A Flexible Directory Query Language for the Efficient Processing of Service Composition Queries

Walter Binder¹, University of Lugano, Switzerland
Ion Constantinescu, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
Boi Faltings, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

ABSTRACT

Service composition is a major challenge in an open environment populated by large numbers of heterogeneous services. In such a setting, the efficient interaction of directory-based service discovery with service composition engines is crucial. In this article we present a Java-based directory that offers special functionality enabling effective service composition. In order to optimize the interaction of the directory with different service composition algorithms exploiting application-specific heuristics, the directory supports user-defined selection and ranking functions written in a declarative query language. Inside the directory queries are transformed and compiled to JVM bytecode which is dynamically linked into the directory. The compiled query enables a best-first search of matching directory entries, efficiently pruning the search space.

Keywords: indexing techniques; efficient query processing; service directories; service discovery and composition

INTRODUCTION

There is a good body of work which addresses the service composition problem with planning techniques based either on theorem proving (e.g., Golog (McIlraith & Son, 2002), SWORD (Ponnekanti & Fox, 2002)) or on hierarchical task planning (e.g., SHOP-2 (Wu, Parsia, Sirin, Hendler, & Nau, 2003)). All these approaches assume that the relevant service descriptions are initially loaded into the reasoning engine and that no discovery is performed during composition.

However, due to the large number of services and to the loose coupling between service providers and consumers, services are indexed in directories. Consequently, planning algorithms must be adapted to a situation where planning operators are not known a priori, but have to be retrieved through queries to these directories. (Lassila & Dixit, 2004) addressed
the problem of interleaving discovery and composition, but they considered only simple workflows where services had one input and one output.

Our approach to automated service composition is based on matching input and output parameters of services using type information in order to constrain the ways how services may be composed. Our composition algorithm allows for partially matching types and handles them by introducing switches in the composition plan. Experimental results show that using partial matches significantly decreases the failure rate compared with a composition algorithm that supports only complete matches (Constantinescu, Faltings, & Binder, 2004d).

We have developed a directory service with specific features to ease service composition. Queries may not only search for complete matches, but may also retrieve partially matching directory entries (Constantinescu, & Faltings, 2003). As the number of (partially) matching entries may be large, the directory supports incremental retrieval of the results of a query. This is achieved through sessions, during which a client issues queries and retrieves the results in chunks of limited size (Constantinescu, Binder, & Faltings, 2004a).

As in a large-scale directory the number of (partially) matching results for a query may be very high, it is crucial to order the result set within the directory according to heuristics and to transfer first the better matches to the client. If the heuristics work well, only a small part of the possibly large result set has to be transferred, thus saving network bandwidth and boosting the performance of a directory client that executes a service composition algorithm (the results are returned incrementally, once a result fulfills the client’s requirements, no further results need to be transmitted). However, the heuristics depend on the concrete composition algorithm. For each service composition algorithm (e.g., forward chaining, backward chaining, etc.), a different heuristic may be better adapted. Because research on service composition is still in the beginning and the directory cannot anticipate the needs of all possible service composition algorithms, our directory supports user-defined selection and ranking heuristics expressed in a declarative query language. The support for application-specific heuristics significantly increases the flexibility of our directory, as the client is able to tailor the processing of directory queries. For efficient execution, the queries are dynamically transformed and compiled by the directory.

As the main contributions of this article, we show how our directory supports user-defined selection and ranking heuristics. We present a dedicated query language and explain how queries are processed by the directory. In a first step, the directory transforms queries in order to better exploit the internal directory organization during the search. This allows a best-first search that generates the result set in a lazy way, reducing response time and workload within the directory. In a second step, the query is compiled in order to speed up the directory search. Compared with previous work (Binder, Constantinescu, & Faltings, 2004; Constantinescu, Binder, & Faltings, 2004b; Constantinescu, Binder, & Faltings, 2005), the novel, original contributions of this article are the declarative directory query language, the transformation mechanism to make better use of the internal directory structure, and the compilation of queries to JVM bytecode, which is dynamically linked into the directory. These techniques, which have not been applied in the context of service directories before, provide a flexible and efficient mechanism for query processing.

This article is structured as follows: the second section reviews some related work concerning service discovery and composition. An overview of our service description formalism and of the internal index structure of our directory is given in the third section. In the fourth section we present a simple, functional query language which allows us to express application-specific selection and ranking heuristics. The fifth section explains the processing of directory queries and introduces query transformations that enable a best-first
Early Capacity Testing of an Enterprise Service Bus
[www.igi-global.com/article/early-capacity-testing-enterprise-service/34106?camid=4v1a](www.igi-global.com/article/early-capacity-testing-enterprise-service/34106?camid=4v1a)

A Spanning Tree Based Approach to Identifying Web Services
[www.igi-global.com/article/spanning-tree-based-approach-identifying/3034?camid=4v1a](www.igi-global.com/article/spanning-tree-based-approach-identifying/3034?camid=4v1a)