# Managing Architectural Reconfiguration at Runtime

*Sihem Loukil, ReDCAD Laboratory, University of Sfax, Sfax, Tunisia*

*Slim Kallel, ReDCAD Laboratory, University of Sfax, Sfax, Tunisia*

*Mohamed Jmaiel, ReDCAD Laboratory, University of Sfax, Sfax, Tunisia*

## ABSTRACT.

*Managing dynamic reconfiguration of software systems is a tedious task in the software development because of the substantially increasing need for continuously available systems even at runtime. In particular, the software architecture of dynamically adaptive systems must continuously adapt to varying environmental conditions and user requirements. Therefore, they propose a wide range of possible configurations. The static enumeration of all the possible configurations is a difficult task. Moreover, not all dynamic reconfiguration operations can be foreseen at design time. Some reconfigurations may appear when the system is already deployed. In this context, we propose to combine the Architecture Description Languages and the Aspect-Oriented Software Development paradigm in order to make the dynamic reconfiguration process easier to design, understand and possible to validate. Also, this combination allows to easily evolving the reconfiguration policies even at runtime.*

*Keywords:* *Software architecture, Dynamic reconfiguration, ADL, Models@Runtime, Architectural constraints*

## INTRODUCTION

Software architecture modeling using Architecture Description Languages (ADLs) is becoming increasingly popular in the early phases of system development. Such languages facilitate the construction of high-level models in which systems are described as compositions of components. They play an important role in developing software systems deployed in large number of domains (companies, banks, air-ports, etc.). Such systems must be always available and continuously adapt to varying environmental conditions and user requirements even at runtime. Hence, they should be modified/maintained during their execution, for example to include new functionalities, without being obliged to stop the system. This dynamic reconfiguration to maintain the system available presents a tedious task. In fact, not all possible reconfigurations that will be applied to the system can be foreseen at the time it is initially built and deployed. Therefore, the system must be flexible to support new needs that may appear during execution.

Most of proposed works (Morin, Barais & Jézéquel, 2008; Morin, Barais, Jézéquel,

DOI: 10.4018/jwp.2013010105

Fleurey & Solberg, 2009; Morin, Barais, Nain & Jézéquel, 2009; Morin, Fleurey, Bencomo, Jézéquel, Solberg, Dehlen & Blair, 2008, Morin, Ledoux, Ben Hassine, Chauvel, Barais & Jézéquel, 2009) do not support the unforeseen reconfigurations at design time. They rely on a set of predefined reconfigurations specified at design time before the deployment and the execution of the system. Thus, such approaches limit the flexibility of the system and do not allow accommodating new requirements at runtime.

To resolve such limitations, we propose an approach to manage the dynamic reconfiguration of adaptive systems. These dynamic reconfigurations are performed through architectural reconfigurations. Architectural reconfigurations consist in modifying the system structure by adding or removing components or connections. Compared to existing work, our approach allows including new functionalities at runtime, which were not foreseen at design time. This is achieved through the combination of the Architecture Description Languages, the Aspect-Oriented Software Development (Filman et al., 2005) paradigm and the Hook methods (IBM) technique. This combination allows to make the dynamic reconfiguration process easier to design, understand and possible to validate. Moreover, it allows to easily evolve the reconfiguration policies even at runtime.

Our approach supports two types of dynamic reconfigurations. The first type is the reconfigurations resulting from a change in the execution context of the running system (adaptive reconfiguration) (Ketfi, Belkhatir & Cunin, 2002). The second type is the reconfigurations resulting from the appearance of new user requirements (evolutional reconfiguration) that requires the manual intervention of the designer on the architectural specification of the system.

In both cases (adaptive and evolutional reconfigurations), the reconfiguration actions are performed first on the model representing the architecture of the system. Applying the reconfiguration actions at the model level before applying them to the running system has the advantage to be able to test their effect when applied as a whole without actually changing the system. Thereby, it is always possible to jump back to the state before starting to apply the reconfiguration actions in case an error is detected saving costly executions of roll-back operations on the system.

After applying the reconfiguration actions on the model, the validity of the obtained configuration is checked against a set of architectural constraints specified at the model level.

If the new configuration is valid (no constraint is violated), then the reconfiguration actions are committed to the running system. Otherwise, it is simply discarded.

We selected the Architecture Analysis and Design Language (AADL) (SEA, 2004), as an ADL, for specifying the architecture of dynamically adaptive systems. This language was extended in our previous work (Loukil et al., 2010) to support the aspect-oriented concepts through our proposed AO4AADL language. AO4AADL allows capturing the crosscutting concerns related to the non-functional and technical properties. It is used in this work to allow the designer monitoring the running system and performing the corresponding reconfigurations. The Hook methods technique is intended to capture the unforeseen reconfigurations.

To make easier the work of the designer, we offer as part of this work a graphical editor called AADL Graphical Editor as an Eclipse plug-in. This editor allows modeling aspect-oriented architectures by integrating all AADL concepts as well as AO4AADL ones. It provides a global view of the modeled system with some level of abstraction that simplifies its management. Once the model is designed using the AADL language and integrating the technical properties as AO4AADL aspects, an automatic code generation process is used to obtain an adequate code using Ocarina tool suite (Vergnaud, Zalila, & Hugues, 2006).

Our approach has the following advantages: (1) An architectural dynamic reconfiguration guaranteed, by updating the application at runtime using a reconfiguration process easy to understand and manipulate, (2) A global overview of the distributed applications due to the