Chapter 4.2
Runtime Service Discovery for Grid Applications

James Dooley
City University, UK

Andrea Zisman
City University, UK

George Spanoudakis
City University, UK

ABSTRACT

This chapter describes a framework to support runtime service discovery for Grid applications based on service discovery queries in both push and pull modes of query execution. The framework supports six different types of trigger conditions that may prompt service replacement during run-time of grid business application, and evaluates the relevance of a set of candidate services against service discovery queries. The chapter also describes the language used to express service discovery queries and the three types of fitness measurement used to evaluate the candidate services against these queries. Both synchronous (pull) and asynchronous (push) mechanisms for service discovery are presented and shown to be complimentary in dealing with all six service discovery trigger conditions. The chapter is illustrated through examples.

INTRODUCTION

Traditionally, grid computing has been a form of distributed computing that is concerned with resource sharing across communications networks. In this model, individual computers are called nodes and virtualize certain resources such as processor, memory and storage. Other nodes can then access these virtual resources over a network connection. These nodes are physically distributed and are usually under the control and ownership of different entities. Examples of controlling entities are: governments, universities, corporations, and businesses. Grids implement models of virtualisation and rely on standard protocols for communication, both of which are captured in a software layer known as middleware.

DOI: 10.4018/978-1-4666-0879-5.ch4.2

More recently, Service Oriented Architecture (SOA) has emerged from the IT world in the form of web-services. SOA is an approach to enable a set of loosely coupled functional components to exist and be remotely usable. As opposed to grid computing, the emphasis of SOA is more on how to exchange information which is well defined whilst still being of open standard (as opposed to grid computing that places emphasise on the infrastructure). Foster and Tuecke (2005) capture the description of a service as *"A service is a self-contained implementation of some function(s) with a well defined interface specifying the message exchange patterns used to interact with the function(s)."*

The web-services incarnation, allows a service to exist as a Uniform Resource Locator (URL) to which requests can be sent and responses solicited in both synchronous and asynchronous modes. Over the following four paragraphs we account for some approaches that have been recently proposed to support different areas of SOA. The approaches discussed are concerned with (a) languages to describe services, (b) service discovery, (c) service composition, and (d) service monitoring, validation, verification, and evolution.

Various XML-based languages have been proposed to support descriptions of service components and choreography. Web Services Description Language - WSDL (Christensen et al., 2001) is currently the most used language for service description and supports interface-based definition in terms of input/output signatures. The semantic markup for Web Services - OWL-S (Martin et al., 2004) is an ontology that exists within the web ontology language (OWL) and extends input/output signatures by allowing the description of pre- and post-conditions to represent value-based restrictions, while Web Services Modelling Ontology - WSMO (Bruijn et al., 2005) provides a language for describing semantic aspects of services. The Business Process Execution Language for Web Services - BPEL4WS (Andrews et al., 2003) is a language that describes observable behaviour of web services by message-flow oriented interface. The Web Service Conversation Language - WSCL, (Banerji et al., 2002) goes beyond description of input/output messages, and defines transitions with associated conditions. More recently, OpenModel Modelling Language - OMML (Hall & Zisman, 2004b) has been proposed to support full behaviour specification of computer-based services.

Different approaches have been proposed to support service discovery. Some semantic match-making approaches have been suggested to support service discovery based on logic reasoning of terminological concept relations represented as ontology's. METEOR-S (Aggarwal et al., 2004) adopts a constraint driven discovery approach in which queries are integrated into a system composition process. In (Horrocks et al., 2003) the discovery of services is addressed by matching queries specified as a variant of description logic with services specified in OWL-S. The work in (Klusch, 2006) extends existing approaches by supporting explicit and implicit semantic by using logic based, approximate matching, and information retrieval techniques. The discovery mechanism in (Hausmann et al., 2004) is based on the use of Resource Description Framework Data Query Language (RDQL) by testing sub-graph relations and establishing whether specification matching relation holds between the query and the service description. The approach in (Horrocks et al., 2003) identifies services that satisfy task requirement properties expressed in temporal logic by using a lightweight reasoning tool. A flexible and modular Web Service Discovery Architecture (WSDA) was introduced in (Hoschek, 2002) where interface queries are checked based on string matching and cannot account for changes in the order of the parameters. In addition, the service selection problem has been proposed as multi-dimensional (Wang et al., 2006) or multi-ontology (Oundhakar et al., 2005) based where matching is performed against more than one service quality. Such qualities include functional, non-functional and quality of service information.