA Approach for OOHDM. *Proceedings of International Conference of Web Engineering*.

Rumbaugh, J., et al., 1991. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ, USA.