Methodology Support for the Model Driven Architecture

Fatemeh Chitforoush, Maryam Yazdandoost, Raman Ramsin
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[chitforoush,yazdandoost]@ce.sharif.edu, ramsin@sharif.edu

Abstract

Model-driven approaches to software engineering have expanded their influence in recent years, with Object Management Group's Model-Driven Architecture (MDA) being the major force behind this boost. However, despite its merits, MDA remains insufficient for software system development, in the sense that it does not provide a concrete and comprehensive process for governing software development activities. There is therefore a strongly felt need for new model-driven software development methodologies. In this paper we review a number of existing model-driven methodologies, and propose a general framework for Model-Driven Development (MDD) based on MDA. The framework can be used for assessing and comparing methodologies, engineering new methodologies, and adapting existing ones so that they meet the special requirements of the model-driven approach. We have used the framework herein to show how agile methodologies fare in this model-driven development context.

1. Introduction

New technologies and platforms are emerging in software development on a continuous basis, resulting in a higher degree of effort needed for making use of the new capabilities that they offer, as well as satisfying the constraints that they impose. This situation has resulted in various problems as to portability, integration and interoperability. OMG's Model-Driven Architecture (MDA) initiative [15] aims at a global approach to software development that addresses these problems. In this approach, the bulk of the development effort will be dedicated to modeling the business concerns through elaborating a specification for the system which abstracts away from technical details, resulting in a so-called Platform-Independent Model (PIM). The transformation of a PIM into a Platform-Specific Model (PSM) is then achieved through introducing into the PIM the technical considerations depending on the chosen platform.

As a matter of necessity, MDA has had to remain general and abstract. As a consequence, MDA does not prescribe any specific development process for enacting model transformations in the context of a software development effort; that is, MDA offers no guidance as to the process (phases, activities and roles) to be used. Furthermore, MDA technologies and standards are not explicitly related to any activities within existing software development processes, since these technologies are being developed to be generally applicable in combination with all development processes. Since MDA does not prescribe a specific development methodology, each MDA-based development project has to define its own process, or select a process from the extremely sparse set of MDA-based methodologies available.

In this paper, we review a number of existing methodologies for MDA-based development of systems. Some of these methodologies incorporate precise processes, some just introduce an approach for system development loosely based on MDA concepts of modeling, while others strictly adhere and make extensive use of MDA standards. Based on this review, an MDA-based software development methodology framework is proposed, which can be used

a. as a yardstick for assessing and comparing methodologies,
b. as a generic MDA-based development process, based on which new methodologies can be engineered, and
c. as a template, into which existing methodologies can be fused, and thereby acquire MDA-based model-driven capabilities.

We have used the framework to show how agile methodologies fare in this model-driven development context. Agile methodologies [23] have been chosen for this purpose because of their widespread use, and also to challenge the common perception that agile methodologies cannot contend well in this context because of their inherent model-phobic nature.

This paper is organized as follows: Section 2 provides an overview of the Model Driven Architecture. Section 3 contains a brief survey of five prominent MDA-based methodologies. In section 4 we present the proposed MDA-based methodology framework, and show how an existing (agile) methodology can be augmented and adapted into an MDA-based methodology through fusion

1530-1362/07 $25.00 © 2007 IEEE
DOI 10.1109/ASPEC.2007.58

into the proposed framework. A criteria-based analysis of the methodologies surveyed and our proposed framework has been presented in section 5. Conclusions are reported in section 6, along with opportunities for furthering this work.

2. Model Driven Architecture (MDA)

The MDA approach proposed by OMG [15] views system development as a sequence of model transformations and refinements. Through these sequential steps, abstract models are gradually transformed into more concrete ones by adding technical details until an executable system is ultimately produced. OMG has put forward the following as the elements and principles governing MDA:

- **Model**: Models are an important means for specifying large-scale solutions, and must be expressed by means of well-defined notations. There are four types of models introduced in MDA, namely: 1) the Computational Independent Model (CIM), which as the problem domain (business) model, is independent from the use of the system as a computer system, and excludes any implementation details; 2) the Platform Independent Model (PIM) which describes the system from various perspectives regardless of its operating platform; 3) the Platform Specific Model (PSM) which provides a platform-dependent description of the same system specified by the PIM, and is constructed by transforming the PIM according to a Platform Model through adding details which are dependent on the operating platform; and 4) the Implementation Specific Model (ISM), which specifies all implementation details.
- **Model Transformation**: System development is implemented through a series of sequential transformations between models of various predefined types.
- **Meta-model**: Models are themselves expressed by meta-models, which enable meaningful integration and transformation between models, specifically via tools. MDA is based on a four layer meta-modeling architecture: 1) Meta-meta-modeling layer, including the Meta-Object Facility (MOF) [18], which defines an abstract language for specifying meta-models, 2) Meta-model layer which consists of meta-models defined in MOF (e.g. UML meta-model), 3) Model layer including models of the real world, and 4) Real world layer which includes things from the real world.

In the MDA development approach, modeling and model transformations are the main activities during system development. Models and model transformation specifications can be reused extensively across different solutions, organizations, and domains, as well as any specific platform. MDA tools may support automatic or semi-automatic transformations from model to model, and ultimately to the executable code.

OMG also provides a number of supporting standards for MDA including the Unified Modeling Language (UML) [19], the Meta-Object Facility (MOF) [18] as a modeling language for model definition, and the XML Metadata Interchange (XMI) [20] which facilitates automatic generation of an XML-based document for a model according to its MOF definition.

3. MDA-Based Methodologies

The following sections contain a brief overview of a number of prominent MDA-based software development methodologies.

3.1. MODA-TEL

MODA-TEL is proposed as a software development process based on MDA principles and concepts [8,9,16]. It is specialized for distributed applications, but is general enough to be applicable to other domains and situations as well. The MODA-TEL process is defined in accordance with OMG's Software Process Engineering Meta-model (SPEM) [17].

MODA-TEL separates preparation activities from execution activities in distinct phases. The following phases are identified by this methodology:

1. **Project Management**: During this phase the software development process is selected and described in terms of its activities; identified activities are then allocated to roles, and procedures for quality assurance are put into place.
2. **Preliminary Preparation**: The objective of this phase is to identify modeling and transformation needs. The final execution platform of the system is also identified and expressed as an *abstract platform*. The notion of the abstract platform is discussed in detail in [1]. The appropriate modeling language for the project and its specific needs are also identified. Required transformations between models are specified based on the selected modeling languages. The traceability support strategy is implemented via *traces*, which are used to track requirements and changes.
3. **Detailed Preparation**: Models and model transformations are specified in this phase. The specifications of modeling languages and model transformations are prepared according to the needs identified in the previous phase.
4. **Infrastructure Setup**: In this phase, tools are selected for supporting MDA-based development activities. For example, tools for automatic/semi-automatic code generation may be selected. The metadata management facility is also defined in this phase.

5. **Project Execution**: This phase spans the main software development activities. Activities of this phase depend on the software development process selected in the project management phase. The methodology proposed in [8, 9, 16] specifies seven activities in this phase: *requirements analysis*, *modeling*, *verification and validation*, *transformation*, *coding and testing*, *integration and deployment*, and *operation and maintenance*.

Phases of this methodology are performed in an iterative-incremental fashion.

3.2. MASTER

MASTER is a European IST project during which a MDA-based methodology with the same name has been proposed [14]. The process consists of eight main phases, each of which includes various activities. The phases are:

- **Capture User Requirements:** The objective of this phase is to elicit and document customer requirements. Products which are produced during this phase are: an Application Model in which customer requirements are formalized, an initial Application PIM, and an initial functional requirements specification.
- **PIM Context Definition:** The system goals as well as the scope of the system are defined in this phase. Other activities of this phase are: identifying the external actors of the system, specifying the main services offered by the system, and identifying the business objects exchanged between actors and the system.
- **PIM Requirements Specification:** The main activity in this phase is refining the PIM Context produced in previous phase, as well as specifying use cases, and identifying non-functional requirements as well as modeling their relationship with functional requirements.
- **PIM Analysis:** In this phase, system functionalities and QoS aspects are described with a view to the interior of the system. Traceability to the Requirements PIM is also verified in this phase.
- **Design:** In this phase, a platform-independent design is first performed for all the requirements. The design is then refined in order to denote the platform-specific solution.
- **Coding and Integration:** According to the ideal MDA approach, the code is to be produced automatically from the PSM through transformation engines.
- **Testing:** Test cases are to be generated automatically from the test model (which is a refinement of the PIM) through transformation engines.
- **Deployment:** In this phase, the developed system is delivered to the customer.

3.3. MIDAS

MIDAS is a model driven methodology framework for agile development of Web Information Systems (WIS) [4, 5]. UML is used for representing the different PIMs and PSMs which are proposed in this framework. It also defines mapping rules for transforming the models. Instead of introducing a specific process, MIDAS focuses on three dimensions of modeling a WIS: 1) *levels,* which refer to the hypertext content and the presentation levels, 2) *phases*, which refer to the phases of the software lifecycle, and 3) *aspects,* which refer to the structural and behavioral modeling viewpoints. Platform Independent Models, Platform Specific Models, and Computation Independent Models as well as the relevant mapping rules are defined according to these dimensions.

3.4. C^3

C^3 is a software development process which is embedded into a concurrent, collaborative and component-based methodology enriched with MDD techniques [12]. The main feature of the C^3 architecture is its domain repository which contains domain-specific metadata and components. Its process consists of two main phases:

- **Standardization Phase:** In this phase, domain software assets which are archived in a repository are accessed and downloaded onto the *project repository*, which is specific to the project in hand. Component models and domain-specific model elements can also be uploaded for future reuse.
- **Software Development Phase:** This phase includes three main steps which are common in most methodologies, namely: 1) *Model Design*, in which component developers select a business application architecture from the project repository to work on. Consistency checks against the architecture are also performed for each and every component, 2) *Code Generation***,** which is performed after modeling the business application with UML or XMI on a platform-independent level. Code generation tools transform the models into platform-specific software components, and 3) *Application Deployment*, in which components are deployed into the user environment based on the architectural framework designed.

3.5. ODAC

ODAC is a methodology based on the Reference Model of Open Distributed Processing (RM-ODP) [13] with the potential to be a MDA-oriented methodology [10]. The ODAC process consists of three main phases: 1) *Analysis*, in which the *Behavioral Specification* – a PIM

describing the system according to its objective and its role in the business – is produced; 2) *Design*, in which the *Engineering Specification* – a PDM that is the description of the execution environment – is produced; and 3) *Implementation*, in which the *Operational Specification* – a PSM that is the result of the transformation of the PIM as configured according to the PDM – is produced.

ODAC prescribes steps for each phase as well as a number of guidelines for producing the relevant artifacts.

4. Proposed Methodology Framework

Since MDA has its roots in object-orientation and component-based development, it can be merged with many existing software development processes. It has been conceived to allow existing development processes in organizations and projects to be reused. MDA's general, abstract and flexible nature facilitates integration with existing methodologies. Based on the insight gained through our survey of MDA-based methodologies, we propose a methodology framework that can be used for augmenting established software development processes with MDA concepts, principles and technologies. The framework can also be used for engineering new methodologies and comparing existing ones.

The proposed methodology framework is divided into four main phases, each of which consists of a number of stages and their constituent activities. The following sections contain descriptions of these phases, as well as the project-wide umbrella activities complementing them.

4.1. Project Initiation Phase

The project's objectives are identified, and its size and scope are estimated. All constraints and risks involved in the project are explored and important people, organizations, and external systems which interact with the system are identified. Team members are also assigned at this stage; typical roles involved include domain experts, MDA experts, methodology engineers and architects.

4.2. Software Development Process Analysis and Selection Phase

The requirements of the software development process needed for the specific project and domain at hand will be analyzed in this phase. The requirements specified are used as a basis for selecting or engineering a software development process (SDP) to be used in the SDP Execution phase. If an existing SDP is selected, it typically requires adaptation to satisfy the requirements; all the necessary modifications are applied in this phase and the next, and will result in the production of a precise description for the process in terms of the activities that should be followed. Producers typically involved in this phase are methodology experts, MDA experts and domain experts.

4.3. MDA Support Phase

This phase can be performed in parallel with the previous phase. This is where decisions about model-driven development of the system and its high-level architecture are made. Roles for performing these activities include domain experts, system architects, and MDA experts. Activities performed in this phase are as follows:

- **Platform Identification and Specification:** The target platform upon which the system will be implemented and ultimately deployed is identified by system architects. The notion of platform here includes all the hardware, software and technological aspects of the target system environment. After the identification of the target platform – which is a major architectural decision made by system architects – it should be specified concretely as a Platform Model (PM). The PM developed here will be used when applying model transformations (PIM to PSM). Architects may also decide to reuse a platform model from a previous project, or from a repository of platform models.
- **Modeling Language Identification and Specification:** Different perspectives of the system to be modeled are identified, and a set of models is selected accordingly. This activity typically involves domain experts and MDA experts. As a typical example, experts may decide to model the system's functionality, behavior and structure in use case models, statecharts, and class diagrams respectively. In accordance with these models, a modeling language that is expressive enough for the domain and the project situation will be selected. This language will be used to model the PIMs and PSMs of the different perspectives of the system. The experts may decide to modify or extend the metamodel of the selected modeling language in accordance with the specific needs of the project.
- **Transformations Identification and Specification:** Possible or necessary transformations between models are identified; the main focus is on transformations from PIMs to PSMs, but if necessary, transformations between different PIMs or different PSMs are also included. Transformations have to take into account the particulars of the modeling languages selected in the previous activity. Transformation identification can in turn influence the modeling language identification and specification activity. After deciding on what transformations are needed, they must be specified in detail, with transformation rules and annotations properly described. Transformation specifications may

also be selected from a standard repository, or defined based on previous experience on similar domains or project situations; UML-to-J2EE transformations, for instance, have become pretty standard, and can be used to great effect where relevant.

- **Tool Selection**: Tools play an important role in MDA-based development. A number of activities have to be handled by tools, such as the specification of modeling languages and models themselves, model transformation and model-based code generation, and the definition of transformation rules and annotations. In this activity, the necessary tools will be selected to be used in the next phase.

In model-driven development, modeling languages, platform models and transformation specifications are all elements of reuse. Therefore, activities of this phase – such as selecting a platform or deciding on the usage of modeling languages and transformation specifications – are key factors in enabling effective reuse in the next phase.

4.4. SDP Execution Phase

This is the main phase of the project, where all the final products of the project, including the software system, are produced. The specific activities of this phase depend on the methodology selected/engineered in the Software Development Methodology Analysis and Selection phase. As a result of the model-driven nature of this framework, models are the major products of the development process and are refined in iterative and incremental cycles, from PIMs to code. Since models are supposed to drive the whole execution phase, in case failures, defects or other problems are discovered in any of the activities, the process should facilitate the resolution of the issue at the modeling level. All activities in the SDP execution phase can generate feedback used for refining and improving the development process, and can thereby influence the results of previous phases. It is also possible to send feedback to the MDA support phase; in case any changes have to be made to the modeling language or the PM.

This phase is where existing methodologies can be fused into the framework, and thereby be enriched with MDA-based process components. In order to examine how agile methodologies fare in this context, we have reviewed seven agile methodologies: namely DSDM [7], Scrum [22], XP [2], ASD [11], dX [3], Crystal Clear [6], and FDD [21]. In trying to integrate these methodologies into our proposed framework, we have observed that agile methodologies incorporating a modeling process fit well in this framework. Crystal Clear, FDD, and ASD are examples of agile methodologies that can be integrated into this framework with some relatively minor modifications to their processes. On the other hand, methodologies that lack or ignore modeling activities cannot be easily fitted into the framework in their original forms; a possible solution is to first extend and enhance such methodologies through applying complementary methods that add modeling to agile methodologies; Agile Modeling (AM) is a prominent example of such methods [24].

As an example of how the integration can be done, we have fused the ASD process into the SDP Execution Phase of our proposed framework. Most of the modifications on the process will be made to the iterative development engine; i.e. the Component Development stage of the ASD process is modified and made compatible with the MDA approach. Consequently, all activities in the modified stage are based on modeling and model transformations. The resulting ASD process is as described below:

- **SDP Execution Initiation**: In this activity, all the team members prescribed in ASD and not already selected during the project initiation phase are recruited and organized in teams. High-level requirements of the system are identified, as well as an overall architecture. Important technological decisions have already been made in previous phases and the target platform has been selected. At this stage, architectural details are specified so that the actual development may begin. The architecture thus elaborated is modeled as PIMs along with the high level requirements. These PIMs will be iteratively completed and refined in upcoming development cycles.
- **Component Identification and Overall Planning:** Crucial product components are specified and assigned to development cycles according to the risks involved in their implementation, with consideration given to their interdependencies. An initial development plan is then prepared accordingly.
- **Iterative Development Cycles**:
  - **Cycle Planning**: In this activity, the development plan will be revised and recalibrated according to the experience gained in previous cycles. New components may also be introduced and planned for development in later cycles. Tasks of the current cycle are also determined and assigned to team members.
  - **Component Development**: In this stage, the component(s) assigned to the current cycle are designed and implemented via gradual model refinement. This stage contains the following activities:
    - **Modeling:** The PIMs of the component to be implemented are produced according to models selected in previous stages. Platform-independent models of the existing (legacy) systems may also be developed herein by applying reverse engineering.

- **Verification:** This activity is mainly concerned with performing correctness and consistency checks on the models.
- **Model Transformation:** Models produced for the component at hand will be refined to an equivalent PSM by means of the pre-specified transformation specifications based on the PM. This stage may influence the *Modeling Language Identification and Specification* or *Transformations Identification and Specification* phases through modifying the PM or the transformation rules. The resulting PSM is then refined into code. Automatic/semi-automatic code generation tools are used extensively.
- **Coding/Testing:** The code which cannot or should not be produced by code generators is produced and added manually by the developers. Testing is then performed on the produced component.
- **Integration and Deployment:** The produced builds are fed into an integration process which may also deploy the system into the user environment.

○ **Process and Quality Review**: In this activity, group reviews of the components produced are held where the problems confronted are discussed and resolved. The process itself is also reviewed and adjusted, possibly affecting the *Software Development Process Analysis and Selection* phase.

- **Final Release**: The final release of the software into the user environment will be done in this activity. All other support deliverables of the system, e.g. documents and manuals, are also produced and delivered to the customer.

4.5. Umbrella Activities

There is a need for a number of monitoring and management activities to be applied in parallel with all the above phases. These include project management, quality control, risk management and training. The most important activity in this category is *reuse management*, which is one of the most important issues in model driven development of software systems. Since MDA is based on extensive reuse of models, modeling languages, and model transformation specifications, the careful management of reuse is critical to the success of MDA-based projects.

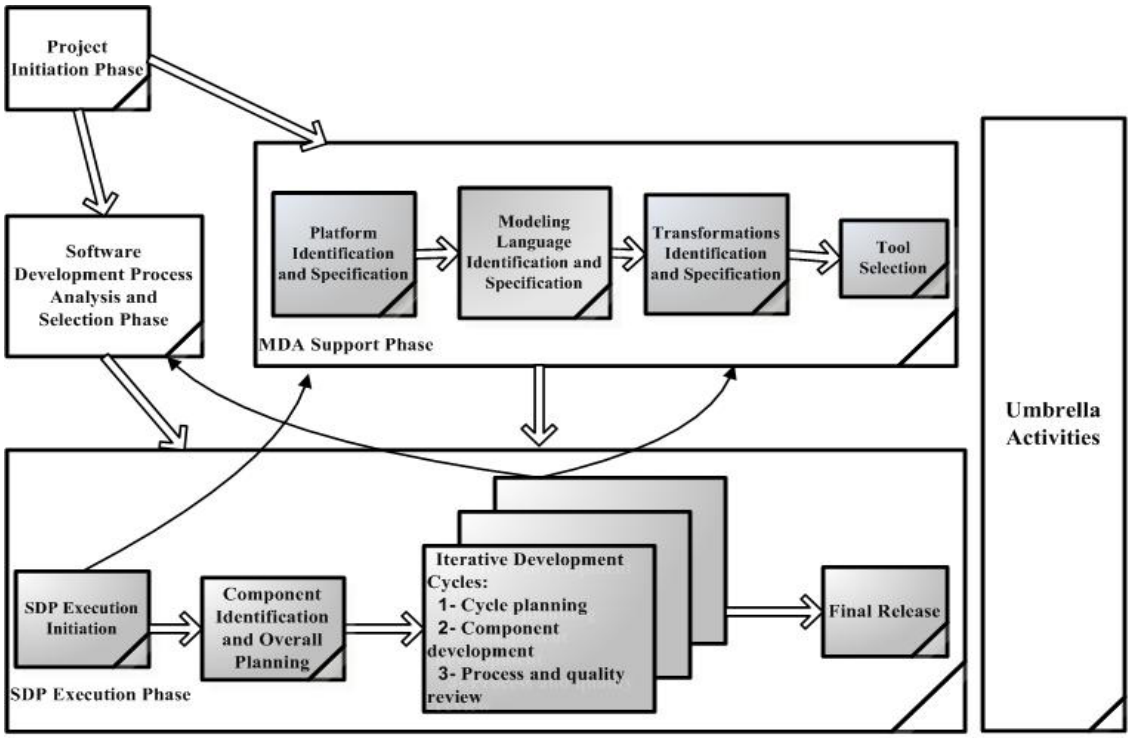

Figure 1. Proposed Methodology Framework: ASD has been fused into the framework as an example

5. Criteria-Based Analysis

In order to gain a better understanding as to the merits of the methodology framework proposed herein, we have conducted a criteria-based analysis of the reviewed methodologies and our proposed framework, the results of which are summarized in Table 1.

The analysis has been made according to the following criteria:

- **MDA Support**: The extent to which MDA concepts and practices are supported by the methodology; all the methodologies reviewed herein claim to be MDA-based, but they use different MDA concepts, principles, and techniques. For example, C^3 uses MDA standards (UML and XMI) as well as model transformation in its proposed process, but does not incorporate model types, i.e. PIM, PSM, and PM.
- **Process Inclusion**: Whether the methodology incorporates a specific and concrete process or not; many methodologies claiming to be MDA-based do not prescribe a concrete process, claiming that MDA satisfies their process needs.
- **Coverage of the Generic Lifecycle**: The extent to which a methodology covers the phases and activities of the generic Software Development Life-Cycle (SDLC).
- **Process Precision**: Whether the process is detailed and precise enough to be followed effectively and efficiently by the people involved in the development; some MDA-based methodologies do incorporate a process, but it is not precise enough. For example, MIDAS and MASTER do not propose exact activities in their proposed processes.
- **Application Scope**: The scope of projects and domains to which the methodology is applicable; some proposed methodologies are not general-purpose enough to be applicable to all projects. MIDAS, for instance, is targeted at web information systems, while MODA-TEL is mainly focused on distributed applications.

6. Conclusions and Future Work

While much has been invested in MDA, methodology support for model-driven development has been largely overlooked. There are very few MDA-based software development methodologies available, and those with precise processes are even fewer. In this paper, we have surveyed a number of prominent MDA-based methodologies with special attention to their processes. We have also proposed a MDA-based methodology framework to be used for engineering MDA-based software development methodologies. Existing methodologies can be fused into this framework, and thereby augmented and adapted into MDA-based methodologies. This will allow organizations to continue using their current methodologies, augmenting them with MDA-based capabilities when necessary.

This work can be extended through further refinement and elaboration of the framework, focusing on the stages and their constituent activities, as well as the roles involved. Further investigation can also be directed at exploring the types of methodologies that lend themselves better to fusion into the proposed framework. It would also be worthwhile to devise concrete procedures for engineering and/or adapting software development methodologies based on the proposed framework.

7. Acknowledgments

We wish to thank the Research Vice-Presidency of Sharif University of Technology for sponsoring this research.

Table 1. Analysis Results

	MDA Support	Process Inclusion	Coverage of the Generic Lifecycle	Process Precision	Application Scope
MODA-TEL	Full	Yes	Yes	High	Yes
MASTER	Full	Yes	Yes	Low	Yes
MIDAS	Full	Partial	No	Very low	No (Web Information Systems)
C3	Partial (Model Transformation, MDA standards)	Partial	Partial	Low	No (Business application software and domain-specific software assets)
ODAC	Full	Yes	Partial	Low	Yes
Our Proposed Framework	Full	Yes	Yes	High	Yes

8. References

[1] J.P.A. Almeida, *Model Driven Design of Distributed Applications*, CTIT Ph.D.-Thesis Series, No. 06-85, Telematica Instituut Fundamental Research Series, No. 018, 2006.

[2] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2004.

[3] G. Booch, R.C. Martin, and J. Newkirk, *Object Oriented Analysis and Design with Applications*, 2nd ed. (Unpublished). Addison Wesley, 1998. Available on the Web at: http://www.objectmentor.com/resources/articles/RUPvsXP.pdf.

[4] P. Cáceres, E. Marcos, and B. Vela, "A MDA-Based Approach for Web Information System Development", Workshop in Software Model Engineering (WiSME), 2003.

[5] P. Cáceres, E. Marcos, and V. Castro, "Integrating Agile and Model-Driven Practices in a Methodological Framework for the Web Information Systems Development", International Conference on Enterprise Information Systems (ICEIS), 2004, 523-526.

[6] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*, Addison-Wesley, 2004.

[7] DSDM Consortium, *DSDM: Business Focused Development*, 2ndEd., J. Stapleton (Editor), Addison-Wesley, 2003.

[8] A. Gavras, M. Belaunde, L. Ferreira Pires, and J.P.A. Almeida, "Towards an MDA-based development methodology for distributed applications*"*, In Proceedings of the 1st European Workshop on Model-Driven Architecture with Emphasis on Industrial Applications (MDAIA), University of Twente, Enschede, The Netherlands, March 2004, pp. 43–51.

[9] A. Gavras, M. Belaunde, L. Ferreira Pires, and J.P.A. Almeida, "Towards an MDA-Based Development Methodology", *EWSA*, 2004.

[10] M.P. Gervais, "Towards an MDA-Oriented Methodology", Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02), IEEE (Ed), Oxford, England, August 2002, pp. 265-270.

[11] J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House, 2000.

[12] T. Hildenbrand, and A. Korthaus, "A Model-Driven Approach to Business Software Engineering", Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI), Volume IV Information Systems, Technologies and Applications: I, IIIS, Orlando, Florida, USA, July 18-21, 2004.

[13] ISO, IS 10746-x, ODP Reference Model Part x, 1995.

[14] X. Larrucea, A.B.G. Diez, and J.X. Mansell, *Practical Model Driven Development Process*, Technical Report, NUMB 17, University of Kent, 2004, pp. 99-108.

[15] J. Miller and J. Mukerji, *MDA Guide Version 1.0.1,* OMG Document, June 2003, see: http://www.omg.org/docs/omg/03-06-01.pdf.

[16] MODA-TEL project. Deliverable D3.2, *Guidelines for the application of MDA and the technologies covered by it*, 2003. see: http://www.modatel.org/public/deliverables/D3.2.htm.

[17] Object Management Group, *Software Process Engineering Metamodel Version 1.1 (SPEM),* see: http://www.omg.org/cgi-bin/doc?formal/2005-01-06.

[18] Object Management Group, *Meta Object Facility Version 2.0*, see: http://www.omg.org/cgi-bin/doc?formal/2006-01-01.

[19] Object Management Group, *Unified Modeling Language Specification Version 2.1.1*, see: http://www.omg.org/cgi-bin/doc?formal/07-02-05.

[20] Object Management Group, *XML Metadata Interchange Specification Version 2.1*, see: http://www.omg.org/cgi-bin/doc?formal/2005-09-01.

[21] S.R. Palmer, and J.M. Felsing, *A Practical Guide to Feature-Driven Development*, Prentice-Hall, 2002.

[22] K. Schwaber, M. Beedle, *Agile Software Development with Scrum*, Prentice-Hall, 2001.

[23] J. Highsmith, *Agile Software Development Ecosystems*. Addison-Wesley, 2002.

[24] S. W. Ambler, Agile Modeling: Effective practices for eXtreme Programming and the Unified Process, Wiley, 2002.