# Dynamic Reconfiguration of Software Architectures Through Aspects

Cristóbal Costa[1], Nour Ali[1], Jennifer Pérez[2], José Ángel Carsí[1] , and Isidro Ramos[1]

[1] Department of Information Systems and Computation,
Polytechnic University of Valencia, Camino de Vera s/n, 46022 Valencia, Spain
ccosta@dsic.upv.es, nourali@dsic.upv.es,
pcarsi@dsic.upv.es, iramos@dsic.upv.es
[2]Technical University of Madrid (UPM)
E.U. Informática, Ctra. Valencia km 7, 28051 Madrid, Spain
jeperez@eui.upm.es

**Abstract.** Currently, most software systems have a dynamic nature and evolve at run-time. The dynamic reconfiguration of software architectures has to be supported in order to enable their architectural element instances and their links to be created and destroyed at run-time. Complex components also need reconfiguration capabilities to evolve their internal compositions. This paper introduces an approach to support the dynamic reconfiguration of software architectures taking advantage of aspect-oriented techniques. It enables complex components to autonomously reconfigure themselves: they are capable of both having knowledge of their current configuration and reconfiguring themselves at run-time. This approach has been developed for the PRISMA aspect-oriented architectural model. A new kind of aspect has been created in PRISMA in order to provide dynamic reconfiguration services to each complex component; it is called the *Configuration Aspect*.

**Keywords:** Dynamic reconfiguration, software architectures, AOSD.

## 1 Introduction

Currently, most software systems have a dynamic nature that requires them to evolve and adapt to changes at runtime. The development of this kind of systems is a complex task since they are large and may consist of heterogeneous entities. As a result, a great effort has to be done in order to provide system reconfiguration at runtime. Software architectures [8] provide techniques for describing the structure of complex software systems in terms of architectural elements (components and connectors) and interactions among them. Software architectures have a hierarchical structure where components can be composed into complex components and these, in turn, can be composed into more complex components.

Dynamic reconfiguration [3] has to be supported in order to enable architectural element instances and links to be created and/or destroyed at run-time. In the last years, a lot of research efforts have been done to address dynamic reconfiguration of software architectures [1,2]. However, most of these approaches address dynamic reconfiguration from a centralized point of view: a global entity is responsible of providing dynamic reconfiguration capabilities to all the architecture. However, architectures are often made up of heterogeneous components and each one has different reconfiguration needs. For this reason, complex components usually need to reconfigure their internal composition by themselves in an autonomous way.

Aspect-Oriented Software Development (AOSD) [4] proposes the separation of the crosscutting concerns of software systems into separate entities called aspects, avoiding the tangled concerns of software and allowing the reuse of the same aspect in different entities of the software system as well as its maintenance. Dynamic reconfiguration is a concern that crosscuts the architecture of those software systems that have a dynamic nature. In order to prevent the dynamic reconfiguration concern from being tangled with the rest of the architecture, a new kind of aspect called *Configuration* has been defined. Thereby, a *Configuration* aspect encapsulates the properties and the behaviour of this concern. Only few proposals have addressed dynamic reconfiguration through aspects [9], and their work is focused at the implementation level instead of dealing reconfiguration at the architectural level.

This paper presents an approach for supporting dynamic reconfiguration in software architectures by means of aspects. The approach enables PRISMA complex components to autonomously reconfigure themselves at runtime, through a *Configuration* aspect, whereas other systems or components of the software architecture are unaware of these dynamic changes.

## 2 PRISMA

PRISMA [5] is a model that integrates AOSD and software architectures in order to describe the architectural models of complex software systems. In PRISMA, architectural elements are specified by importing a set of aspects. Aspects are first-order citizens of software architectures and encapsulate the properties and the behaviour of *concerns* (safety, coordination, etc) that crosscut the software architecture. A PRISMA architectural element can be seen from two different views: the internal and the external. In the external view, architectural elements encapsulate their functionality as black boxes and publish a set of services through their ports. The internal view shows an architectural element as a prism (white box view). Each side of the prism is an aspect that the architectural element imports. The aspects of architectural elements are synchronized among them through *weavings* relationships.

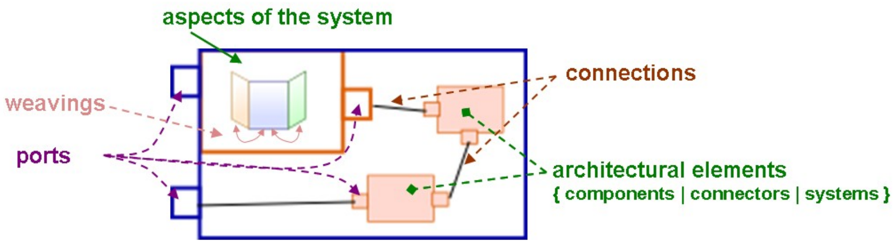

**Fig. 1.** PRISMA Systems

PRISMA has three kinds of architectural elements: components, connectors, and systems. Components and connectors are simple; systems are complex components. A component captures the functionality of software systems whereas a connector acts as a coordinator among other architectural elements. A PRISMA system is a complex component that includes a set of architectural elements (connectors, components and other systems), the connections among them, and its own aspects and weavings (see Figure1).

## 3 Dynamic Reconfiguration Through an Aspect

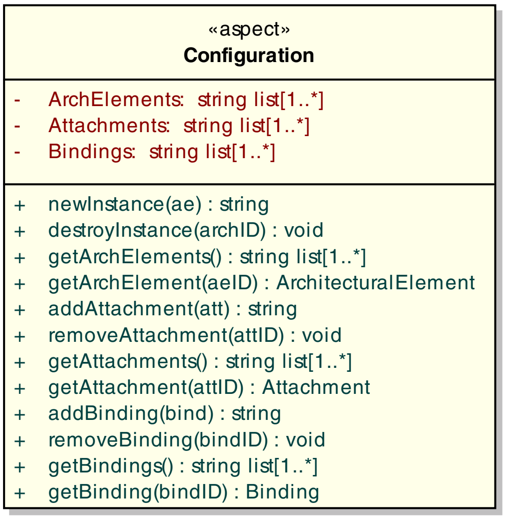
«aspect»
Configuration
- ArchElements: string list[1..*]
- Attachments: string list[1..*]
- Bindings: string list[1..*]
+ newInstance(ae) : string
+ destroyInstance(archID) : void
+ getArchElements() : string list[1..*]
+ getArchElement(aeID) : ArchitecturalElement
+ addAttachment(att) : string
+ removeAttachment(attID) : void
+ getAttachments() : string list[1..*]
+ getAttachment(attID) : Attachment
+ addBinding(bind) : string
+ removeBinding(bindID) : void
+ getBindings() : string list[1..*]
+ getBinding(bindID) : Binding

**Fig. 2.** Configuration Aspect

A PRISMA system is an architectural pattern that defines the type of architectural elements a system is composed of and the kind of valid connections among them. This architectural pattern is instantiated to a specific configuration in order to start the architecture execution. However, dynamic systems need to change their initial configuration several times as a result of its normal behaviour (always satisfying their architectural patterns). For this reason, the system configuration at run-time must be considered as a part of the system state. Thus, a system must have services to query its current configuration, and to change it at run-time. In this way, a system is aware of its current configuration and can decide whether to change it or not. A new concern has been created using the PRISMA Aspect-Oriented Architecture Description Language (AOADL) [6] to encapsulate these services into aspects. This is the *Configuration* concern. A *Configuration* aspect encapsulates every property and behaviour related to dynamic reconfiguration. Any system that needs reconfiguration capabilities to evolve its internal composition imports the *Configuration* aspect.

This aspect has a set of attributes that contain the current configuration of the system and a set of services that maintain and evolve this configuration. The attributes the *Configuration* aspect provides are dynamic lists which store *references* to: (i) the architectural element instances of the system at run-time (component, connector, and system instances), and (ii) connection instances of the system at run-time. The services that the *Configuration* aspect provides (see Figure 2) allow systems to know and modify the data that these attributes store (the system configuration): (i) create or destroy instances of system architectural elements, (ii) establish connections between system architectural elements, and (iii) provide query services in order to make the system aware of its current configuration. In this way, reconfiguration characteristics are specified without extending the PRISMA AOADL with new primitives, and the reconfiguration concern is not tangled with other concerns.

The infrastructure where the software architecture is running is responsible for providing to each system instance the mechanisms of dynamic reconfiguration. PRISMA software architectures are executed by the PRISMANET middleware [7], which provides the mechanisms required for knowing and modifying the running configuration of each system instance at run-time. PRISMANET guarantees that the configuration dependencies among the different elements are preserved.

## 4 Conclusions and Further Work

This paper has presented an approach to dynamically reconfigure software architectures taking advantage of aspect-oriented techniques. This approach consists of providing each complex component with an aspect called *Configuration,* that provides dynamic reconfiguration services at run-time. As a result, this approach has the advantage of keeping dynamic reconfiguration properties and behaviour from being tangled with the rest of the architecture. This has been done without increasing the complexity of the PRISMA AOADL, only by using its original primitives.

Autonomous system instance reconfigurations must satisfy the constraints of the system architectural pattern. We are currently working on specifying how the reflection mechanisms ensure that the requested changes do not violate the restrictions described in the architectural pattern of the system.

**Acknowledgements.** This work is funded by the Dept. of Science and Technology (Spain) under the National Program I+D+I, META project TIN2006-15175-C05-01. This work is also supported by a fellowship from Conselleria d'Educació i Ciència (G. Valenciana) to C. Costa.

## References

1. Bradbury, J.S., Cordy, J.R., Dingel, J., Wermelinger, M.: A Survey of Self-Management in Dynamic Software Architecture Specifications. In: WOSS'04. Proc. of 1st ACM SIGSOFT Workshop on Self-Managed Systems, Newport Beach, California, pp. 28–33. ACM Press, New York (2004)
2. Cuesta, C.E.: Dynamic Software Architecture Based on Reflection (in Spanish). PhD Thesis, Department of Computer Science, University of Valladolid, Spain (2002)
3. Cuesta, C.E., Fuente, P.d.l., Barrio-Solárzano, M.: Dynamic Coordination Architecture through the use of Reflection. In: SAC 2001. Proc. of 2001 ACM Symposium on Applied Computing, pp. 134–140 (2001)
4. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.V., Loingtier, J., Irwin, J.: Aspect-Oriented Programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
5. Pérez, J.: PRISMA: Aspect-Oriented Software Architectures. PhD Thesis, Department of Information Systems and Computation, Polytechnic University of Valencia, Spain (2006)

6. Pérez, J., Ali, N., Carsí, J.Á., Ramos, I.: Designing Software Architectures with an Aspect-Oriented Architecture Description Language. In: Gorton, I., Heineman, G.T., Crnkovic, I., Schmidt, H.W., Stafford, J.A., Szyperski, C.A., Wallnau, K. (eds.) CBSE 2006. LNCS, vol. 4063, pp. 123–138. Springer, Heidelberg (2006)
7. Pérez, J., Ali, N., Costa, C., Carsí, J.Á., Ramos, I.: Executing Aspect-Oriented Component-Based Software Architectures on .NET Technology. In: Proc. of 3rd International Conference on .NET Technologies, Pilsen, Czech Republic, pp. 97–108 (June 2005)
8. Perry, D.E., Wolf, A.L.: Foundations for the Study of Software Architecture. ACM SIGSOFT Software Engineering Notes 17(4), 40–52 (1992)
9. Rasche, A., Polze, A.: Configuration and Dynamic Reconfiguration of Component-Based Applications with Microsoft.NET. In: ISORC'03. Proc. 6th IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing, Hakodate, Japan, pp. 164–171 (2003)