Intelligent Web Based on Mathematic Theory
Case Study: Service Composition Validation via Distributed Compiler and Graph Theory

Ahmad Karawash, Hamid Mcheick, and Mohamed Dbouk

Abstract. This paper discusses a model for verifying service composition by building a distributed semi-compiler of service process. In this talk, we introduce a technique that solves the service composition problems such as infinite loops, deadlock, and replicate use of the service. Specifically, the client needs to build a composite service by invoking other services but without knowing the exact design of these loosely coupled services. The proposed Distributed Global Service Compiler, by this article, results dynamically from the business process of each service. As a normal compiler cannot detect loops, we apply a graph theory algorithm, a Depth First Search, on the deduced result taken from business process files.

Keywords: SOA (Service Oriented Architecture), Compiler, Business Process Execution Language (BPEL), Depth First Search (DFS), Distributed Global Service Compiler (DGSC).

1 Introduction

Web services are defined as self-contained, modular units of application logic which provide business functionality to other applications via an Internet connection. Web services support the interaction of business partners and their processes...
by providing a stateless model of atomic synchronous or asynchronous message exchanges (B. Srivastava and J. Koehler). Every service consists of a domain of computers and computers in a domain can just communicate with each other through predefined functions (Amirjavid F. et al., 2011). Services can be invoked by other services or applications. Services are designed for interaction in a loosely coupled environment, and therefore are an ideal choice for companies seeking inter or intra business interactions that span heterogeneous platforms and systems (K. LI, 2005).

In many cases, a single service is not sufficient to the user's request and often services should be combined through service composition to achieve a specific goal. Nowadays, researches show that problems of composing web services are expressed in designing, discovery, validation and optimization of service composition. The paper proposes a way of dynamic validation of service composition to prevent some errors (infinite loops, blocked services, incorrect business flow design and many others) at the beginning of the design phase of new composite service.

Before designing a new composite service, the service discovery process returns a set of candidate services and at some of those services are used in the composition process according to non-functional criteria (cost, time, and context-aware). But nothing, in this discovery notifies service developer if the invoked services are working normally or not.

In the dynamic world of service-oriented architectures, however, what is sure at design time, unfortunately, may not be true at run time. The actual services, to which the workflow is bound may change dynamically perhaps in an unexpected way, may cause the implemented composition to deviate from the assumptions made at design time. Traditional approaches, which limit validation to being a design time activity, are no longer valid in this dynamic setting. Besides performing design-time validation, it is also necessary to perform continuous run-time validation to ensure that the required properties are maintained by the operating system. The compiler is the only way to validate the sequence of service process. It is the program that translates one language to another. Thus our goal is to implement a distributed dynamic compiler that compiles the composition of every new composite service. When a client designs a new composite service, the related compiler Grammar rules, of the invoked services, are sent to him as XML files then combined together to constitutes a local compiler that validate new service composition at design phase.

Section 2 describes the previous methods of service validation. Section 3 gives two service composition examples to highlight the problem of service validation. While section 4 proposes a new service composition validation model in the service design development phase; two techniques of parsing and depth first search are described and a simulation steps are given in this section. Section 5 summarizes the ideas as a conclusion and gives perspectives for future works.