Towards Open Self-Adaptation in a Distributed Multimedia Middleware Platform

José Augusto de Medeiros[1], Adilson Barbosa Lopes[1], Carlos Eduardo da Silva[2]

[1]Departamento de Informática e Matemática Aplicada (DIMAp) – Universidade Federal do Rio Grande do Norte (UFRN)
59.072-970 – Natal – RN – Brazil

[2]Escola de Ciências e Tecnologia (ECT) – Universidade Federal do Rio Grande do Norte – 59072-970 – Natal – RN - Brazil

jamedeiros@gmail.com, adilson@dimap.ufrn.br, carlos.silva@ect.ufrn.br

Abstract. *A self-adaptive software system uses a sequence of actions (called adaptation plan) to modify its structure and/or behavior in response to change. Normally, such actions are defined at development time, resulting in a closed adaptation, where all possible plans are well known beforehand. In this paper we propose the use of a process generation framework based on AI planning and model transformation for achieving open adaptation in a distributed multimedia platform.*

1. Introduction

Self-adaptation has been increasingly receiving attention from different research communities as the means for dealing with the increasing complexity and dynamism of today's software system. Among the techniques employed to achieve self-adaptation, we focus our attention to feedback control loops, which has been recognized as an important factor in software management and evolution, as show by [Brun et al. 2009]. This technique is composed of four basic activities, which provide a fundamental role in managing software self-adaptation. The cycle starts with a collection of data about the system. This data is then analyzed to determine whether an adaptation is necessary. Once an adaptation is needed, an adaptation plan is defined, and then executed for modifying the system.

Those ideas have been applied in the domain of distributed multimedia system through the development of the Cosmos Framework [Lopes 2006]. This framework was designed to support configuration and management of resources in a distributed environment. Cosmos adopts a closed adaptation model, where a set of plans for adapting the system are defined at development time. However, it is difficult to anticipate at design-time all possible contexts of adaptations. For example, resources considered during the design of the adaptation plan may not be available during the system execution [Da Silva and De Lemos, 2011].

In order to effectively deal with the variability and complexity of today's environment, self-adaptive software system must gain some autonomous characteristics being able to generate adaptation plans during runtime, and for such, the use of artificial intelligence becomes an interesting approach. In this context, our goal is to change the

adaptation model in the Cosmos framework to an open adaptation model, where the adaptations plans are produced in runtime [Oreizy et al. 1999]. This will be achieved through a refactoring of the Cosmos framework and its integration with an instantiation of a framework for the dynamic generation of processes presented by [Da Silva, 2009]. This paper presents the refactoring of Cosmos framework for supports the open adaptation model through the use of Process Generator framework

2. Background: A Framework for Process Generation

The framework for dynamic generation of process presents a reusable solution for generation of adaptation plans, and can be instantiated into different application domains. The framework is based on three technologies, model transformation, artificial intelligence (AI) planning and workflow management. Model transformation is used to translate domains specific model into AI planning problems, which are then used to create a plan that is translated into a workflow and executed by a workflow management system. The framework is composed of three main elements: a domain model, a reference process and a supporting infrastructure.

The domain model encompasses domain independent and domain dependent models, metamodels and transformations rules that are used during the generation. The reference process describes the main activities associated with the generation, while the supporting infrastructure identifies the different components that provide support for the execution of the references process. These constitute the basis of the framework and are customized according with the application domain where the framework is being instantiated.

According to [Da Silva 2009], the instantiation of the framework for process generation requires the definition of some models and transformation based on the application domain. In this way, we need to create: a metamodel that captures the elements of the application domain; a planning domain model according to this metamodel; a set of transformations rules for translating models that conform to this metamodel into planning problems; and a set of tasks templates that correspond to several adaptations actions that can be performed in the domain and will be used to compose an adaptation plan.

3. The Proposed Solution

Our main purpose is to incorporate into Cosmos the ability to generate adaptation plans during runtime. To this end, we are integrating Cosmos with a framework for dynamic generation of process, which required some modifications in the Cosmos Framework. In order to accommodate these modifications, we have conducted a refactoring of Cosmos. In the sequel, we present an overview of our proposed Cosmos architecture, which serves as execution environment for generated adaptation plans.

Figure 1 show an overview of our proposed solution. The activities of the feedback loop are delegated to specialized elements. Cosmos has a Model Manager component, which is responsible for maintaining a model of the system that is used to support state monitoring and control of system during execution. Based on this model, Probes are put in place for monitoring the components of running application. These Probes collect QoS related data that is fed into QoS Manager component.

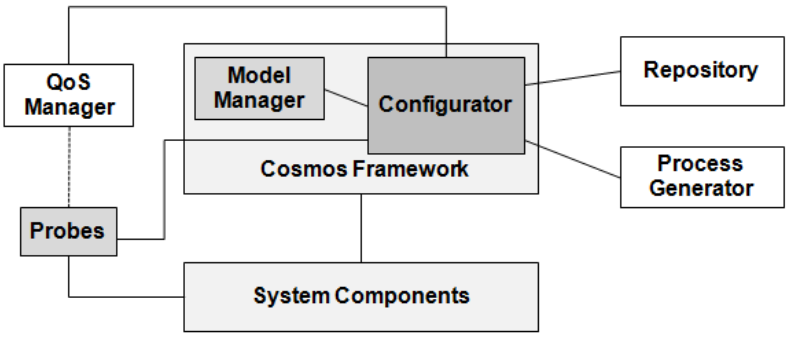

Figure 1. Cosmos Framework Overview.

The QoS Manager analyzes the collected data based on predetermined criteria and, if there is a need to adapt, informs the Configurator. At this point, the Probes are deactivated, while the Configurator is activated. It selects a new configuration for the system and sends it, together with a model of the current system configuration to the Process Generator component. The Process Generator component then generates and executes an adaptation plan through the Configurator component, which is responsible for effecting adaptations on the system components and for updating the Model Manager with a model of the new configuration. Once an adaptation is successfully completed, the Probes are reactivated and the system resumes its operation.

In order to support the new adaptation process, we modified the Cosmos component model so it can represent both the running system and its architecture, and to do so, we extended the model used by *xADL* architectural language, adding the architectural elements of the Cosmo framework. With the changes in the representation of Cosmos component model, we updated the Model Manager accordingly. Also we modified the Configurator component in order to support to new adaptation model, in this adjustment, we excluded interfaces that were not in use and we added other interfaces that are used during adaptation.

4. Related Work

Fonseca et al. [Fonseca, Di Beneditto e Werner, 2012] presents a mechanism for the execution of adaptation plans that generated by a planner. The generated plan is transformed into adaptation actions that are enacted through call to the API offered by the execution. If an error occurs, the actions that were executed are undone and the effected components returned to the initial state. In our approach the Process Generator, which contains a planner, generates and executes the adaptation plans through the integration interface of Cosmos framework. In case of failures during execution of the adaptation process, a new adaptation plan is generated and executed.

Tajalli et al. [Tajalli et al. 2010] presents a Plan-based Layered Architecture for Software Model-driven Adaptation. Their approach utilizes an ADL together with planner to enable dynamic replanning in the architectural domain. This architecture collects data from system components and analyzes these data to check for the need to adapt. With need for adaptation, it creates an adaptation plan to be executed affecting the system components. An architectural middleware platform to support was introduced, however, no implementation details are given of this middleware. In our

approach besides realize adaptations through a planner, we focus on do it in specialized components to obtain a low coupling. With this, we can use other planners and QoS Managers that they provide integration interfaces.

5. Conclusion

This work has been based on the refactoring of the Cosmos framework, in which we incorporated the use of an ADL for representing system configurations, adjustments for several Cosmos components to support a new model representation. Such changes were needed for integrating Cosmos with a Process Generator framework in order to include supports to a new adaptation model.

To demonstrate the use of the Cosmos framework, we are preparing a broadcast video system, composed of transmission and reception components. From this application we may see and evaluate the use of an open adaptation model in the Cosmos framework.

References

Blair, G. Bencomo, N. and France, R. B. (2009). Models@run.time. IEEE Computer, 42(10):22-27.

Brun, Y. et al. (2009). Engineering Self-Adaptive Systems through Feedback Loops. Software Engineering for Self-Adaptive Systems, pp 48-70. Springer.

Da Silva, C. E. and De Lemos, R. (2011). A framework for Automatic Generation of Process for Self-Adaptive Software Systems. Special Issue on Automatic and Self-Adaptive Systems at Informatica, vol. 35, pp 3-13.

Fonseca, F. L., Di Beneditto, M. E. M., Werner, C. M. L. (2012). Um mecanismo extensível para a execução de um plano de reconfiguração arquitetural sob o framework OSGi+IPOJO. Nata: UFRN.

Lopes, A. B. (2006). Um Framework para Configuração e Gerenciamento de Recursos e Componentes em Sistemas Multimídia Distribuídos Abertos. Thesis (Ph.D). Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas.

Oreizy, P. et al. (1999). An architecture-based approach to self-adaptive software. Intelligent Systems and Their Applications, IEEE, v.14, p.54-62.

Tajalli, H., Garcia, J., Edwards, G., and Medvidovic, N. (2010). PLASMA: A Plan-based Layered Architecture for Software Model-driven Adaptation. Automated software, 467-476.