A framework to test advanced web services transactions

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

Muhammad Younas
Department of Computing
Oxford Brookes University
United Kingdom
m.younas@brookes.ac.uk

***Abstract*—Transactions are a key issue in the reliability of distributed applications because they ensure all the participants achieve a mutually agreed outcome. However, current research has given little attention to testing transactions in web services. This paper presents a conceptual framework, inspired in risk-based methodologies, to address this gap. It also reports on preliminary results and identifies future work.**

Web service testing; web service transaction, SOA testing

I. Introduction

Transactions are a fundamental concept in building reliable distributed applications. A transaction is a mechanism to ensure all the participants in an application achieve a mutually agreed outcome. Traditionally, transactions have held the Atomicity, Consistency, Isolation and Durability (ACID) properties, which form one of the most important models of the distributed systems.

In Web Services (WS) environment, transactions are complex, involve multiple parties, span many organizations, and can have long duration. Strictly enforcing the ACID properties is not appropriate to a loosely coupled world of autonomous trading partners (represented through web services) due to the increased length of time that forbids the use of locks on resources, and hence makes roll-back activities unsuitable. In order to deal with these new features, various extended transaction models have been adapted for WS. These models mainly relax the strict atomicity and isolation policy of ACID properties so that intermediate results of active transactions are visible to other transactions [1].

Despite the fact that the literature presents a number of approaches and techniques on WS testing, there is a lack of research work on testing WS transactions [2]. To the best of author´s knowledge, there are no works about testing transactional requirements in WS environments. Some works are focused on verifying the long-lived transactions from a theoretical point of view. Lannotte et al. [3] developed a model of communicating hierarchical timed automata suitable to describe long-running transactions and the automaton-theoretic approach allows the verification of properties by model checking. Emmi et al. [4] use a technique to translate programs with compensations to a tree automata in order to verify the illusion of atomicity. Also Li et al. [5] proposes a formal model to verify the requirement of relaxed atomicity with temporal constraints whilst Gaaloul et al. [6] use event calculus to verify the transactional behaviour of WS compositions.

Our research studies the viability of a practical approach to test the transactional requirements in WS environments. This approach is inspired by the risks methodologies and comprises several steps: i) deep study of the process, ii) decompose the whole scope into subsystems, iii) identification of hazards in each subsystem, iv) ways to mitigate the likely faults. We have already adapted some of these steps to test web transactions.

The rest of the paper is organized as follow. Section II summarizes the general framework. The work done to date is highlighted in Section III. Section IV comments the actual and future work. Finally, conclusions are presented in Section V.

II. Conceptual Framework

The proposed conceptual framework to test WS transactions is hierarchically organized in different levels. The basic work is a thorough study and analysis of this kind of transaction characteristics. Using that knowledge, the first level presents a method to define functional transactional requirements. The second level presents the division of the process on subsystems. These subsystems, called *transaction system properties*, represent general characteristics that have to be checked in order to ensure the right functioning of a WS transaction. The next level is where a risk analysis is applied for each property in order to identify potential failures and their possible causes. This analysis is carried out by taking into account the current WS transaction standards behaviour. With that information, the fourth level studies how to apply testing techniques in order to generate test scenarios and to mitigate the identified risks. In the last level we will specify how to execute the proposed tests. Fig. 1 depicts the proposed framework.

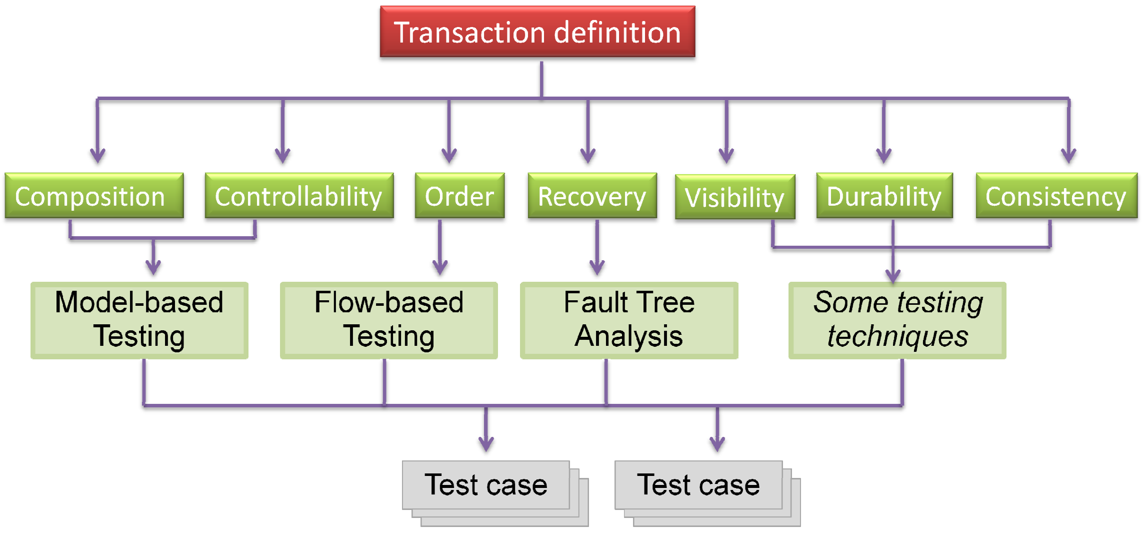

Figure 1. Conceptual framework.

III. Work Done to Date

We have studied the evolution of transaction models from ACID to current WS transactions approaches.

Review of Existing Transaction Models: A transaction is a unit of work that involves one or more resources and is either completed in its entirety or is not done at all. Participating resources are locked for the duration of the transaction. When database systems became distributed the well-known *two phase commit protocol* (2PC) [7] was proposed to ensure the ACID properties. Nevertheless the 2PC protocol cannot be completely applied in some distributed transactions where increased length of time forbids the use of locks on resources. To deal with these problems the *Advance Transaction Models* (ATM) [8], were proposed. The *Nested transaction* model [9] proposed the idea to decompose a transaction into subtransactions with independency to commit. In the *SAGA* [10] model, each subtransaction has associated a compensation subtransaction that semantically undoes the effects of its committed associate. A further analysis of web transactions properties, and also a first approach of our conceptual framework, was presented in [11].

In order to manage WS transactions, several standard specifications have been published. Business Transaction Protocol (BTP) [12] is a specific framework to manage transactions based on the 2PC but allows managing long-lived transactions using an adaptation of the Nested transaction model. Web Services Composite Application Framework (WS-CAF) [13] is a set of WS specifications for applications composed of multiple web services and Web Services Transaction Management (WS-TXM) is the one to manage transactions. WS-TXM allows both 2PC transactions and compensate-based transactions using the *SAGA* model. Both Web Service Atomic Transactions (WS-AT) [14] and Web Service Business Activity (WS-BA) [15] are built on top of Web Service Coordination (WS-COOR) [16]. WS-AT specific a classic 2PC protocol to ensure ACID properties while WS-BA coordinates long-running transactions using the *SAGA* model

.

The proposed framework: According to the conceptual framework, the first level defines a way to specify functional transactional requirements. A model and its notation were proposed in [17]. Our approach decomposes the web service transaction in a set of independent subtransactions each one with a compensation associated. For each subtransaction, we identify three different sets of functional requirements that should be checked:

- *Initial state* is the necessary requirement so that the subtransaction can be executed.
- *Executed state* defines the requirements that have to be satisfied once the subtransaction has correctly finished.
- *Compensate state* defines the requirements that have to be satisfied once the compensation has been executed.

After the deep study of *state of the art* about transactions, especially in web service environments, we have identified a set of properties that have to be tested in order to ensure the correct behaviour. The transactions systems properties are summarized in Table I. A further explanation of these properties was presented in [18].

TABLE I. SYSTEM PROPERTIES

Composition	A *web service transaction* is composed of independents subtransactions that may be executed by independent services
Order	The subtransactions have a specific order of execution. Also there are relationships between them that have to be satisfied.
Visibility	A *web service transaction* allows other (sub) transactions to see the partial results of its subtransactions.
Durability	When the transaction is finished successfully the results will remain permanent in the system
Consistency	Subtransactions or their compensating transactions must maintain the required consistency of web services
Recovery	A subtransaction can be undone executing its compensatory action. So any web service transaction can be undone reaching an initial equivalent state if all executed subtransactions are compensated.
Controllability	This requires that the coordinator has to ensure the Composition, Consistency, Durability and Recovery properties of the transactions.

In the previous work we also presented the Fault Tree Analysis (FTA) [19] for the *recovery* property and how it can be used to define test cases. Fig. 2 depicts a small part of the fault tree in order to model the risks of the WS transaction compensatory mechanism, which include: the compensatory action is not executed at all (2) or incorrectly executed (3) or it fails due to the loss of messages between the participants and the coordinator (4). This could be due to problems with participant messages (5) or coordinator messages (6). The problems related to participant are that it does not receive the compensation message (7), it receives the message when it should not receive this message (8) or the compensate message has finished with timeout (9).

When a participant has problems receiving compensate messages from the coordinator, it could receive the message in an unsuitable state and wrongly execute the compensatory action. It means that the participant receives the message without executing its subtransaction (10), the participant receives a compensate message when it has finished its participation in the transaction (11), it receives the compensate message when it has already executed its compensatory action (12) or it receives the compensate message when it does not need to execute any compensatory action (13). One of the possible situations identified in (10) is that a participant was in a failing state (14). One possible situation based in (11) is that a compensate message is received when the participant has already executed its subtransaction and it was not necessary to execute the compensatory action (15). Based on the situations identified in the leaf nodes, we define a test scenario for each one specifying the transactions notifications to reach this situation.

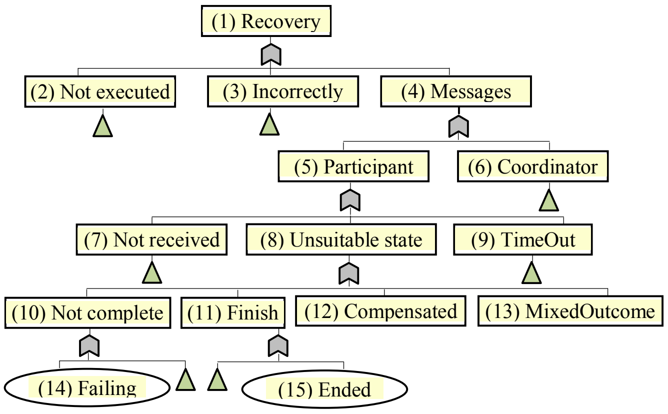

Figure 2. Recovery Fault Tree

IV. CURRENT AND FUTURE WORK

A short term work is to study which are the most suitable testing techniques to check the rest of system properties. As we have commented, FTA technique was successfully applied for the *recovery* property.

Currently we have developed an abstract model to pattern the participants in a web service transaction. It describes semantically the behaviour of the participants independently of the transaction protocol used. This abstract model takes in account the possible faults that may occur during the process. Now we are working to apply model based testing techniques over that model, thus the *composition* and *controllability* properties will be covered.

Godart et al [20] have done a interesting work about transactional patterns in web service compositions. We have identified it as a key issue in the *order* property too. So we are starting collaboration in order to study how to apply flow based testing over composite service transactional behaviour. For the rest of properties more research is needed to select the suitable testing techniques.

In order to validate our approach, firstly we will use the test cases achieved using the framework over a transaction simulation environment. We will apply fault injection techniques to measure the quality of the generated test cases. Using that information we will be able to feedback the framework and therefore, improve the test cases. The next step in the validation process will be to execute the defined test cases in a real implementation by developing the some simple examples (i.e. using Jboss Transactions [21] API). A new useful feedback will be used again to refine the framework. Finally, we will execute the test cases in a real application.

CONCLUSIONS

Although transactions are a key issue in web service compositions, there are no practical approach to test them. The main contribution of our research is to create a specific conceptual framework to testing web service transactions. The framework is an adaptation of risk based methodologies.

The work done to date seems to show the viability of our approach. Also a clear line of research is specified to follow in the current and future work.

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), grant BES-2008-004355.

REFERENCES

[1] M. Younas, K. Chao, C. Lo, and Y. Li, "An Efficient Transaction Commit Protocol for Composite Web Services," International Conference on Advanced Information Networking and Applications (2006).

[2] G. Canfora, and M. Penta, "Service-Oriented Architectures Testing: A Survey," Software Engineering: International Summer Schools, ISSSE 2006-2008, Salerno, Italy, Revised Tutorial Lectures, Springer-Verlag, 2009, pp. 78-105.

[3] R. Lanotte, A. Maggiolo-Schettini, P. Milazzo, and A. Troina, "Design and verification of long-running transactions in a timed framework," Science of Computer Programming (2008) 76-94.

[4] M. Emmi, and R. Majumdar, "Verifying Compensating Transactions," International Conference Verification, Model Checking, and Abstract Interpretation (2007).

[5] J. Li, H. Zhu, and J. He, "Specifying and Verifying Web Transactions," International conference on Formal Techniques for Networked and Distributed Systems (2008).

[6] W. Gaaloul, M. Rouached, C. Godart, and M. Hauswirth, "Verifying composite service transactional behavior using event calculus," Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I, Springer-Verlag, Vilamoura, Portugal, 2007, pp. 353-370.

[7] J. Gray, "Notes on Data Base Operating Systems, Operating Systems, An Advanced Course," Springer-Verlag, 1978, pp. 393-481.

[8] M. Kaufmann, "Database Transaction Models for Advanced Applications".

[9] E.B. Moss, "Nested Transactions: An Approach to Reliable Distributed Computing," Massachusetts Institute of Technology (1981).

[10] H. Garcia-Molina, and K. Salem, "Sagas," Proc. SIGMOD 87 (1987) 249-259.

[11] R. Casado, and J. Tuya, "Testing Transactions in Service Oriented Architectures," International Conference on Web Engineering DC, San Sebastian, Spain, 2009.

[12] OASIS, "Business Transaction Protocol," http://www.oasis-open.org/committees/tchome.php?wg-abbrev=business-transaction.

[13] OASIS, "Web Services Composite Application Framework," http://www.oasis-open.org/committees/tc-home.php?wg-abbrev=ws-caf.

[14] OASIS, "Web Services Atomic Transaction," http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec/wstx-wsat-1.1-spec.html.

[15] OASIS, "Web Services Business Activity," http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-errata-os.pdf.

[16] OASIS, "Web Services Coordination," http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-errata-os/wstx-wscoor-1.1-spec-errata-os.html.

[17] R. Casado, J. Tuya, and M. Younas, "Specifying and testing recoverability requirements in WS-BusinessActivity transactions," Jornadas Cientifico-Tecnicas en Servicios Web y SOA, Valencia, Spain, 2010.

[18] R. Casado, J. Tuya, and M. Younas, "Testing Long-Lived Web Services Transactions Using a Risk-Based Approach," Proceedings of the 2010 10th International Conference on Quality Software, IEEE Computer Society, 2010, pp. 337-340.

[19] W. Vesley, F. Goldberg, N. Roberts, and D. Haasl, "Fault Tree handbook," U.S. Nuclear Regulatory Commission, 1981.

[20] S. Bhiri, C. Godart, and O. Perrin, "Transactional patterns for reliable web services compositions," Proceedings of the 6th international conference on Web engineering, ACM, Palo Alto, California, USA, 2006, pp. 137-144.

[21] Jboss, "Jbosss Transaction," http://jboss.org/jbosstm, 2010