The Journal of Systems and Software 99 (2015) 135–154

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Bringing Test-Driven Development to web service choreographies

Felipe Besson[a,*], Paulo Moura[a], Fabio Kon[a], Dejan Milojicic[b]

[a] Department of Computer Science, University of São Paulo, Brazil
[b] Hewlett Packard Laboratories, Palo Alto, USA

ARTICLE INFO

Article history:
Received 17 April 2013
Received in revised form
23 September 2014
Accepted 23 September 2014
Available online 7 October 2014

Keywords:
Automated testing
Test-Driven Development
Web service choreographies

ABSTRACT

Choreographies are a distributed approach for composing web services. Compared to orchestrations, which use a centralized scheme for distributed service management, the interaction among the choreographed services is collaborative with decentralized coordination. Despite the advantages, choreography development, including the testing activities, has not yet evolved sufficiently to support the complexity of the large distributed systems. This substantially impacts the robustness of the products and overall adoption of choreographies. The goal of the research described in this paper is to support the Test-Driven Development (TDD) of choreographies to facilitate the construction of reliable, decentralized distributed systems. To achieve that, we present Rehearsal, a framework supporting the automated testing of choreographies at development-time. In addition, we present a choreography development methodology that guides the developer on applying TDD using Rehearsal. To assess the framework and the methodology, we conducted an exploratory study with developers, whose result was that Rehearsal was considered very helpful for the application of TDD and that the methodology helped the development of robust choreographies.

1. Introduction

Service-Oriented Architecture (SOA) is a set of principles and methods that uses services as the building blocks for the development of distributed applications. Web services can be composed to create more complete services and then to implement complex business workflows. Different from orchestrations, choreographies are a more scalable approach for composing services (Guimaraes et al., 2012). The interaction among the choreographed services is collaborative: coordination is distributed across all choreography participants. Locally, each choreography participant is only concerned with the actions it must take to play the desired role. For this reason, choreographies are a good candidate architecture for fully decentralized workflows (Barker et al., 2009).

Over the years, the way information is created, shared, used, and integrated in the Internet is changing at a fast pace. As a result, the current Internet is evolving towards the Future Internet, which is foreseen as a federation of services that will provide built-in mechanisms such as scalable service access, mobility of network and devices, in a secure, reliable, and robust way (Tselentis et al., 2010). With respect to service access, service choreographies are proposed as an adequate architecture to deal with the large scale nature of the Future Internet, which can be translated into a high number of interacting entities, parallel single service accesses, and massive service load (Issarny et al., 2011).

A few standards have been proposed for modeling choreographies, such as the Web Services Choreography Description Language (WS-CDL), a W3C standard candidate proposed in 2004, and, more recently, the OMG Business Process Model and Notation version 2 (BPMN2). However, up to now, none of them have experienced wide adoption. This is also true for development methods such as the one proposed by the Savara project (Madurai, 2009). Due to the inherent characteristics of SOA – such as dynamism, inter-organization integration, reuse of service – the automated, and even the manual, testing of choreographies is not conducted properly. This scenario results in choreographies implemented using *ad hoc*, often chaotic, development processes. As a consequence, choreography development activities cannot be performed properly. Normally, neither the functional behavior nor scalability of choreographies is verified or assessed properly.

The goal of our research is to apply Test-Driven Development (TDD) to choreographies to facilitate their construction and improve their adoption. TDD is a design technique that guides the development of software through testing (Beck, 2003; Fowler, 2011). With TDD, test cases are written before the code which they test, in a technique called *test-first programming*, which forces the developer to think about what could possibly go wrong even before

* Corresponding author. Tel.: +55 1130916135.
E-mail addresses: besson@ime.usp.br (F. Besson), pbmoura@ime.usp.br (P. Moura), kon@ime.usp.br (F. Kon), dejan.milojicic@hp.com (D. Milojicic).

http://dx.doi.org/10.1016/j.jss.2014.09.034
0164-1212/

the implementation itself begins (Erdogmus et al., 2005). Experiments and empirical observations in the industry have shown that TDD increases code quality and reduces defect density. A case study (Bhat and Nagappan, 2006) conducted in Microsoft assessed the impact of TDD in two different teams. In the first one, although the initial development time of the project using TDD increased by 35%, the density of defects decreased by 62%. In the second case, the initial development time increased 15% while the density of defects decreased 76%. Based on this favorable retrospect, our research hypothesis was that TDD has a good potential to aggregate more quality to choreography development.

This paper presents two major contributions:

1. Rehearsal, a novel framework for automated offline (development-time) testing of web service choreographies.
2. A choreography development methodology, which guides the developer to use the framework with TDD.

Both contributions were evaluated empirically via an exploratory study following sound Empirical Software Engineering principles.

Our research involved a three-year study on the requirements and architecture for provisioning a powerful tool and useful methodology for the development of robust choreographies. During this period, we carried out a comprehensive study of the academic literature on the subject, a detailed analysis of related software tools, and discussed the subject with tens of programmers, researchers, and practitioners from the software industry. Our research findings indicate that the tool and the methodology, as described in this paper, aid developers significantly in the complex task of building large-scale decentralized systems composed of collections of web services.

Differently from the related works, Rehearsal and the development methodology provide features and guidelines for testing and developing choreographies following agile software development concepts. Therefore, with this framework, the developer can write tests before the service implementation. Following the agile culture, the tests are created by developers based on the choreography specification. This process is performed incrementally, and the tests guide the design of the choreographed services. During the service integration, the exchanged messages can be intercepted and validated through service proxies, which helps the developer to monitor and debug the choreography behavior at the development environment. Moreover, Rehearsal provides a feature to emulate (mock) third-party services that cannot be used during offline tests. This feature can be invoked through a fluent interface that is similar to the interface of Java mocking tools (e.g., Mockito[1] and EasyMock[2]).

This paper is structured as follows. Section 2 provides a brief background, a practical choreography example, and requirements for a comprehensive testing infrastructure to deal with real problems arising from testing choreographies. Section 3 discusses related work and its relationship with our approach. In Sections 5 and 6, we present Rehearsal and our methodology proposal and how these artifacts address the problems presented in our choreography example. To assess both artifacts, an exploratory study has been conducted with advanced Computer Science students. The study design and obtained results are presented and discussed in Section 7. Then, an industrial validation carried out in the context of CHOReOS project is summarized in Section 8. Finally, we draw our conclusions and discuss ongoing and future work in Section 9.

2. Fundamental concepts

In this section, we present a brief introduction to web service choreographies and Test-Driven Development (TDD), two fundamental concepts related to this work. Then, we present the FutureMarket, an example of a choreography for distributed shopping. This example guides the explanation of the Rehearsal features in the next sections. Finally, we present the existing challenges of choreography testing that this work aims to cover.

2.1. Web service choreographies

The ability of composing web services effectively is one of the critical requirements for service oriented computing (Erl, 2007). In this context, orchestrations and choreographies have been proposed as approaches for composing web services. Orchestrations correspond to a centralized approach where internal and external web services are composed into an executable business process (Peltz, 2003). Some standards, such as the Business Process Execution Language (BPEL),[3] have been proposed for orchestrating services. In an orchestration, a central party (node) controls the interaction flow of the other parties, unlike in choreographies, where the control is decentralized.

A choreography is a collaborative interaction in which each involved node plays a well defined role. A role defines the behavior a node must follow as part of a larger and more complex interaction. When all roles have been set up, each node is aware of when and with whom to communicate, based on pre-established messages specified by a global model (Barker et al., 2009). Therefore, when the choreography is started (enacted), there is no central entity driving the interaction of the whole choreography. In a web service choreography, each role is formally specified through a Web Service Description Language (WSDL) document.

Since orchestrations are executable processes, choreographies can be executed via distributed orchestrations (Autili and Ruscio, 2011). In this approach, the developer associates an orchestration to each choreography role. During the choreography information flow, many coordinators (orchestrations) are then responsible for different parts of the flow. In this manner, there is no entity keeping a global interaction state or, in other words, coordinating the entire business flow.

In an abstract (higher) level, the messages exchanged among the choreography roles are specified by using modeling languages (e.g., BPMN 2). However, in an executable (lower) level, a role implementation consists of an orchestration of a service or a set of services. In this case, the WSDL interface exposed by the orchestration must be compatible with the interface required by the implemented choreography role.

Fig. 1 depicts a role implementation through orchestrations. The orchestration in the left side implements the *Store* role. During the choreography execution, this orchestration sends a payment message to the *Bank* role that forwards part of the payment to the *Shipper* role. Finally, this orchestration sends a confirmation message back to the *Store* role. During the message exchange, the information flows across the choreography roles in a decentralized way.

2.2. Test-Driven Development (TDD)

Test-Driven Development (TDD) consists of a design technique that guides software development through testing (Beck, 2003;

[1] http://code.google.com/p/mockito.
[2] http://www.easymock.org.
[3] BPEL: http://www.oasis-open.org/committees/wsbpel.