Chapter 9

Distributed Directories of Web Services

Michael Schumacher, Alexandre de Oliveira e Sousa, Ion Constantinescu, Tim van Pelt and Boi Faltings

9.1 Introduction

This chapter presents WSDir, the federated directory system used in CASCOM. Its main functionality is to let heterogeneous Semantic Web Service descriptions be registered and searched by certain clients. As such, it realizes a lookup function with basic retrieval schemes.

There are several main requirements for a distributed directory system. First, it should be easy to invoke by any client. This led us to define a Web Service interface to WSDir: it is a universally accepted standard, it provides a well-defined method to use the directory, and it allows for interacting with a heterogeneous set of clients. The sole requirement on the part of the client is that it should be able to communicate over a Web Service interface. Second, the nature of the applications to be realized requires the directory system to be distributed, for instance applying a geographical specialization of the directories. Third, the construction of the network should induce minimal overhead and should be scalable; also, the network should be robust to changes in topology and the number of interactions with the system. Fourth, the directory should allow a great number of services to be registered, and this in a very dynamic way, including lease times.

Our system is modeled as a federation: directory services form its atomic units, and the federation emerges from the registration of directory services in other directory services. Directories are virtual clusters of service entries stored in one or more directory services. To create the topology, policies are defined on all possible operations to be called on directories. For instance, they allow for routed registration and selective access to directories.

The chapter is organized as follows. In Section 9.2, we explain the service
entries of Semantic Web Services that can be stored in WSDir. Sections 9.3 to 9.6 explain the architecture of WSDir by presenting directories, directory services, directory operations, and policies. In Section 9.7, we give the concrete network architecture used in CASCOM. Sections 9.8 and 9.9 discuss respectively usability and vulnerability issues of WSDir. After referring related work in Section 9.10, we conclude the chapter in Section 9.11.

9.2 Service Entries

Services are described using the Web Ontology Language for Web Services, OWL-S [4]. Internally, the directory system stores then service entries in the FIPA SL0 description language. SL0 has been chose because the whole CASCOM infrastructure is using this language for interpretability between agents. Furthermore, as we achieve with SL0 an independent way to store any kind of services and not only OWL-S descriptions.

The internal service representation contains a subset of the information provided in the original service description. This information can be used to find matching services in the directory. In addition, the original service description in OWL-S is stored in a separate slot. This field is used to retrieve the original description, e.g., to retrieve the grounding(s) of a service at service execution.

In the following, we present the information that a service entry in WSDir contains:

**ServiceCategories** Refers to an entry in some ontology or taxonomy of services. The value of the property is a set containing elements of the class ServiceCategory, which is defined in the OWL-S ontology. The information is ultimately derived from the service categories defined in the service profile.

**ServiceProfileURIs** This slot contains a set of profile URIs that is referred to in the service description. If the profiles are included in the service description as full-text, no URIs are stored. The URIs point to an externally stored, but (web-)retrievable service profile.

**ServiceProcessURI** A process URI that is defined in the service description. If none is included for the service description, the slot will be empty. If the process is included in the service description as full-text, no URI is stored. The process URI points to an externally stored, retrievable service process.

**ServiceGroundings** The slot that contains a set of full-text service groundings for the service. Empty set if no grounding is associated with the service (abstract service). Makes it possible to retrieve only service groundings.

**OWLSServiceDescription** The slot that contains the original OWL-S service description as a full-text entry. Service profile(s) and process may be referred to as URIs, though service groundings must be included as full-text.