Chapter 18
XML Query Evaluation in Validation and Monitoring of Web Service Interface Contracts

Sylvain Hallé
University of California, Santa Barbara, USA

Roger Villemaire
Université du Québec à Montréal, Canada

ABSTRACT

Web service interface contracts define constraints on the patterns of XML messages exchanged between cooperating peers. The authors provide a translation between Linear Temporal Logic (LTL) and a subset of the XML Query Language XQuery, and show that an efficient validation of LTL formulae can be achieved through the evaluation of XQuery expressions on message traces. Moreover, the runtime monitoring of interface constraints is possible by feeding the trace of messages to a streaming XQuery processor. This shows how advanced XML query processing technologies can be leveraged to perform trace validation and runtime monitoring in web service production environments.

INTRODUCTION

Service-oriented architectures (SOA) have become an important concept in systems development with the advent of web services (Alonso et al., 2004). Because of their flexible nature, web services can be dynamically discovered and orchestrated together to form value-added composite applications. However, this appealing modularity is also the source of one major issue: by dynamically combining heterogeneous, cross-business services, how can one ensure the interaction between each of them proceeds as was intended by their respective providers? Whether for specifying performance or interoperability constraints, business policies or legal guidelines, a good web service has to have a well-defined and enforceable *interface contract* (Meredith & Bjorg, 2003).

While not strictly an XML technology, a web service depends heavily on XML for both its definition language, the Web Service Description Language (WSDL) (Christensen et al., 2001) and its messaging protocol, the Simple Object Access Protocol (SOAP). Specification languages such as DTD, RELAX NG, Schematron and XML Schema (XSD) have become natural candidates for express-

DOI: 10.4018/978-1-61520-727-5.ch018

ing so-called *data contracts*, specifying the shape of each message sent or received by a service. However, while data contracts are relatively straightforward to specify and verify, interface contracts go beyond such static requirements and also include a temporal aspect. For example, the online documentation for the popular Amazon E-Commerce Service (Amazon, 2009) elicits constraints on possible *sequences* of operations that must be fulfilled to avoid error messages and transaction failures (Hughes et al., 2008).

Compliance to interface these contracts can be checked in two different and complementary ways. First, an interaction can be recorded by some observer into a trace (or log) file, which can then be analyzed *a posteriori* to discover violations: this is called *trace validation* or *log analysis*. The same validation can also be performed on-the-fly, intercepting and analyzing messages as they are received or sent and warning of violations in real time: this is called *runtime monitoring*.

Although it is widely believed that interface contracts should, *in theory*, be verified or monitored in various ways, these tasks are often dismissed in execution environments on the grounds that: 1) they will cause an overhead sacrificing performance; 2) they require the development of new algorithms and the use of new, possibly untrusted software components.

The goal of this chapter is to challenge these two beliefs by showing how advanced XML query processing technologies can be leveraged to perform trace validation and runtime monitoring. The key idea consists in representing message exchanges as the progressive construction of a global XML "trace" document. Since contracts impose restrictions on the content and sequence of messages, validating and enforcing them simply becomes appropriately querying this document —harnessing the available computing capabilities of readily-available XML query engines in web service environments.

BACKGROUND

In layman's terms, a web service can be described as an independent software system providing its functionality through the exchange of messages. Generally, a service resides on a remote, public server accessible through the Internet using standard protocols. For example, when using the Simple Object Access Protocol (SOAP) over HTTP, communications to and from a web service are realized through the sending of XML documents to an endpoint URL. Among popular web service providers, we mention Amazon and Google. Figure 1 shows how, for example, one can send a request to a map service to obtain the geographical coordinates of a street address.

An Example: The Online Trading Company

Although some web services, like the previous example, require *stateless* request/response patterns, many web services allow for long-running, *stateful* transactions spanning multiple messages. Examples of such services abound in the literature; the reader is referred to Ghezzi & Guinea (2007), Hughes et al. (2008), Mahbub & Spanoudakis (2005), Hallé et al. (2009c), IBM (2002).

Let us consider the case of an online trading company, adapted from Josephraj (2005). A trading company is responsible for selling and buying stocks for its customers; it exposes its functionalities through a web service interface. An external buyer (which can be a human interfacing through a web portal, or another web service acting on behalf of some customer) can communicate through a set of XML messages, each representing a business operation performed by the trading company.

A client service first logs into the system by providing a user name. The shop offers a discount if a user connects with the commitment to buy at least one product, which is signalled with the commitToBuy XML element. The shop responds