A Model Driven Methodology for enabling Autonomic Reconfiguration of Service Oriented Architecture

Emna Mezghani[1,2,3], Riadh Ben Halima[1,2,3], Ismael Bouassida Rodriguez[1,2,3] and Khalil Drira[1,2]
[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France
[3] University of Sfax, ReDCAD, B.P.W, 3038 Sfax, Tunisia
{emna.mezghani,khalil.drira}@laas.fr, riadh.benhalima@enis.rnu.tn, bouassida@redcad.org

ABSTRACT

Autonomic systems are known by their abilities to manage and reconfigure themselves according to the context changes that can include the evolution of functional and/or non-functional requirements, without human intervention. The design and the management of such complex systems manually is a hard task since both functional and non-functional requirements should be taken into consideration. In this paper, we propose a model driven methodology which enables the dynamic reconfiguration by generating autonomic architectures from high level descriptions of functional requirements. Based on transformation and refinement rules, this methodology automates the incorporation of non-functional requirements to the initial architecture. Our work follows the Model Driven Architecture (MDA) to cover the different abstraction levels.

Keywords

Automated Design, Model Driven Architecture, Autonomic Computing, Graph transformation/refinement

1. INTRODUCTION

From the user-oriented point of view, the QoS comes at the price of dependability, due to runtime variations in terms of services availability, and network connection/infrastructure availability. In order to provide the QoS to users, the application should be able to dynamically reconfigure itself at runtime according to requirements. This is a challenge especially for SOA applications which are dynamic and heterogeneous by nature. Several research activities handle the dynamic reconfiguration at runtime and show different ways to configure systems. The most of work adopt the autonomic computing to build their solutions.

However, for existing white box applications -we have access to the code source-, the modification of the code to integrate the autonomic components is usually a source of errors and bugs. While, for black box applications, we can not extend them with some autonomic features (e.g. monitoring component) because we have no access to the code source. The solution of this problem is to conceive the dynamic reconfiguration at design-time. In this context, software architecture play an important role in the development of softwares. It is the role of the architect to specify, formalize and refine the system requirements. But, achieving this formalization manually is a complex task for the architect, especially for complex systems. Therefore, providing model-driven methodologies and tools able to transform and refine software architectures is essential to reduce the complexity of designing autonomic architectures and the related dynamic reconfiguration issues. In this paper, we present our model driven methodology for automating the design of autonomic architectures.

The rest of this paper is organized as follows. In Section 2, we present different related work in the field of transforming and refining systems, then in Section 3, we detail our methodology and describe the different levels. And, finally, Section 4 concludes the paper and gives an overview of our future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$10.00.

2. RELATED WORK

Many research activities adopt the MDA [2, 4, 3] in order to separate concerns at design time. MDA allows refining architectures by adding details (assumptions on the resources, constraints, and services) required to map an abstract model to the chosen platform. It is composed of three levels CIM (Computational Independent Model), PIM (Platform Independent Model) and PSM (Platform Specific Model). Refining the architecture from an abstract viewpoint to a specific platform is widely supported by the MDA approach. Except the work of [3], these works do not consider autonomic systems. Moreover, no details about the formality of the automatic transformations and refinements are provided in order to get a dynamic architecture.

In the context of architecture transformation, several works [5, 1] implement the graph grammar as a formal method to transform and refine their architectures. However, these approaches does not take into account the non-functional requirements and all the transformation describe the functional side of the system.

The refinement of an abstract architecture to obtain a dynamic reconfigurable one is a hard task of the architect especially for the complex systems. We propose a model driven methodology that tackles the complexity of designing autonomic architectures. In our work, we handled non-functional requirements with a focus on managing time related QoS

degradation of SOA applications.

3. OUR MODEL DRIVEN METHODOLOGY

The main goal of our methodology is to automate the design of autonomic SOA based systems starting from an initial model describing functional requirements. This automation, which is based on several transformation and refinement rules between the different MDA levels, allows incorporating new non-functional components to the initial architecture. As shown in Figure 1, our methodology states

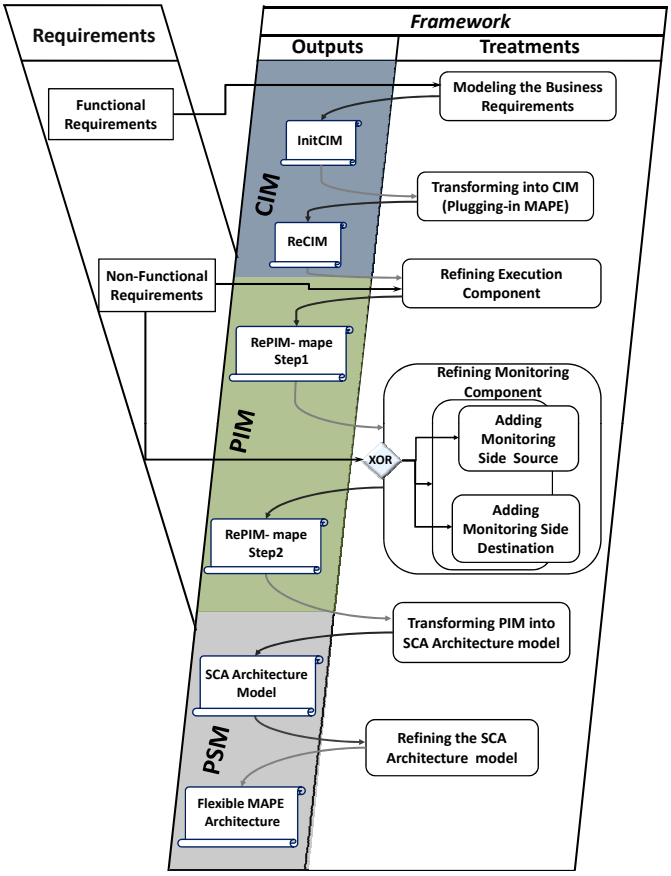

Figure 1: Model Driven Methodology for the automated design of autonomic architecture

two branches namely *Requirements* and *Automated Steps*. The *Requirements* branch specifies both the functional and the non-functional requirements. These requirements will be formalized and refined by the *Automated Steps* that generate intermediated models. These models are clustered into three levels as described in the MDA approach.

In our work, the *CIM* provides an abstract description of the system then it is extended by the autonomic control loop components (MAPE: Monitoring, Analysis, Planning and Execution) to manage non-functional requirements. The *PIM* details the different MAPE components. And, the *PSM* represents the real system design for implementation. Both of the CIM and the PIM levels, modeled with graphs, are detailed into sub-models in order to refine the abstract model. The *CIM* level consists in two sub-models: the *InitCIM*, which describes the initial system, and *ReCIM*, which extends the *InitCIM* with new components to support the QoS management. The *ReCIM* is refined by applying successive graph grammars productions to the PIM which is composed of the RePIM-mape-step1 and the *RePIM-mape-step2* models. The first model details the execution component, while the second one details the monitoring component. These refinements steps include incorporating the non-functional requirements to the system. In the *RePIM-mape-step2*, the refinement is highly dependent from the monitored QoS parameters. We distinguish three models to detail the monitoring module and two nodes can be added having the type "SourceSideMonitor" and/or "DestinationSideMonitor".

After that the transformation toward the *PSM* model is achieved through the *"Transformation Engine"* that takes as input the description of the architecture representing the *PIM* and provides as final output the SCA (Service Component Architecture) based architecture representing the *PSM*. As a result of this methodology, we obtain an autonomic architecture able to dynamically reconfigure itself at runtime.

4. CONCLUSION

In this paper, we introduced a model driven methodology to automate the design of QoS-enabled SOA systems. The result of this methodology is an autonomic system able to dynamically reconfigure itself by managing the QoS at runtime. This methodology leads to obtain significant gains on software scheduling and cost, especially for complex systems.

Our future work will focus on extending this methodology to support the UML as a modeling language and mapping this metamodel to graphs in order to keep on the correct by design concept. Also, we propose to introduce the semantic description to characterize the functional and the non-functional requirement in order to provide an accurate service selection during the reconfiguration.

5. REFERENCES

[1] L. Baresi, R. Heckel, S. Thöne, D. Varro, D. Varró, and P. D. Milano. Style-based refinement of dynamic software architectures. In *In Proc. 4 th Working IEEE/IFIP Conference on Software Architecture, WICSA4*, 2004.

[2] X.-X. Cao, H.-K. Miao, and Q.-G. Xu. Modeling and refining the service-oriented requirement. In *Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, TASE '08, 2008.

[3] J. Pena, M. G. Hinchey, R. Sterritt, A. Ruiz-Cortes, and M. Resinas. A model-driven architecture approach for modeling, specifying and deploying policies in autonomous and autonomic systems. In *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, DASC '06.

[4] V. Rafe, R. Rafeh, P. Fakhri, and S. Zangaraki. Using mda for developing soa-based applications. In *Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01*, ICCTD '09, 2009.

[5] I. B. Rodriguez, K. Drira, C. Chassot, K. Guennoun, and M. Jmaiel. A rule-driven approach for architectural self adaptation in collaborative activities using graph grammars. *Int. J. Autonomic Comput.*, 1(3), May 2010.