Reconfiguration Mechanisms for Service Coordination*

Nuno Oliveira and Luís S. Barbosa
HASLab/INESC TEC & Universidade do Minho
{nunooliveira, lsb}@di.uminho.pt

Abstract. Models for exogenous coordination provide powerful glue-code, in the form of software connectors, to express interaction protocols between services in distributed applications. Connector reconfiguration mechanisms play, in this setting, a major role to deal with change and adaptation of interaction protocols. This paper introduces a model for connector reconfiguration, based on a collection of primitives as well as a language to specify connectors and their reconfigurations.

1 Introduction

The purpose of a service-oriented architecture (SOA) is to address requirements of loosely coupled and protocol-independent distributed systems, where software resources are packaged as self-contained services providing well-defined functionality through publicly available interfaces. The architecture describes their interaction, ensuring, at the same time, that each of them executes independently of the context or internal state of the others.

Over the years a multitude of technologies and standards have been proposed for describing and orchestrating web services, publish and discover their interfaces and enforce certain levels of security and QoS parameters. Either to respond to sudden and significative changes in context or performance levels, or simply to adapt to evolving requirements, some degree of adaptability or reconfigurability is typically required from a service-oriented architecture. By a (dynamic) reconfiguration we mean a process of adapting the architectural current configuration, once the system is deployed and without stopping it, so that it may evolve according to some (emergent) requirements or change of context.

Reconfigurations applied to a SOA may be regarded from two different point of views. From one of them, they target individual services. In particular, such reconfigurations are concerned with dynamic update of services, substitution of a service by another with compatible interfaces (but not necessarily the

* This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-010047. The first author is also supported by an Individual Doctoral Grant with reference number SFRH/BD/71475/2010.
same behaviour) or even their plain removal. Such reconfigurations are usually triggered by external stimulus [19][13][17][18][23]. Form another point of view, a reconfiguration is entirely decided by the system itself and targets the way components or services interact with each other, as well as the internal QoS levels measured along such interactions. In particular, such reconfigurations deal with substitution, addition or removal of communications channels, moving communication interfaces from a service to another or rearranging a complex interaction structure.

This paper studies reconfiguration mechanisms for the service interaction layer in SOA. Adopting a coordination-based view of interaction [20], the model proposed here represents the ‘gluing-code’ by a graph of channels whose nodes represent interaction points and edges are labelled with channel identifiers and types. A channel abstracts a point-to-point communication device with a unique identifier, a specified behaviour and two ends. It allows for data flow by accepting data on a source end and dispensing it from a sink end. We call such a graph a coordination pattern. A subset of its nodes are intended to be plugged to concrete services, forming the pattern interface.

To keep things concrete, we assume channels in a coordination pattern are described in a specific coordination model, that of Reo [3][2]. Actually, this choice is not essential: the reconfiguration mechanisms are directly defined over the graph and concern only its topology. Only when one intends to reason about the system’s behaviour or compare the behavioural effect of a reconfiguration, does the specific semantics of the underlying coordination model become relevant. Such is not addressed, however, in this paper.

Coordination patterns are introduced in Section 4 and instantiated in the context of the Reo coordination model. Section 3 discusses reconfigurations, formally defining a collection of primitives. It is shown how the latter can be combined to yield ‘big-step’ reconfiguration patterns which manipulate significative parts of a pattern structure. The CooPLa language is introduced in Section 4 as an executable notation for specifying both coordination and reconfiguration patterns. Reconfiguration mechanisms are illustrated through a detailed example in Section 5. Section 6 concludes the paper.

2 Coordination Patterns

A pattern is an effective, easy to learn, repeatable and proven method that may be applied recurrently to solve common problems [10]. They are common in several domains of Software Engineering, namely in SOA [22] and business process [25].

Similarly, in this paper, a coordination pattern encodes a reusable solution for an architectural (coordination) problem in the form of a specific sort of interaction between the system constituents. A solution for an architectural problem is, therefore, the description of interaction properly designed to meet a set of requirements or constraints. It is reflected in a coordination protocol, which acts as glue-code for the components or services interacting within the system.