

# On the Interplay of Aspects and Dynamic Reconfiguration in a Specification-to-Deployment Environment

Thais Batista<sup>1</sup>, Antônio T.A. Gomes<sup>2</sup>, Geoff Coulson<sup>3</sup>,  
Christina Chavez<sup>4</sup>, and Alessandro Garcia<sup>3</sup>

<sup>1</sup> Federal University of Rio Grande do Norte (UFRN) – Natal, RN, Brazil

<sup>2</sup> National Laboratory for Scientific Computing (LNCC) – Petrópolis, RJ, Brazil

<sup>3</sup> Lancaster University – Lancaster, UK

<sup>4</sup> Federal University of Bahia (UFBA) – Salvador, BA, Brazil

thais@ufrnet.br, atagomes@lncc.br, geoff@comp.lancs.ac.uk,  
flach@dcc.ufba.br, a.garcia@lancaster.ac.uk

**Abstract.** In this paper, we propose the application of concepts from aspect-oriented software development to facilitate modular treatment of dynamic reconfiguration descriptions in specification-to-deployment environments. Our strategy differs from earlier work in the area by blending aspects and architecture abstractions simply and seamlessly through a special kind of connector — called an *aspectual connector* — that encapsulates reconfiguration interactions. More specifically, we propose an aspect-oriented specification-to-deployment environment, called AO-Plastik, that uses our AspectualAcme ADL to specify dynamic reconfiguration by means of aspectual connectors, and maps these specifications onto a reflective component runtime platform.

**Keywords:** Dynamic reconfiguration, Aspect-oriented software development, Architecture description language, Component-based software.

## 1 Introduction

There has recently been significant research [1,3,8] on dynamic reconfiguration in environments that couple Architecture Description Languages (ADLs) with underlying runtime environments in a systematic and integrated way—so-called “specification-to-deployment” environments. Most of this research has proposed annotating conventional ADL abstractions—components, connectors, etc.—with additional reconfiguration statements. Nevertheless, dynamic reconfiguration is a *crosscutting concern*, and as such cannot be effectively modularized at the architecture level using only conventional ADL abstractions. To enable a better modularized specification of crosscutting concerns at the architecture level, Aspect-Oriented Software Development (AOSD) [4] has received increasing attention from the software architecture community. In spite of several Aspect-Oriented ADLs (AO-ADLs) being proposed [3,5], few researchers have considered dynamic reconfiguration as a crosscutting concern at the architecture level. Moreover, the approaches that do so have in common the introduction of whole new sets of ADL abstractions—usually derived

directly from AO implementation techniques—which overburden architects used to the conventional ADL abstractions.

In this paper, we propose an aspect-oriented specification-to-deployment environment called AO-Plastik, which builds on our previous work on the Plastik environment [8] and on the AspectualAcme ADL [5]. The first Plastik release supports dynamic reconfiguration both at the architecture and runtime levels, which were causally connected. At the architecture level Plastik used the original Acme ADL [6] extended with reconfiguration statements such as ‘*on-do*’ clauses. In AO-Plastik, we provide a more modular way of specifying such statements by employing AspectualAcme, a general-purpose ADL that extends the Acme’s metamodel with a special kind of connector—an *aspectual connector (AC)*—for representing crosscutting interactions. AC is used in AO-Plastik for encapsulating reconfiguration interactions. At the runtime level Plastik employed the OpenCOM reflective component runtime [2] extended with services such as configuration management, style enforcement, and reconfiguration transactions and notifications. In the AO-Plastik environment, we add to the configuration management service (CMS) a set of facilities for mapping aspectual connector specifications onto OpenCOM reflective primitives.

## 2 AO-Plastik

### 2.1 The Architecture Level

AO-Plastik system specifications must follow an *AOPlastikMF* style in order to support the mapping of AO-Plastik architectural elements onto the runtime level. This style packages one port type (*BasePort*) and one component type (*BaseComponent*). An architectural rule is defined on instances of *BaseComponent* so that they are required to have one port of type *BasePort*.

Reconfiguration statements in AO-Plastik are always defined over ports of type *BasePort*. The semantic of this port type is that it exposes the internal structure of its enclosing component; therefore, any component on which architectural reconfigurations may be effected must be an instance of *BaseComponent*. This is particularly useful for representing reconfigurations on composite components. An Acme system in AO-Plastik is regarded as a special case of outermost composite which implicitly offers a port *basePort* (of type *BasePort*). Reconfiguration is always triggered by conditions (specified by the *on-do* clause) occurring within an instance of *BaseComponent* (or the system itself). The aspectual connector encapsulates this reconfiguration protocol by associating the conditions with actions specified in a (aspectual) component to which this connector is attached through its crosscutting role. Such actions—resembling ‘*advices*’ in AOSD parlance—are specified in AspectualAcme through a new clause introduced by AO-Plastik: the ‘*action*’ clause.

Fig.1 illustrates the use of AOPlastik in the specification of a client-server system. In this example, the aspectual connector *ReconfConn* has its base role attached to the system’s implicit port *basePort*. The reconfiguration protocol in this connector, as indicated in its *glue* clause, is responsible for triggering an action (*changeServer*) in component *Reconfigurer* after the condition happens. As Fig.1 shows, two new functions are also defined in AO-Plastik for handling architectural elements involved in a reconfiguration protocol. The function *BaseElement()* is usually applied to the ‘on’

part of the *on-do* clause. It receives as its only argument a base role and returns the architectural element—either an instance of component type *BaseComponent* or a system—to which this role is attached. The function *AttachedPort()* is usually applied to the ‘do’ part of the *on-do* clause. It receives as its only argument a crosscutting role and returns the port of the aspectual component to which this role is attached. This port will be typically the place where the *action* clause will be defined. In Fig.1 the port returned by *AttachedPort()* is used for triggering the action *changeServer* defined in the *changerPort* port of the *Reconfigurer* component.

```

System ClientServer = new ClientServerFam, AOPlastikMF extended with {
Component Client = new ClientT;
Component PrimServer = new ServerT extended with {
Property failure: boolean = false; }
Connector Conn = { Role requestor; Role servicer; }
AspectualConnector ReconfConn = {
BaseRole triggerRole;
CrosscuttingRole changerRole;
Glue after = {
On (exists cp: Component in BaseElement(triggerRole).Components |
exists cn: Connector in BaseElement(triggerRole).Connectors |
attached(cn, cp) and declaresType(cp, "ServerT") and
cp.failure == true)
Do { AttachedPort(changerRole).changeServer(cp, cn); }
}
} //end of ReconfConn
Component Reconfigurer = {
Port changerPort = new ProvidedPort extended with {
Action changeServer(cp: Component, cn: Connector) = {
Detach cn.servicer from cp.service;
Remove cp;
Component BackupServer = new ServerT extended with {
Dependencies { Attachments { cn.servicer to BackupServer.service; } } }
}
} //end of Reconfigurer
Attachments { Client.request to Conn.requestor;
Conn.servicer to PrimServer.service;
Reconfigurer.changerPort to ReconfConn.changerRole;
ReconfConn.triggerRole to self.basePort; }
} //end of ClientServer

```

Fig. 1. A dependable client-server description in AO-Plastik

The modularization of reconfiguration statements into aspectual components—and the separation between aspectual components and aspectual connectors that localize the reconfiguration protocols—introduce an additional level of architectural flexibility as they allow the same aspectual component to define reconfiguration statements that may act over different systems, according to different conditions expressed in different aspectual connectors. Thus, *AO-Plastik promotes better modularized reconfiguration specifications, and better reuse.*

## 2.2 The Runtime Level

OpenCOM is originally a non-AO component runtime. A suite of AO extensions is therefore needed to support the mapping from the AspectualAcme abstractions to this runtime. We propose AO extensions that are compliant to the generic ‘AO middleware reference architecture’ defined by the AOSD-Europe consortia [7]. In the reference

architecture, aspects and advices are roles played in a non-AO middleware by components and interface operations, respectively. This is directly in line with the AO extensions available in AspectualAcme, which implies that the mapping is therefore trivial. The reference architecture also defines a new kind of binding, the *AO binding*, which supports the composition between aspectual components and regular components, similarly to aspectual connectors and attachments of AspectualAcme. In other words, aspectual connectors and attachments are mapped to AO bindings.

The reference architecture also states that aspects in a non-AO middleware should be provided by an *aspect management functionality* built on top of a reflective meta-model layer. We added this functionality to the configuration management service (CMS) in the original Plastik environment. In AO-Plastik, this service is also in charge of parsing action clauses and attachments involving aspectual connectors and inserting interceptors at the interfaces these attachments comprise. This allows CMS to receive notifications from these interfaces, which can trigger reconfiguration actions. Since causality is kept in the mapping, *AO-Plastik provides better alignment between the architecture and runtime levels*, thereby promoting on-line traceability.

### 3 Final Remarks

So far we have successfully trialed key aspects of AO-Plastik design, including the runtime support and the parsing of AspectualAcme specifications. Some planned future work includes: (i) handling conflicts or accommodating several reconfiguration strategies in AO-Plastik systems by means of ‘*precedence*’ and ‘*xor*’ relationships [5] between aspectual connectors and crosscutting ports, and (ii) packaging family-oriented reconfiguration strategies that can promote an even higher level of reuse.

### References

1. Batista, T., Joolia, A., Coulson, G.: Managing Dynamic Reconfiguration in Component-Based Systems. In: Morrison, R., Oquendo, F. (eds.) EWSA 2005. LNCS, vol. 3527, pp. 1–17. Springer, Heidelberg (2005)
2. Coulson, G., Blair, G., Grace, P., Taiani, F., Joolia, A., Lee, K., Ueyama, J., Sivaharan, T.: A Generic Component Model for Building Systems Software. ACM Transactions on Computer Systems 26(1) (February 2008)
3. Costa, C., Ali, N., Pérez, J., Carsí, J.A., Ramos, I.: Dynamic Reconfiguration of Software Architectures Through Aspects. In: Oquendo, F. (ed.) ECSA 2007. LNCS, vol. 4758, pp. 279–283. Springer, Heidelberg (2007)
4. Filman, R., Elrad, T., Clarke, S., Aksit, M.: Aspect-Oriented Software Development. Addison-Wesley, Boston (2005)
5. Garcia, A., Chavez, C., Batista, T., Santanna, C., Kulesza, U., Rashid, A., Lucena, C.: On the Modular Representation of Architectural Aspects. In: Gruhn, V., Oquendo, F. (eds.) EWSA 2006. LNCS, vol. 4344, pp. 82–97. Springer, Heidelberg (2006)
6. Garlan, D., Monroe, R., Wile, D.: Acme: An Architecture Description Interchange Language. In: CASCON 1997, Toronto, Canada, November 1997, pp. 169–183 (1997)
7. Greenwood, P., et al.: Validation of the Reference Architecture. AOSD-Europe Deliverable D-68, Lancaster University, pp. 1–38 (March 2007)
8. Joolia, A., Batista, T., Coulson, G., Gomes, A.T.A.: Mapping ADL Specifications to an Efficient and Reconfigurable Runtime Component Platform. In: WICSA 2005, Pittsburgh, USA (November 2005)