An Abstract Transaction Model for Testing the Web Services Transactions

Rubén Casado, Javier Tuya
Department of Computing
University of Oviedo
Gijón, Spain
rcasado@lsi.uniovi.es, tuya@uniovi.es

Muhammad Younas
School of Technology
Oxford Brookes University
Oxford, United Kingdom
younas@brookes.ac.uk

***Abstract*—Transactions are a fundamental technology for building efficient and reliable web service based applications. Various models and protocols have been developed by academic and industrial research community in order to effectively manage web services transactions. We propose a novel abstract model for dynamically modeling distinct web services transaction protocols. Model-based testing techniques can be used on the abstract model in order to automatically generate test scenarios.**

Transactions; web services testing; model-based testing

I. INTRODUCTION

A transaction is defined as a set of operations of an application such that all the operations achieve a mutually agreed outcome. The conventional method for achieving such outcome is the enforcement of the Atomicity, Consistency, Isolation and Durability (ACID) properties. Web Services (WS) transactions are more complex as they involve multiple parties, span many organizations, and may take a long time to finish. Strictly enforcing the ACID properties is not appropriate to a loosely coupled world of WS doing the lock of resources unsuitable. In order to meet the requirements of WS, various extended transaction models have been adapted. So WS do not have a standardized transaction model such as ACID model for classical transactions. Instead there are a diversity of transaction models and protocols such as OASIS Business Transaction Protocol (BTP) or OASIS Web Services Business Activity (WS-BA). These approaches follow distinct methods and protocols in enforcing the integrity and consistency of transactions.

Although transactions have been identified as a key issue in WS environments, current research does not focus on a crucial issue of testing them. The literature presents a number of approaches to testing WS applications such as testing service integration, but they fall short of testing the transactional aspects of WS [1].

In this paper we propose an abstract transaction model that serves as a template for modeling and testing the WS transactions in terms of reliability to failures. The model automatically generates specific test scenarios for various WS transaction models and their processing tasks. It may adopt the standard testing techniques of transition coverage for generating test scenarios.

II. THE ABSTRACT TRANSACTION MODEL

The abstract model has been developed based on main concepts shown in the literature [2]. Its objective is to be easy to understand as well as capable to pattern the actual web service transaction models. We have used the *UML statecharts* notation since the model is event-driven (messages between participants).

A web service transaction (*wT*) is a set of activities (subtransactions) executed by different web services (participants) that can take a substantial amount of time to complete. We identify four different roles between the participants. Figure 1 shows an overview of the roles model where the behavior is captured using states. Note that each role model is broken down into a more detailed model.

- *Executor*: a participant responsible for executing and terminating a subtransaction.
- *Coordinator*: it coordinates the *wT* and manages failures and compensations. It also collects the results from the participants in order to provide system with a consistent state after the execution of *wT*.
- *Initiator*: it starts the *wT*. First it requests the coordinator for a transaction context. Then it asks to the others participants to participate in the *wT*.
- *Terminator*: it decides when and how the *wT* has to be finished. Thus it participates in the coordination tasks so it can be a subcoordinator.

There are different types of subtransaction according to the execution effect on the environment of the global transaction. A subtransaction is *lockable* if the resources that it uses can be lock until the final decision. A subtransaction is *compensable* if its execution effect can be semantically undone after commitment by executing compensation. If a subtransactions is neither lockable nor compensable is said to be *pivot*. Any compensable subtransaction has a compensation that undoes, from a semantic point of view, the actions performed by the subtransaction. A subtransaction is *retriable* if once it has failed, the participant can try to execute it again without extra risks. A subtransaction is *replaceable* if there is another participant that can execute the same task.

The modeling of WS transaction standards is achieved following these three algorithms:

- *Role identification and modeling*: it identifies the roles of participants in a target WS transaction standard and models it using the roles defined in the abstract model.
- *State transitioning*: it captures the important states of the target WS transaction standards and maps them to the state transitions of the abstract model.
- *Message syntax*: it transforms the messages from abstract transaction model to a specific standard.

Below we summarize the modeling of BTP and WS-BA protocols. The states and transitions used to pattern such models is a concrete subset of the abstract model according to their specific characteristics. The transformation of messages from the abstract transitions to both specific message syntax is automatically generated by a prototype tool.

A. Business Transaction Protocol

BTP is based on nested transactions, wherein a parent transaction is composed of subtransactions (*Superior:Inferior* relationship). This relationship can be recursively extended to defining a transaction tree where the intermediates nodes are superior and inferior at the same time. The role identification and modeling is depicted in Figure 2 where (a) shows the BTP model and (b) its mapping.

B. Web Services Business Activity

WS-BA uses a compensation-based model. The role of *initiator* is taken by the first participant who interacts with a coordinator. WS-BA define two protocols. In *MixedOutcome*, the coordinator is the *terminator* since each executor may have its own decision. In *AtomicOutcome* the role of *terminator* is taken by all the participants since everyone can cancel the transaction.

III. Model-Based Testing

The main goal of testing is failure detection i.e., the observable differences between the behaviors of implementation and what is expected. We use model-based testing since our abstract model allows us to pattern the transaction behavior. The steps used in the process of definition of test scenarios are described as follows:

- *Test criterion selection:* The transition coverage criterion defines that the set of test scenarios must include tests that cause every transition in the model to be taken.
- *Generating abstract test scenarios*: An abstract test scenario is defined as a sequence of states and transitions of a participant using the abstract model.
- *Generating specific test scenarios*: An abstract test scenario is transformed to a sequence of messages between participants using a specific WS transaction standard.

Our prototype tool automatically obtains the set of test scenarios. It applies transition coverage criterion over the abstract model and obtains a set of independent paths.

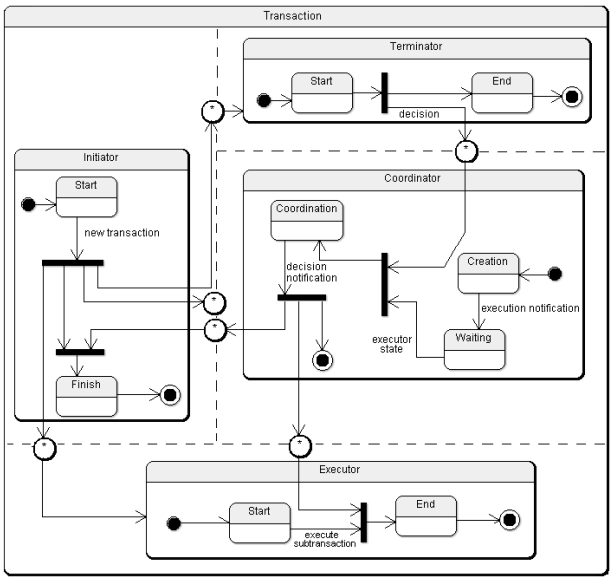

Figure 1. Transaction roles.

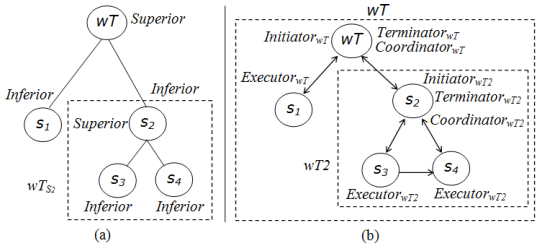

Figure 2. BTP relationships modeling

Each path defines an abstract test scenario. The tool also generates the mapping from the abstract test scenario to a specific test scenario (sequence of message using the syntax of BTP or WS-BA).

IV. Conclusions

Our work addresses the issue of modeling and testing WS transactions. It proposed a novel abstract transaction model which dynamically models them. We use model-based testing techniques in order to automatically generate test scenarios for applications that use standards such as BTP and WA-BA.

Acknowledgment

This work was partially funded by the Department of Science and Technology (Spain) and ERDF funds under the National Program for Research, Development and Innovation, project Test4DBS (TIN2010-20057-C03-01), and the grant BES-2008-004355.

References

[1] G. Canfora, and M. Penta, "Service-Oriented Architectures Testing: A Survey," ISSSE 2006-2008, Revised Tutorial Lectures, pp. 78-105: Springer-Verlag, 2009.

[2] R. Casado, J. Tuya, and M. Younas, "A Framework to Test Advanced Web Services Transactions," Proc. IEEE ICST 2011, pp. 443-446, DOI 10.1109/ICST.2011.44