An Agile Method for Model-Driven Requirements Engineering

Grzegorz Loniewski, Ausias Armesto, Emilio Insfran
ISSI Research Group, Department of Computer Science and Computation
Universidad Politecnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
{gloniewski, einsfran}@dsic.upv.es, aarmesto@novasoft.es

Abstract - **The complexity and pervasiveness of software applications has increased over the last few years. In this context, software development processes have also become complex and difficult to use. It is widely recognized that requirements engineering has become a critical activity within this process. In this paper, we aim to provide a methodological approach which focuses on requirements engineering within the Model-Driven Development (MDD) context. Our approach is an OpenUP extension in which the requirements discipline is placed in the model-driven context. We believe that the integration of requirements engineering and MDD into one consistent process will provide practitioners with the benefits of both. This paper presents the definition of the proposed process, OpenUP/MDRE, including its activities, roles, and work products. We also provide an example of its use in a SOA-based software development project. The use of our approximation guides the activities of requirements engineering and promotes automation by means of model transformations.**

Keywords - Model-Driven Development, Requirements Engineering, agile methodology, OpenUP.

I. INTRODUCTION

Software systems are becoming more and more complex, and the success of their development should not depend on individual efforts and heroics. Successful software development can only be accomplished by using a well-defined software development process. Requirements Engineering (RE) is recognized as being one of the most critical aspects of this process. Errors made at this stage may have negative effects on subsequent development steps, and on the quality of the resulting software.

Several software development approaches with which to support the development of complex systems have been proposed, of which Model-Driven Development (MDD) is one of the most promising. MDD promotes the separation of concerns between the business specifications and the implementation of these specifications on specific platforms [4]. This separation is obtained by using models that allow the level of abstraction to be elevated [9]. In this context, Model-Driven Architecture (MDA) [13] is the best known realization of the MDD. It encourages the use of models and model transformations, among several models: the Computation Independent Model (CIM), the Platform Independent Model (PIM), the Platform Specific Model (PSM) and code.

However, most MDA-based approaches focus on the transformation strategies from PIM to PSM and from PSM to code. Unfortunately, less attention is paid to the CIM to PIM transformations upon which requirements engineering places emphasis. Loniewski *et al.* [7] have shown that there is no systematized development process that applies RE techniques in the MDD context. Although various techniques for CIM to PIM model transformations exist, those software development projects which attempt to use them often fail owing to the lack of well-defined methods and processes describing the entire development life cycle.

Another problematic issue as regards existing MDD supporting approaches is that the clear assignation of the methodology's artifacts to the MDA abstraction levels is frequently impossible. This situation arises as a result of the unclear definition of CIM and PIM, which confuses developers. It is consequently very difficult to apply the MDA life cycle by starting from the CIM level, and thus obtain most of the benefits that the MDD process should provide, i.e., automation in model transformations and traceability management.

In this paper, we introduce a methodology that provides MDD processes with an agile method which incorporates the RE activities. This method has been developed as an extension of OpenUP [2], an agile methodology, which is a minimally sufficient software development process that provides only fundamental content for small or medium size projects that deliver software as a main product. It mainly focuses on the collaborative nature of software development. The main extension is the replacement of the Requirements discipline with Model-Driven Requirements with which to elicit, model and manage requirements in the MDD context. Although OpenUP was not initially created to support MDD processes, it offers a flexible, agile and extensible means to introduce a model-driven process integrated with RE activities. This work may be an interesting contribution for those software process engineers who are faced with the challenge of guiding software development projects that follow an MDD approach from the requirements elicitation. Moreover, in projects already using the OpenUP method, the agility feature of our method makes the incorporation of the improved MDD-complaint OpenUP extension quite quick and smooth.

The remainder of this work is as follows. Section 2 provides an overview of the software process as an engineering process and also introduces some related approaches. Section 3 introduces the improved OpenUP-based methodological approach, presenting details of the content and process elements of the new Model-Driven Requirements discipline. Section 4 puts this approach into practice, discussing an example of its application to a SOA-based middleware platform development. Finally, Section 5 contains some conclusions and future work.

II. Research Background

This section describes the background of software process engineering along with other important approaches related to OpenUP/MDRE.

A. *Software Process Engineering*

When methodologies first emerged, each software development process used its own concepts and notations to define the contents of the methodology. The need to unify all these concepts and notations therefore emerged. The OMG thus introduced the Software Process Engineering Metamodel (SPEM) [12] standard. SPEM provides a complete metamodel based on the Meta Object Facility (MOF) [11] to formally express and maintain development method content and processes.

Various tools supporting this standard currently exist, one of which is the Eclipse Process Framework (EPF) [1]. EPF is a comprehensive process authoring tool which provides extensive method authoring and publishing capabilities. EPF uses the concept of a plug-in library to allow process engineers to define and extend methodologies. The fact that OpenUP is itself a plug-in library permits it to define new processes or extend existing ones. In this paper, this tool is used to extend OpenUP by incorporating a model-driven requirements engineering approach.

B. *Related Work*

Various existing approaches provide model-based requirements specifications incorporated into an agile methodology. The Agile Unified Process (AUP) is a simplified version of the RUP which applies agile techniques to Agile Model-Driven Development (AMDD). This approach considers the model as the principal artifact of the requirements specifications. However, its use in the model-driven context is not clear. There has been another attempt to create a lightweight methodology upon the MDA principles: OpenUP/MDD. However, its stable version has not been released. This proposal was focused solely on the transformations from the PIM to PSM level of the MDA framework. In this respect, our proposal and the OpenUP/MDD approach are complementary. The methodological approach presented in this work focuses on the CIM level transformations and generates the desired model at the PIM level.

Several attempts to establish a methodology with model-driven principles can be found in literature, but none of them focuses on the CIM-level requirements and a complete process to derive PIM-level specifications. Methods and techniques that describe particular transformations of requirements also exist (e.g., [5] or [15]), but they hardly ever possess a well-defined description of their use in a development process.

III. OpenUP Extension for the Model-Driven RE

This section introduces an extension of OpenUP for model-driven requirements engineering - OpenUP/MDRE, signifying that it focuses on a discipline for requirements engineering based on models in the model-driven context. We believe that this new discipline will improve the effectiveness of requirements engineering and will also make a significant contribution towards supporting analysts and developers by providing a well-defined process.

The first proposal of the methodology presented in [8] was defined on a base method adapted from the Rational Unified Process (RUP) [6]. This proposal was validated in a case study in an academic context and some of its weaknesses were identified. The RUP-based model-driven requirements engineering proposal was too strict and complex. This limitation has been solved by changing the base process of our approach to the agile OpenUP. Both RUP and OpenUP provide an iterative and incremental life cycle. However, OpenUP decreases the ceremony of the process, incorporating one of the strong points of agile methodologies such as Extreme Programming and Scrum. Our new method restricts neither the requirements elicitation methods, nor their specification form. The tasks provided in our approach allow an easy adaptation of any type of requirements specifications when using them in a model-driven process leading to PIM definition.

OpenUP is also available under the Eclipse Public License and is developed as a plug-in library of the Eclipse Process Framework (EPF) [2] tool, giving process engineers a powerful mechanism with which to provide content variability of its process elements by means of contribution, extension and replacement.

Figure 1 illustrates the OpenUP hump chart in which the Requirements discipline is redefined by the Model-Driven Requirements discipline.

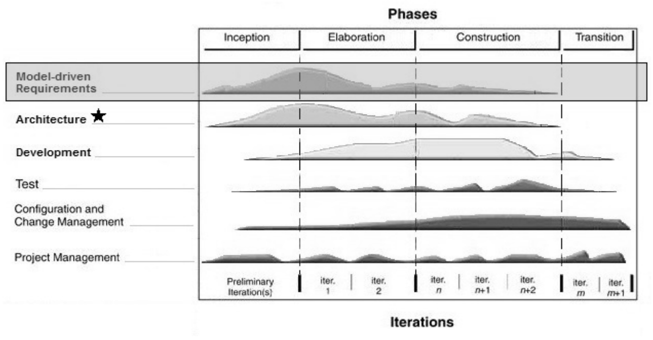

Figure 1. OpenUP Extension for Model-Driven Requirements

As is depicted in Figure 1, the redefined Model-Driven Requirements discipline is a concern from the inception phase to the construction. Since the hump chart emphasizes the workload within disciplines, the diagram shows that the new discipline is particularly important during the inception and elaboration phase, in which the product vision is created and the architecture is established.

In this new discipline, a CIM requirements model is first created on the basis of the stakeholders' needs, and this is then transformed into an analysis model at the PIM level. Specifying the CIM to PIM transformations reduces the system analysts' workload and responsibilities by including domain experts and stakeholders in the system modeling. The analyst's workload therefore decreases, particularly in the elaboration phase. Since we are concentrating on model use in the MDD context, the workload in the Development discipline in the elaboration phase also decreases, depending on the degree of automation of the specification generation tasks in the Model-Driven Requirements discipline.

The Architecture discipline (marked with a star) is only performed if the architecture (adequate architectural elements, models, or patterns) has not been defined. However, once this architecture has been defined, the Architecture discipline is optional and may be narrowed to refine the reference architecture provided.

Owing to space constraints, we shall comment only briefly on each activity of the main extensions of the OpenUP methodology, which is the Model-Driven Requirements discipline, pointing out the roles responsible for each task, along with input and output artifacts.

Our approach maintains the roles originally defined by OpenUP, but also introduces two roles related to the model-driven context activities: Model Analyst and Transformations Specifier.

A set of new activities has been provided and the workflow has been replaced. Figure 2 shows the Model-Driven Requirements workflow represented through a tailored version of a UML activity diagram. It is based on the OpenUP's Requirements workflow tasks, but also introduces new activities and tasks, which are crucial to the MDD process.

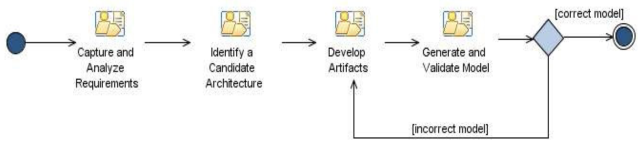
Capture and Analyze Requirements
Identify a Candidate Architecture
Develop Artifacts
Generate and Validate Model
[correct model]
[incorrect model]

Figure 2. Model-Driven Requirements workflow

The following subsections discuss each of the activities of the proposed workflow, including a description of the new activities, accompanied by a detailed diagram of tasks, their input and output artifacts, and responsible roles. These diagrams show not only the roles which are responsible for a particular task, but also the roles who participate in its realization.

A. *Capture and Analyze Requirements*

This activity, which is mainly composed of tasks taken from the original OpenUP requirements discipline, involves reaching an agreement on a statement of the problem to be solved, identifying the stakeholders and clearly defining the system's boundaries and constraints.

Stakeholders' needs and potential features, which represent the high-level user or customer view of the system, are captured and documented in the Vision document. The potential for possible misunderstandings between the Analyst and the other different domain background stakeholders is minimized by establishing and maintaining a common vocabulary in the Glossary.

The purpose of this activity is, amongst others, to identify and capture functional and non-functional requirements for the system. The idea is to initially understand and determine the requirements at a high-level, and then describe these requirements with enough detail to validate understanding of the requirements, to ensure concurrence with stakeholder expectations, and to permit software development to begin.

The artifacts that result from the tasks performed constitute the principal input for further modeling tasks. The tasks defined in this activity are shown in Figure 3.

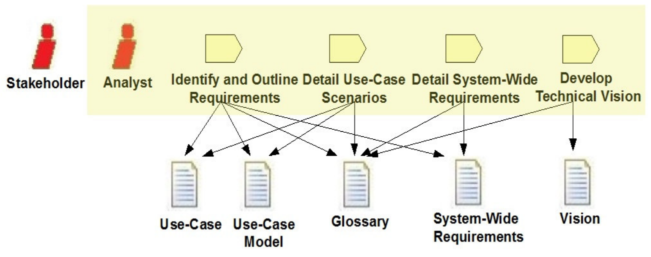

Figure 3. Roles, tasks and work products of the Capture and Analyze Requirements activity

B. *Identify a Candidate Architecture*

This activity is essential for the software development process. It determines the content of artifacts in the RE phase, which also conditions the MDD process to be followed. In this activity, the main architectural elements are identified and the metamodels for models at the CIM-level and PIM-level of the MDD process are established. This approach is architecture-centric, signifying that it is the architecture that demands a certain type of model to be created. The architectural pattern identified for the system becomes a basis from which to derive further analysis artifacts. For example, if the architecture chosen is Service Oriented Architecture (SOA), the model that describes requirements at the CIM-level may be given as Business Process Modeling Notation (BPMN) and it is supposed that the model at the PIM-level may be a service model. A detailed diagram of particular tasks, roles and artifacts for this activity is shown in Figure 4.

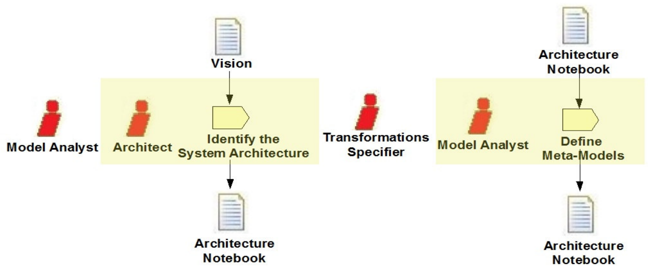

Figure 4. Roles, tasks and work products of the Identify a Candidate Architecture activity

C. *Develop Artifacts*

This activity, like the Identify a Candidate Architecture, is essential in our approach. The CIM-level requirements model (RM) is created in this activity, and this model conforms to the metamodel selected for this purpose in the previous activity. Model-driven transformations are also specified and planned. In particular, the transformation language is chosen, along with the transformation automation level and tool support that are specified. Transformation rules are described in a specially prepared Transformation Rules Catalog (TRC). This document is the principal artifact supporting transformation execution, but it is also essential for the requirements traceability, which is the means used to control changes in requirements, maintain agreements with the customer and set realistic expectations as to what will be delivered. This requirements traceability is performed in a new task, Manage Dependencies, and the Model Dependencies Specification document is produced as a result of this.

A Transformation Iteration Plan (TIP) is created in this activity to describe not only the elements of a source model to which the transformation applies, but also the order of the transformation rule application. For example, if the architecture chosen is SOA, the CIM model contains BPMNs and the PIM model contains service models. An example of the transformation iteration plan could specify that the transformation rules between a BPMN of a higher level and a BPMN of a lower level should be executed before the transformations rules from a BPMN of a lower level to a service model. The tasks defined within this activity are shown in Figure 5.

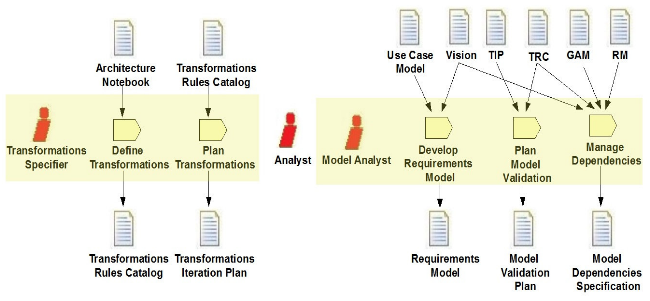

Figure 5. Roles, tasks and work products of the Develop Artifacts activity

D. *Generate and Validate Model*

This activity concludes the entire requirements modeling process by generating the principal artifact of the requirements engineering process, which is the Generated Analysis Model (GAM). GAM represents requirements specification at the PIM-level of the MDA lifecycle. For example, GAM can be specified by a UML sequence diagram in the case of a client-server software project, or a service model in the case of SOA platform development.

Artifacts, such as a requirements model (RM) or transformation rules catalog (TRC), developed as a result of the previously performed tasks, are the input artifacts for performing model transformations in order to create the GAM artifact. These transformations can be manual or automated, depending on their level of complexity. Their execution may be supported by appropriate tools. Although the GAM is systematically obtained by the transformation rules, we believe that it is necessary to validate it with regard to its consistency and correctness. This type of validation should be previously described while defining the transformation rules in a separate Model Validation Plan (MVP) document. The RM can also be validated against the specific conceptual standards of the domain in which it is applied. The validation result is stored in the Model Validation Record (MVR) document. The tasks defined in this activity are shown in Figure 6.

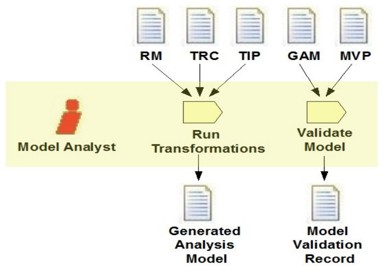

Figure 6. Roles, tasks and work products of the Generate and Validate Model activity

IV. APPLYING OPENUP/MDRE

The main objective of this section is to show the applicability and feasibility of the OpenUP/MDRE approach in the development process of a SOA-based system. In this example of methodology usage, the system specification is developed on the basis of user requirements, which were captured as user scenarios. We assume that the requirements scenarios and use cases defined in the Capture and Analyze Requirements activity have been correctly captured and documented in the initial stages of the project. These artifacts constitute the input for the subsequent model-driven process.

Each of the main tasks of the OpenUP/MDRE application is commented on in the following subsections.

A. *Identify the System Architecture*

Our proposal for the OpenUP extension was applied to a domain in which current systems have to deal with many complex processes, multiple stakeholder views and users, a distributed environment, changing requirements and many other factors. These factors and the domain's complexity lead to system specifications of the same complexity, and the principal issues to be considered are interoperability and distributed use of the software functionalities. These characteristics indicate that the Service-Oriented Architecture (SOA) is a primary candidate architecture. SOA strengthens such factors as reusability, scalability or interoperability. In this case, the Architecture Notebook (artifact from the architecture OpenUP's discipline) contains the SOA reference architecture description adapted to this particular project.

B. *Define Meta-Models*

Since SOA is the selected architecture, it demands a certain type of functionality specification. It also implies the type of models that should be used in the RE process. The Architecture Notebook therefore includes the metamodels identified for the CIM-level requirements model and PIM-level analysis model. In this example, requirements specified as scenarios and use case models provided by stakeholders and captured by the Analyst serve to create the requirements model that conforms to the features metamodel (Figure 7.A). The most important concept of this metamodel is the Service Feature, with one of three refinement types (Decomposition, Specialization, and Implemented-by).

Since SOA was chosen as the reference architecture for the project, the PIM-level analysis models will cover the business process and service layers of the architecture. In this case, models that will be generated in the MDD process are established to conform to the BPMN process metamodel (Figure 7.B) and service metamodel (Figure 7.C).

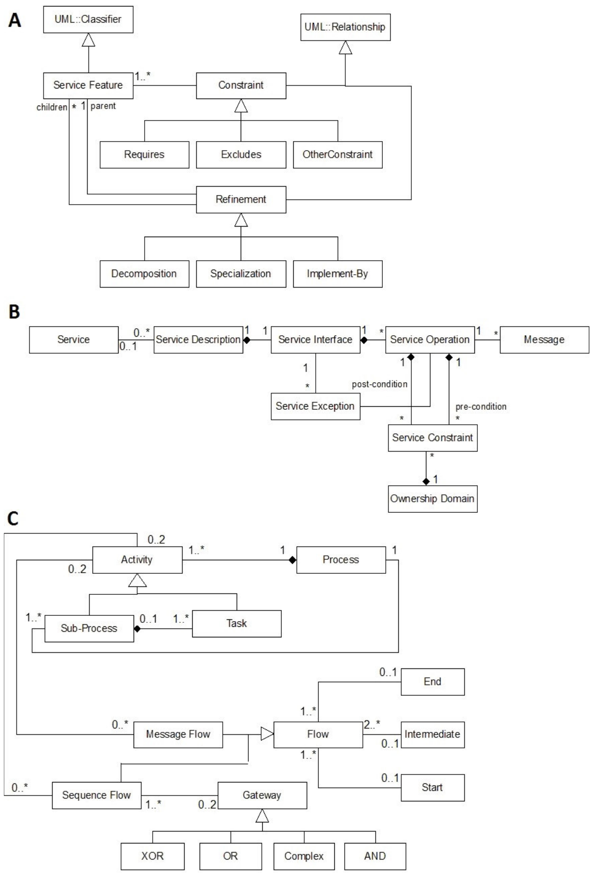

Figure 7. Simplified metamodels used for requirements specification

The most important concepts of these models are the service with its description containing operations, messages and exceptions, and also the process with the flow of activities.

C. Define and Plan Transformations

At this stage, the Transformations Specifier prepares the TRC, which documents transformations from the CIM model to the PIM-level model. In this case, these rules describe a transformation from the features model to reach target models such as BPMN and service specification. The transformation described in this example is not straight forward, but consists of two steps: one from the features model to BPMN, and one BPMN to the service specification. In this case, the TIP document describes the order of use of specified transformations.

The transformations in this particular example are performed manually by the Model Analyst. However, in other cases they may be executed automatically through the use of tools that support a particular transformation.

An example of this feature to BPMN transformation, which is described in detail by Montero *et al.* [10], is shown in Figure 8, in which the left-hand side of the transformation presents an element of the source model and the right-hand side presents a corresponding element of the target model.

The service specification is then obtained from the BPMN model by applying one of existing techniques. In this case, the services and their operations were identified using the method described by Azevedo *et al.* [3].

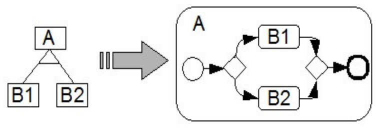

Figure 8. Example of transformation rules

D. Develop Requirements Model

Once the architecture and the metamodels have been identified, the Requirements Model is created. This is done manually. The features model, which constitutes the input for the CIM to PIM transformation, is created on the basis of scenarios and use-cases previously described by the stakeholders. Figure 9 shows an example of a features model for the system's Actor Management functionality. This functionality contains three independent and optional functionalities with which to manage actors.

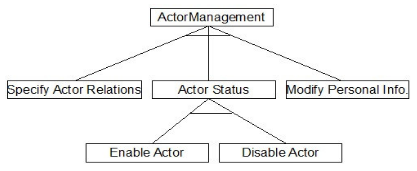

Figure 9. Example of the requirements model

E. Run Transformations

Once the transformations have been defined and planned, the Model Analyst generates the PIM-level models in order to produce the Generated Analysis Model artifact. The specification produced is the input for further design and implementation in the Development discipline. Figure 10 shows a simplified specification of the Actor Management business process represented as BPMN (Figure 10.A), along with the Actor Management service specification (Figure 10.B) that conforms to the aforementioned metamodels.

The Model Dependencies Specification document, which is prepared during the aforementioned process, includes the traceability links between the requirements and all subsequent models that are created as intermediate or final products of the model-driven process.

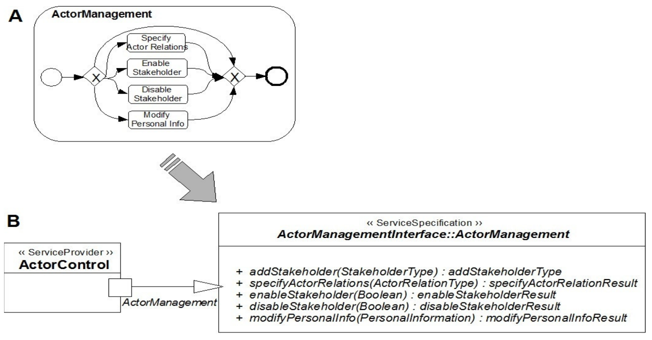
A
ActorManagement
Specify Actor Relations
Enable Stakeholder
Disable Stakeholder
Modify Personal Info
B
« ServiceProvider »
ActorControl
ActorManagement
« ServiceSpecification »
ActorManagementInterface::ActorManagement
+ addStakeholder(StakeholderType) : addStakeholderType
+ specifyActorRelations(ActorRelationType) : specifyActorRelationResult
+ enableStakeholder(Boolean) : enableStakeholderResult
+ disableStakeholder(Boolean) : disableStakeholderResult
+ modifyPersonalInfo(PersonalInformation) : modifyPersonalInfoResult

Figure 10. Example of Generated Analysis Model for Actor Management

The example presented here shows how the requirements specification process may be conducted for SOA-based systems development. Taking advantage of the model-driven requirements process signifies that it is systematized but also agile when preparing different kinds of specifications. The process is accompanied by a set of artifacts that provide well-documented guidelines for all interested project development members.

V. CONCLUSIONS AND FURTHER WORK

This paper presents an extension of OpenUP, emphasizing the use of models as requirements specifications in the context of MDD. This extension redefines the original Requirements discipline in the OpenUP and proposes a new discipline called Model-Driven Requirements. Our methodological approach is an agile RE method for project managers who would prefer to adapt a MDD RE process to particular software architecture rather than using another general approach. We believe that this flexible MDD approach is a solution to the common "one method fits all" problem of generic methodologies.

In our approach we apply model-driven techniques to extend OpenUP to support different architectures and project needs. It improves the development process defined by OpenUP in that it is not only model-based, but also model-driven. This makes OpenUP/MDRE more compliant to maturity model approaches (such as that of the MDD Maturity Model [14]) needed in industry for the incremental adoption of MDD processes. The extension was developed as a plug-in library for EPF. It includes new content elements, such as: artifacts, roles, tasks, and process elements, i.e., activities and capability patterns, to guide software engineers who attempt to follow an MDD approach in their software projects.

The application of this approach to an MDD project has been described, and shows that the use of a model-driven RE has an important influence on the entire development process.

As further work, we plan to provide a tool support with which to easily create the artifacts defined for this model-driven development process (transformation rules, transformations iteration plan, model validation plan, etc.). This will be addressed by providing document templates and creating artifacts with wizards.

Finally, we are involved in the redefinition of the OpenUP/MDRE based on the artifact-driven approach which, in our opinion, better covers the different aspects of an MDD process definition.

ACKNOWLEDGEMENT

This research work is supported by the MULTIPLE project (with ref. TIN2009-13838) funded by the "Ministerio de Ciencia e Innovación (Spain)".

REFERENCES

[1] The Eclipse Process Framework (EPF) project, www.eclipse.org/proposals/beacon/ (last accessed 8/8/2011)

[2] The OpenUP methodology web site (November 2010), http://epf.eclipse.org/wikis/openup/ (last accessed 8/8/2011)

[3] L.G. Azevedo, F. Santoro, F. Baio, J. Souza, K. Revoredo, V. Pereira, and I. Herlain, A Method for Service Identification from Business Process Models in a SOA Approach. In: Enterprise Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing, vol. 29, pp. 99-112. Springer Berlin Heidelberg (2009)

[4] M. Azoff, The Benefits of Model Driven Development. White paper, Butler Group (March 2008), http://www.ca.com/us/products/detail/CA-Gen.aspx (last accessed 8/5/2011)

[5] P. Jamshidi, S. Khoshnevis, R. Teimourzadegan, A. Nikravesh, and F. Shams, Toward Automatic Transformation of Enterprise Business Model to Service Model. In: PESOS '09: Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, pp. 70-74, IEEE CS, Washington, DC, USA (2009)

[6] P. Kruchten, The Rational Unied Process: an Introduction. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)

[7] G. Loniewski, A. Armesto, and E. Insfran, Incorporating Model-Driven Techniques into Requirements Engineering for the Service-Oriented Development Process. In: ME'11: Proceedings of the 2011 Conference on Method Engineering, vol. 351, pp. 102-107, Springer Boston (2011)

[8] G. Loniewski, E. Insfran, and S. Abrahao, A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In: Petriu, D.C., Rouquette, N., Haugen, (eds.) MoDELS. Lecture Notes in Computer Science, vol. 6395, pp. 213-227, Springer (2010)

[9] T. Menzies, Editorial: Model-Based Requirements Engineering. Requirements Eng. 8(4), 193--194 (2003)

[10] I. Montero, J. Pena, and A. Ruiz-Cortes, From Feature Models to Business Processes. In: Proceedings of the 2008 IEEE Int. Conf. on Services Computing, vol. 2, pp. 605-608, IEEE Computer Society, Washington, DC, USA (2008)

[11] OMG (Object Management Group): Meta Object Facility (MOF) Core Specification Version 2.0 (2006), http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[12] OMG (Object Management Group): Software Process Engineering Metamodel (SPEM) (January)

[13] OMG (Object Management Group): Model Driven Architecture: The Architecture of Choice for a Changing World (2010), http://www.omg.org/mda

[14] E. Rios, T. Bozheva, A. Bediaga, and N. Guilloreau, MDD Maturity Model: A Roadmap for Introducing Model-Driven Development. In: ECMDA-FA. pp. 78-89 (2006)

[15] L. Zhang and W. Jiang, Transforming Business Requirements into BPEL: A MDA-Based Approach to Web Application Development. In: WSCS'08: IEEE Int. Workshop on Semantic Computing and Systems, 2008, pp. 61-66 (2008)