Incorporating Model-Driven Techniques into Requirements Engineering for the Service-Oriented Development Process

Grzegorz Loniewski, Ausias Armesto, Emilio Insfran

ISSI Research Group, Department of Computer Science and Computation
Universidad Politécnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
grlo@posgrado.upv.es, aarmesto@indra.es, einsfran@dsic.upv.es

Abstract. Modern information systems, which are the result of the interconnection of systems of many organizations, run in variable contexts, and require both a lightweight approach to interoperability and the capability to actively react to changing requirements and failures. *Model-Driven Development* (MDD) and *Service-Oriented Architecture* (SOA) are software development approaches that deal with this complexity, reducing time and cost development and augmenting flexibility and interoperability. Although, requirements engineering is accepted as a critical activity in these approaches, there is a need to appropriately integrate and automate the requirements modeling and transformation tasks as part of MDD and SOA development approaches. Our proposal is a *Rational Unified Process* (RUP) extension, in which the requirements discipline is placed in a model-driven context in order to derive SOAs. This paper includes the definition of a model-driven requirements process including activities, roles, and work products.

Keywords: Model-driven development, SOA, RUP extension, Requirements Engineering

1 Introduction

The domains and problems for which it would be desirable to introduce information systems are currently very complex and the software development process is thus of the same complexity. Several software development approaches have been introduced in order to speed up and facilitate this process through its automation and the division of the final product into smaller building blocks.

One of these approaches is Service-Oriented Architecture (SOA). SOA is a logical means of designing a software system to provide independent services that are aligned with business processes. SOA strengthens such factors as reusability, scalability or interoperability.

Another approach that improves the development process of complex applications is Model-Driven Development (MDD). This is a model-based approach that promotes the separation of concerns between the business specifications and

their implementation. This separation is achieved through the use of models that allow the level of abstraction to be elevated. It provides a means for development process automation by model transformations and code generation rules.

These approaches are very often used during the design phase of software development, less often in the analysis phase, and hardly ever in the initial phase of a software project when requirements have to be captured, understood and specified. Moreover, even though the aforementioned approaches provide the means to support the software development process, all such techniques, methods or architecture styles are of little use without a well-defined process that places them in a particular context.

In our opinion, the solution to providing a successful automatable development of SOA-based systems is a well-defined, and flexible model-driven process, which is requirements engineering (RE). A good basis for the development of such a methodological approach is the Rational Unified Process. RUP is a costumizable and extensible software engineering process that provides a disciplined approach with which to define tasks and responsibilities in an organized system development [5]. Although various attempts to adapt RUP to MDD principles exist, e.g. Agile Unified Process (AUP), the development process remains mainly manual.

This paper presents a proposal for a RUP extension and adaptation with which to develop SOA-based systems by using model-driven techniques. The main extension in this methodology is the replacement of the Requirements discipline with the Model-Driven Requirements. This work can be considered as an interesting contribution for those software process engineers who are faced with the challenge of guiding software development projects that follow a model-driven development approach from the requirements elicitation.

This work is structured as follows. Section 2 presents works related to the aforementioned area of concern. Section 3 provides an overview of the software process engineering standards. Section 4 presents an overview of the main goals of the methodology, focusing on the content and process elements of the Model-Driven Requirements discipline in the context of SOA-based systems development. Finally, Section 5 contains some conclusions and future work.

2 Related works

A variety of modeling techniques and methodological approaches for service-oriented software development have been published in literature. Ramollari *et al.* [9] present a state-of-the-art survey on current service-oriented development approaches, among others, *Service Oriented Unified Process* (SOUP) [7] and *Service Oriented Modeling and Architecture* (SOMA) [1]. However, none of these methodologies describes a complete methodological automated process that includes RE techniques.

There exist other approaches not included in the aforementioned survey, such as: MINERVA framework [3], or SOA-MDK [2], which apply model-based paradigms to service-oriented development methodology. However, these ap-

proaches do not include any automation while producing the services specification. SOA-MDK approach proposes the application of the Model-Driven Architecture (MDA) principles within the context of reference models. However, the nature of the model-driven base of this approach remains unclear.

Several generic methodologies are based on the MDD principles, since these have gained many enthusiasts over the last decade. However, to the best of our knowledge, a complete development process for MDD that incorporates the requirements techniques has not been defined [6]. One such approach is OpenUP/MDD, which is a very simplified RUP version intended for small teams. It is consistent with the MDA, but focuses solely on the transformations from the PIM to the PSM level and does not cover transformations from requirements. In this context, our proposal for the RUP extension and the OpenUP/MDD approach are complementary.

3 Software Process Engineering

Different software development processes use different concepts and notations to define the contents of the methodology. The need to unify all these concepts and notations has therefore emerged leading to the introduction of the *Software Process Engineering Metamodel* (SPEM) [8] standard by the OMG. SPEM provides a complete metamodel based on the *Meta Object Facility* (MOF) to formally express and maintain development method content and processes. The *Unified Method Architecture* (UMA) is an evolution of SPEM v1.1 and defines the schema and terminology used to represent methods consisting of method content and processes. IBM and OMG have worked on UMA to make it part of SPEM 2.0. The UMA engineering process is employed in this extension, defined by the use of IBM Rational Method Composer (RMC) [4], which is a UMA-based comprehensive process authoring tool that provides extensive method authoring and publishing capabilities [10].

4 RUP Extension for the Model-Driven Requirements

In classic RUP, the *Requirements* discipline serves to establish the agreement with customers with regard to what the system should do, and define boundaries of the system. In our opinion, it should also provide a means for developers to better understand the requirements, it being like a bridge between the domain experts, stakeholders and the IT people.

Figure 1.A illustrates the RUP hump chart in which the *Requirements* discipline is replaced with a new *Model-Driven Requirements* (MDR) discipline. It also emphasizes the *Environment* discipline which serves as a means to adapt this process to SOA-based systems.

As shown in Figure 1.A, the new MDR discipline is a concern from the *Inception* phase to the *Transition*. 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. Since we concentrate on model use in the MDD context, the workload in the *Analysis & Design* discipline in the *Elaboration* phase decreases depending on the degree of automation of activities from the MDR discipline.
This approach was designed to support SOA-based system development. One of the main differences between RUP and the process proposed in our extension is the approach used to relate requirements and the system architecture. In RUP, the architecture is defined on the basis of previously created use cases and scenarios, chosen as the requirements that define strategic architectural elements. This RUP extension is architecture-oriented. It is the architectural pattern identified for the system, in this case to SOA, that becomes a basis for the MDD process definition.

4.1 Activities and workflow

A set of new activities is contributed and the discipline workflow has been replaced. Figure 1.B demonstrates the MDR discipline workflow. New or altered activities introduced with regard to the classic RUP *Requirements* discipline are marked with a star. Owing to space constraints, we shall comment only briefly on the newly introduced activities, with which the PIM-level model is defined and generated.
Identify a Candidate Architecture. This activity is performed in the early *Elaboration* phase and is essential activity for the software development process in that it determine which artifacts need to be developed (type of model at the PIM-level that the architecture implies), and the MDD process to be followed.
Define the Transformation Rules. This activity is the most essential in this approach. Within this activity, the elements of the source and target models are identified and well-documented. The transformation language is also chosen, and the transformation automation level and tool support are specified. Transformation rules are described in a specially prepared Transformation Rules Catalog.

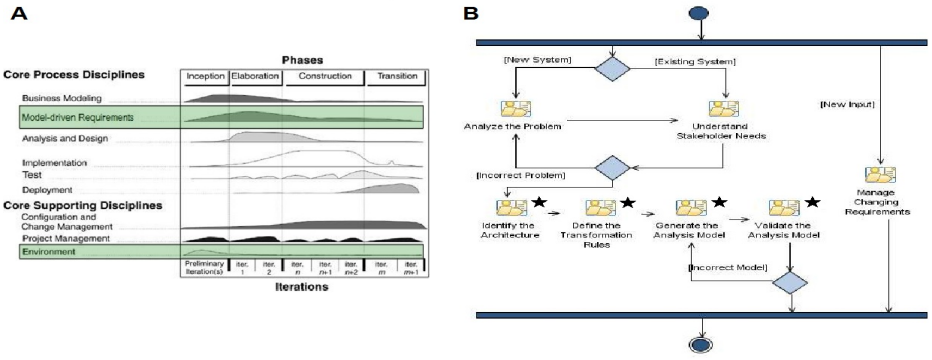

Fig. 1. A) RUP Extension disciplines, B) Model-Driven Requirements Workflow

Generate the Analysis Model. This activity concludes the entire requirements modeling process by creating an architecture-oriented PIM. This is the result of all the previously performed activities, taking advantage of the artifacts created in the *Business Modeling* discipline. The output product of this activity is the input for further process analysis, design and implementation tasks.

4.2 Work Products

Owing to space constraints, we shall comment only briefly on the most important artifacts which have been introduced or improved in the new discipline.
Software Architecture Document (SAD). This artifact, from the *Analysis & Design* discipline, is here initiated on the basis of the system architecture that has been settled on. It is an important artifact for architects and analysts during the entire development process.
Transformation Rules Catalog (TRC). The transformation rules are specified on the basis of the source and target model elements identified. This artifact should consist of a precise description of rules, mappings and refinements, which also provides the basis for the requirements traceability.
Transformation Iteration Plan (TIP). Requirements transformations are usually quite complex and are frequently based on defining intermediate models. A sequence of transformations rather than a single transformation is therefore necessary. This artifact is created to plan a logical order of the transformation to be performed.
Generated Analysis Model (GAM). This is the most important work product in the discipline, it being a source for further transformations to generate PSMs. Its type of content depends on the architecture identified, while the model must suit the architectural pattern considered.

4.3 Roles

As the new discipline is based on the *Requirements* discipline, it maintains the roles originally defined by RUP. The only exception is that the *Requirements Specifier* has been replaced with two additional roles related to the model-driven context activities: *Model Analyst* and *Transformation Specifier*. Only the newly introduced roles are briefly described owing to space limitations.
Model Analyst. During the MDR discipline, the *Model Analyst* coordinates a number of tasks related to: model transformations, model traceability and model validation. The main artifact for which this role is responsible is the GAM. This role also collaborates with the *System Analyst* to accomplish a number of tasks related to requirements modeling and traceability.
Transformations Specifier. This role is responsible for specifying the details of transformation rules to transform requirements model into analysis model. It is a good practice to establish such rules in the meta-model level, which also simplifies the requirements traceability.

5 Conclusions

In this paper we have presented an extension of RUP by placing emphasis on the use of models as requirements representation in the context of MDD. This extension proposes a new discipline called Model-Driven Requirements that substitutes the Requirements discipline from the classic RUP. This approach through the application of architecture-oriented model-driven techniques attempts to extend RUP to specific project needs. It improves the standard development process defined by RUP in that it is not only model-based, but also model-driven.

This extension includes new content elements, such as: artifacts, roles, tasks, activities and capability patterns, to guide software engineers who attempt to follow an MDD approach in their software projects.

As further work, we plan to validate the approach by measuring the effort involved in the maintainability of requirements and the number of failures caused by errors in preparing the requirements specification in comparison to other similar sized projects carried out with the use of classical methodologies.

References

1. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. 47(3), 377–396 (2008)
2. Barn, B., Dexter, H., Oussena, S., Sparks, D.: Soa-mdk: Towards a method development kit for service oriented system development. In: Magyar, G., Knapp, G., Wojtkowski, W., Wojtkowski, W.G., Zupancic, J. (eds.) Advances in Information Systems Development, pp. 191–201. Springer US (2008)
3. Delgado, A., Ruiz, F., de Guzmán, I.G.R., Piattin, M.: A Model-driven and Service-oriented framework for the business process improvement. Journal of Systems Integration vol. 1(3) (2010)
4. Haumer, P.: IBM Rational Method Composer: Part 1: key concepts (December 2005), `http://www-128.ibm.com/developerworks/rational/library/jan06/haumer/`
5. Kruchten, P.: The Rational Unified Process: an introduction. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
6. Loniewski, G., Insfrán, E., Abrahão, S.: A systematic review of the use of requirements engineering techniques in model-driven development. In: MoDELS (2). LNCS, vol. 6395, pp. 213–227. Springer (2010)
7. Mittal, K.: Service Oriented Unified Process (SOUP) (2006), `http://www.kunalmittal.com/html/soup.shtml`
8. OMG (Object Management Group): Software Process Engineering Metamodel (SPEM) (April 2008)
9. Ramollari, E., Dranidis, D., Simons, A.J.H.: A survey of service oriented development methodologies, `http://staffwww.dcs.shef.ac.uk/people/A.Simons/research/papers/soasurvey.pdf`
10. Shuja, A., Krebs, J.: IBM®Rational Unified Process®Reference and Certification Guide: Solution Designer. IBM Press (2007)