Web Service Orchestration with BPEL

Liang Chen, Bruno Wassermann and Wolfgang Emmerich
London Software Systems
Dept. of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
w.emmerich@cs.ucl.ac.uk

Howard Foster
London Software Systems
Dept. of Computing
Imperial College London
180 Queen’s Gate
London SW7 2BZ, UK
hf1@doc.ic.ac.uk

Categories and Subject Descriptors
D.2.6 [Software Engineering]: Programming Environments—integrated environments; D.2.11 [Software Engineering]: Software Architectures—languages; D.2.13 [Software Engineering]: Software Reuse—reuse models

General Terms
Design, Standardization, Languages

1. INTRODUCTION

Software engineers are increasingly adopting service oriented architectures (SOAs) for the automation of business processes and the integration of IT systems, both within and across organizational boundaries. These service oriented architectures frequently rely on web service standards, such as the Web Service Description Language (WSDL) [5] and the Simple Object Access Protocol (SOAP) [10] for the implementation of service invocations across machine boundaries.

The combination of several web services into a more complex web service is a crucial building block for service oriented architectures. Peltz refers to such compositions that integrate the invocation of two or more services into a more complex executable workflow as web service orchestration and contrasts this with web service choreography, which tracks message exchanges between different autonomous domains [13]. Web service orchestration is appealing as it facilitates compositionality and reuse of the components that implement these services without necessarily having to deploy these services locally.

Web service orchestration is supported by the Business Process Execution Language for Web Services (BPEL) [1]. BPEL emerged through consolidation of earlier work on IBM’s Web Service Flow Language (WSFL) [11] and XLANG [14] developed by Microsoft. BPEL was proposed as a standard by Microsoft, IBM, Siebel, BEA

Copyright is held by the author/owner.
ICSE ’06, May 20–28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

2. RELATIONSHIP WITH OTHER STANDARDS

BPEL plays an important role in the web service standards family and is used in concert with a number of other standards. The tutorial identifies these relationships and discusses the following related standards. The Web Services Description Language (WSDL) is used to define the interface of a web service. BPEL relies on WSDL to ensure that the invocation of a service is correctly typed and also to deduce the synchronization behaviour of this invocation. As BPEL is compositional, a process defined in BPEL may expose a WSDL interface itself. This enables the invocation of the process from remote hosts through standard web service mechanisms. The fact that such a web service is a service orchestration remains entirely transparent to clients.

A web service is invoked by a BPEL interpreter using the Simple Object Access Protocol (SOAP), which defines the exchange of messages that are encoded in the eXtended Markup Language [4]. The structure of these messages, as well as the syntax of BPEL itself are defined using XML Schemas [3]. BPEL uses XPath [6] both to extract data from XML messages and assign them to internal variables and also to compose the parameters for invocations from such variables. BPEL is often used together with implementations of the Universal Directory and Discovery Interface (UDDI) [7], which is capable of registering and locating services across autonomous domains. Because orchestrations span autonomous domains, BPEL is often used jointly with WS-Security [12] for encryption of SOAP messages and proof of origin.

3. THE BPEL LANGUAGE

3.1 Basic Concepts

The tutorial then describes basic BPEL concepts along a common example. BPEL is a statically typed Turing complete scripting language. It has a standardized syntax that is defined in XML though most users use proprietary graphical notations supported by the various BPEL editors.

BPEL can maintain temporary data structures in variables. These can be queried and manipulated by a BPEL process in order to pass data from one service invocation to another. BPEL uses XML Schemas to define the type of variables and of service input and output parameters.

The visibility of declarations in BPEL is determined through

The development of this tutorial was supported by EPSRC through Grants GR/R97207/01 and GR/S90843/01 and SAP to the Organization for the Advancement of Structured Information Standards (OASIS) and is currently undergoing a major revision.
scopes, which are used in a similar way as in other modern programming languages. Scopes can then contain a number of variable declarations that are visible in that scope and a set of basic activities or control flow elements.

Basic activities include invocation of a remote service, deterministic or non-deterministic receipt of messages, assignments of XML data structures to variables, and a number of further activities, such as exception handling and creation, wait operations etc.

BPEL defines control flow elements that can be used to combine the basic activities described above. These control flow elements support conditional executions and the definition of various loops as they are known from other structured programming languages.

3.2 Advanced Concepts

BPEL has a number of advanced concepts that are not necessarily found in programming languages and we find it interesting to understand how they are used in the orchestration of services. BPEL defines the notion of concurrent flows which support an interleaved model of concurrency and causes the need for primitives to achieve mutual exclusion when shared variables are being accessed in these flows. We discuss how BPEL uses serializable scopes to achieve such mutual exclusion.

We will discuss asynchronous invocations, which rely on the delegation of message correlation to the BPEL engine in order to identify the receiver of the reply to an asynchronous invocation.

We show how compensation handlers are used to handle failures and achieve atomicity of long-running transactions.

We discuss persistence of process state that is essential when long-running transactions need to survive accidental or deliberate termination of a BPEL engine.

4. ENGINEERING ORCHESTRATIONS

In our experience if web services are designed carefully and with orchestration in mind it becomes considerably easier to combine them into composite services with BPEL. We describe this experience and delineate engineering guidelines with respect to where and how message types should be defined, approaches to unit testing web services and their composition. We discuss the implications potentially concurrent invocation from BPEL on thread safety of web service implementations. We conclude by describing guidelines for achieving scalable and fault-tolerant orchestrations.

5. BPEL TOOL SUPPORT

We present an overview of currently available BPEL tools. These tools can be classified by their support for modelling, validating, enacting and monitoring BPEL workflows.

We will discuss commercial products available from Parasoft, Oracle, Active Endpoints, IBM and Microsoft and also demonstrate BPEL tools that are available under open source licenses, most notably the ActiveBPEL engine and the Eclipse BPEL designer.

6. RESEARCH QUESTIONS

We summarize the tutorial content by highlighting which role BPEL plays in software architectures that span across organisational boundaries. In particular, BPEL allows the reuse and composition of complex services from existing services that are deployed and hosted in different organisational domains.

While the definition of BPEL was very much an industry effort, a software engineering research community has emerged that investigates how to systematically model and reason about BPEL orchestrations. We conclude the tutorial by giving a brief overview of promising approaches, such as model checking BPEL orchestrations [9, 8] and monitoring the quality of service provided by orchestrated web services [2].

We then discuss a number of research questions that arise from the emergence of BPEL. We discuss the need for not just a functional integration between components but also a systematic treatment of compositionality of service quality and we highlight an important problem in the area of model checking BPEL orchestrations under given resource constraints.

7. REFERENCES


