Towards Automated Deployment of Distributed Adaptation Systems

Mohamed Zouari, Ismael Bouassida Rodriguez

Towards Automated Deployment of Distributed Adaptation Systems

Mohamed Zouari[1,2] and Ismael Bouassida Rodriguez[1,2,3]

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, LAAS, F-31400 Toulouse, France
[3]ReDCAD, University of Sfax, B.P. 1173, 3038 Sfax, Tunisia
{mohamed.zouari,bouassida}@laas.fr

Abstract. The development of a single software product is inefficient when groups of product are related since the development cost could be high. In addition, some products need to be self-adaptive in order to take into account the execution context changes. In this case, the implementation and management of the adaptation mechanisms variability is challenging especially for distributed systems due to the distribution issues. We address in this paper such issues by proposing a method for the software engineering of distributed adaptation systems. We propose an architectural model for distributed management of dynamic adaptation. We define also a graph grammar based approach to automate the tasks needed to construct and configure the adaptation system[1].

Keywords: Distributed adaptation, Software architectural model, Automated deployment, Graph grammar

1 Introduction

Several applications running in fluctuating and heterogeneous environments require dynamic adaptation [1]. This is especially necessary when users may have different and variable QoS requirements and resources are highly dynamic and unpredictable. The adaptation approach enables to deal with the different fluctuations in available resources, to meet new user requirements, and to improve the application services.

In general, an adaptation engine monitors the execution context (resources characteristics, user profile, terminal capabilities, etc) in order to trigger dynamic adaptation whenever it detects significant variations. Then, it makes decisions regarding the adaptation and controls the modification of the application in order to achieve the appropriate configuration. When a decentralized application is running in heterogeneous environments, distributed adaptation system may be required in order to improve the adaptation mechanisms quality such as efficiency, robustness, and scalability. The distributed management of adaptation leads to the concurrent execution of multiple adaptation processes performed

[1] This work is partially funded by the IMAGINE IP European project

by several engines. The customization and the deployment of the engines may need to perform complex dedicated tasks and be time consuming. Facilitating these tasks and reducing the development cost of distributed adaptation systems is challenging. In fact, current approaches do not offer development facilities of distributed adaptation systems where the activities of several adaptation engines are coordinated without a central control entity.

In this paper, we propose an approach for easy customization and deployment of distributed adaptation systems.The aim of this approach is to provide facilities for the elaboration of the system configuration and for its deployment in automated way. Actually, we provide firstly an architectural model of distributed adaptation systems. Secondly, we offer a tool called factory that performs the deployment tasks. The factory processes the appropriate system configuration using a graph grammar-based approach and then, it sets up the system. Our case study concerns data management in medical environments for collaborative remote care delivery. We enable to make self-adaptive data replication systems in order to improve the data availability and response times for data requests.

The remainder of the paper is organized as follows. We present our method to build distributed adaptation systems in Section 2. Then, we conclude and discuss future work in Section 3.

2 Method to build self-adaptive applications

Our approach aims at facilitating the component-based development of distributed self-adaptive applications. We follow the separation of concerns principle between the adaptation concerns and the business aspects. Moreover, we externalize the mechanisms of adaptation control for reusability. Some application components provide well defined control interfaces that define primitive operations to observe and modify them (see Figure 1). The adaptation system is connected to the application trough such interfaces to produce a *self-adaptive application*.

We design the adaptation mechanisms in a modular way and we offer facilities for adaptation system building. In fact, we define an tool called factory to facilitate the customization of the adaptation system according to the target application and to ensure the automatic deployment of the system. As shown in Figure 1, an architect provides a software architectural model of adaptation systems and an expert in deployment provides a set of deployment strategies for the set up of concrete systems. This architectural model specifies a set of component types, the possible connections among them and several constraints that must be meet when constructing a concrete system. Each deployment strategy is a set of graph grammars processed to choose the distribution of the adaptation system according to targeted application to adapt and specific required quality criteria of the adaptation system. The configuration manager determinates the adaptation system architecture description that specifies the components that compose the adaptation system, the connections among them, the values of configuration parameters, and the connections with the application

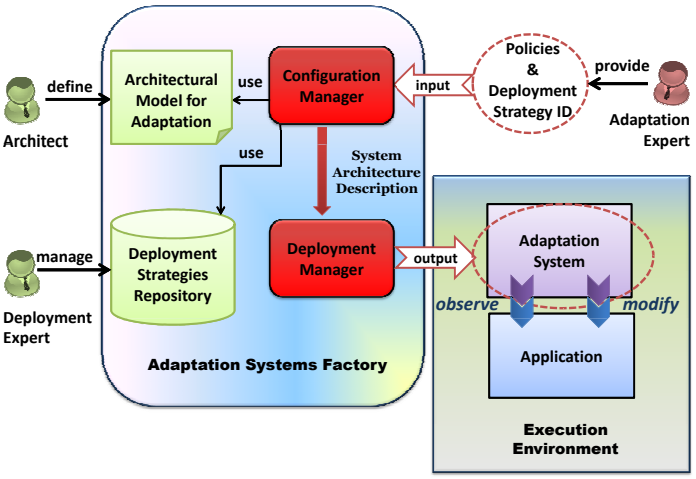

Fig. 1. Building a self-adaptive application

components. The architecture is expressed in the GraphML language which is an XML dialect for representing graphs [2]. For that, the configuration manager applies the grammars related to a chosen deployment strategy and verifies that the constraints of the architectural model are well respected. The selection of the suitable strategy is done by the adaptation expert. We use GMTE[2] in order to process the graph grammars. In addition, the configuration manager uses a set of policies provided by the adaptation expert. These policies enable the configuration manager to customize the behaviour of some adaptation system components. Then, the deployment manager sets up the adaptation system and connect it with the application according to the architecture description provided by the configuration manager. The connections are realized through the dedicated control interfaces provided by the application. The outcome is a self-adaptive application deployed in a decentralized execution environment.

Our architectural model [3] specifies two component types *ContextManager* and *AdaptationManager* to perform the adaptation steps. An adaptation system is composed of several context managers and adaptation managers. A component type *ContextManager* collects, interprets, and aggregates some contextual data. A component type *AdaptationManager* determines which components of the application must be adapted and the means to achieve it.

The factory allows several strategies for deploying the adaptation system. The strategies are divided based on the expected load of the adaptation system, the customization requirements of managers' behaviour, and the distribution requirements. We are currently evaluating the different strategies in order to

[2] Graph Matching and Transformation Engine (GMTE), available at `http://homepages.laas.fr/khalil/GMTE`

provide a guide that allows the adaptation expert to choose the appropriate strategy based on many factors. We are particularly interested in the following deployment strategies: **(1)** *Centralized deployment* : In this situation, a single context manager and a single adaptation manager control the adaptation of all the application components. **(2)** *Location-based deployment* : The difference is the use of multiple context managers and adaptation managers. The distribution is based on geographic location of the application components. Each manager controls a group of components hosted by machines in a specific site (organization, department, vehicle, etc). **(3)** *Service centric deployment* : This deployment strategy is characterized by the consideration of the different services provided by the application components. An adaptation manager (resp., context manager) is associated with a component or group of components that offers specific service (the data consistency achievement, the replica placement, etc). **(4)** *Hybrid deployment* : This strategy combines the two previous strategies: location-based and service centric deployment. The goal is to combine the best of both. **(5)***Distributed deployment* : This strategy is characterized by a fully distributed system where an adaptation manager (resp., context manager) is associated with a single application component.

3 Conclusion

In this paper, we addressed how to manage effectively and in a structured way variability in distributed adaptation systems. We enable variable configuration for distributed adaptation systems in order to meet specific requirements regarding the adaptation system QoS. We presented a systematic approach to model, implant and manage the variability of such systems based on the architectural model. Among the benefits of this model are reusability and the support of several types of variations like the behaviour and the distribution of the system. Our experience shows that our factory provides an effective and easy way to manage variability that turns up during the construction of an adaptation system.

There are several possible directions for future work. We are interested in facilitating more the customization process. Currently, human actor makes decision related to the deployment strategy. It will possible to extend the factory in order to allow choosing automatically the appropriate strategy. Moreover, we are exploring the connection between the architectural model and the existing components models. Our vision is to make the factory able to support several components models and service oriented architectures.

References

1. Cheng, B.H., Lemos, R., Giese, H., Inverardi: Software engineering for self-adaptive systems. Springer-Verlag, Berlin, Heidelberg (2009) 1–26
2. Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., Marshall, M.S.: GraphML Progress Report. In: Graph Drawing. (2001) 501–512
3. Zouari, M., Segarra, M.T., André, F., Thépaut, A.: An architectural model for building distributed adaptation systems. In: IDC2011. (2011) 153–158