MANAGING TRANSACTIONAL COMPOSITIONS OF WEB SERVICE APPLICATIONS

Juha Puustjärvi
Helsinki University of Technology, Innopoli 2, Tekniikantie 14, Espoo, Finland

Keywords: Web service composition, Transaction coordination protocols, Fault tolerance, Ontologies.

Abstract: The ACID transaction model has evolved over time to incorporate more complex transaction structures and to selectively relax the atomicity and isolation properties. Such advanced transaction models are more appropriate for SOA, which is geared toward open environments consisting of autonomous and heterogeneous systems. However, due to the autonomy and heterogeneity of local systems supporting transactional compositions of Web service applications is problematic in SOA. In addition, the interfaces of Web services are not usually designed for transactional compositions. Neither there are mechanisms for registering Web services' abilities to participate transactional compositions nor mechanisms for registering Web services' coordinators. How these problems can be avoided by introducing a Composition server is the topic of this paper.

1 INTRODUCTION

The goal of Web services is to achieve universal interoperability between applications by using web standards. The full potential of Web services will be achieved only when applications and business process interactions are coordinated in a transactional way.

The Web Services Transactions specifications (IBM, 2008; Singh and Huhns, 2005) define mechanisms for transactional interoperability between Web services domains. Particularly, they describe an extensible coordination framework (WS-Coordination) and specific coordination types for short duration ACID transactions (WS-AtomicTransaction) and for Longer running business transactions (WS-BusinessActivity).

WS-Coordination (Papazoglou and Heuvel, 2007) is a general and extensible framework in the sense that its use is not restricted to WS-AtomicTransaction and WS-BusinessActivity but it can be exploited in developing coordinators for any transaction model suitable for composing Web service applications.

During the past few years many transaction models suitable for composing Web services are proposed. For example, the isolation requirements have been relaxed in the models presented in (Alrifai et al., 2006; Puustjärvi, 2004; Choi et al., 2005; Guabtni et al, 2006). The use of semantic information in weakening the atomicity criterion is studied in (Ding et al., 2006; Zhao et al., 2005; Puustjärvi, 2006), and the use of semantics in compensating the failed actions are studied e.g., in (Fauvet et al., 2005; Schmit et al., 2005), and the atomicity protocols are studied in (Xu et al., 2006; Younas and Chao, 2006).

Many of these transaction models and coordination protocols would be usable and appropriate in Service Oriented Architecture (SOA) (Singh and Huhns, 2005). However, their deployment suffers from the absence of appropriate Web services' interfaces as each atomicity protocol and each concurrency control protocol set their own requirements on the functionality and interfaces of the participating Web services.

Further, in many cases, such requirements contradict with the autonomy of the sites providing Web services. The reason for this is that autonomy requires that the components in an environment function solely under their own control but achieving global consensus is not always possible under such settings.

This situation is analogous with distributed database systems, where the atomicity of the distributed transaction is carried by the 2-phase commit protocol (2PC) (Gray, 1993) and concurrency control is carried out by 2-phase

PuustjÃď'rvi J.
MANAGING TRANSACTIONAL COMPOSITIONS OF WEB SERVICE APPLICATIONS.
DOI: 10.5220/0001836403110316
In *Proceedings of the Fifth International Conference on Web Information Systems and Technologies (WEBIST 2009)*, page
ISBN: 978-989-8111-81-4

locking protocol (2PL) (Gray and Reuter, 1993). Here the problem is that each participant must keep log record required by the termination protocol of the 2PC protocol, and locks on data items required by the 2PL protocol until the coordinator of the protocol requests to release the locks. The actual problem here is that local applications are not allowed to access the locked data items, i.e., local systems have lost their autonomy. This is a reason why strict transactional properties are not usually provided in SOA.

Due to the autonomy of local systems supporting transactions is more complex in SOA. Some Web services may not have transactional functionalities (e.g., keeping log records) at all, some may have the ability to perform compensating actions, and some may have the ability to set locks on data items.

Another problem is that the descriptions of Web services, i.e., their WSDL-descriptions (WSDL, 2001), are based on assumption that a single requester uses the service, but in transactional composition requires different dialog where the coordinator has to communicate with the service, and hence each Web service should have a specific interface description (WSDL description) for each coordination protocol.

How to cope with these problems is the topic of this paper. The cornerstone of our solution is a Composition server, which specifies how and which Web services can be composed in a transactional way. Technically the Composition server is a Web service supporting an update and querying interfaces.

The rest of the paper is organized as follows. In Section 2, we first give a short overview of the transaction models that are proposed for composing Web services in a transactional way. Then, we present the structure of our used Composition ontology, and the functionality of the Composition server. In Section 3, we first describe how WS-Coordination specifications can be used in developing coordinators for various coordination types (transaction models). Then, we give an example how UDDI registry and Composition server are exploited in Web services composition. In particular, we describe the message exchange between coordinators and Web service applications in a fault tolerant atomicity communication protocol. Finally Section 4 concludes the paper by discussing the advantages and disadvantages of our developed solutions.

2 MANAGING WEB SERVICES' TRANSACTIONAL PROPERTIES

2.1 Transaction Models

The traditional notion of a database transaction is so called ACID-transaction (Gray and Reuter, 1993), which has the following properties: *Atomicity* means that either all of a transaction is executed or none of it is. *Consistency* means that the state of a database satisfies the consistency constraints of the database. *Isolation* means that concurrently executed transactions do not interfere with each others. *Durability* means that if a transaction has completed its work, then its effect should not get lost even if the system fails.

Supporting ACID-transactions in a distributed environment requires specific concurrency control mechanism and an atomic commitment protocol, e.g., 2PC- protocol.

An alternative optimistic model has been proposed in database research (originally called Sagas), where actions have explicit compensatory actions which negate the effect of the action. In the real world of actions, the existence of compensatory actions is quite common for some actions. For example, for a debit a credit card 100, the compensatory action is to credit the credit card $100. On the other hand, some actions may be difficult to compensate. For example, rolling back a business transaction (e.g., reservations on a flight) is not always free of charge.

XLang (Xlang, 2002) is actually a workflow model but it is based on Sagas and therefore we can also regard it as transaction model, which do not support any of the ACID- properties but rather semantic atomicity. Thus XLang is actually a notation for expressing the compensatory actions for any atomic transaction that needs to be undone. Hence XLang is appropriate transaction model for the actions having common compensatory actions.

BTP Atomic Transactions (BTP, 2002), are similar to transactions in tightly coupled systems (i.e., to ACID-transactions), but the isolation property is relaxed. Thus, BTP has improved the notion of traditional distributed ACID transaction to loosely coupled environments with the required weakening of the I-property (Isolation-property). However, providing BTP-transaction models requires the implementation of the 2PC-protocol.

In order to avoid the problems related to compensatory actions, a transaction model, called

Composed Web Service Transaction model, or CWS-transaction model (Puustjärvi, 2006) for short is also proposed. It deviates from other advanced transaction models in that it is not based on compensating transactions, but rather it divides the traditional business transaction into two successive transactions, called request transaction and decision transaction. The commitment of the request transaction ensures that the decision transaction will not fail, and so the atomicity of the CWS-transaction can be ensured. For example, with hotel reservation case, the successfully executed request transaction makes only a preliminary reservation (not an actual reservation), which is then confirmed to real reservation or rejected by the decision transaction.

2.2 Composition Ontology

The term ontology originates from philosophy where it is used as the name of the study of the nature of existence (Gruber, 1993). In the context of computer science, the commonly used definition is "An ontology is an explicit and formal specification of a conceptualization" (Davies et al., 2002). It consists of a finite set of concepts and the relationship between the concepts. Essentially the used ontology must be shared and consensual terminology as it is used for information sharing and exchange.

The purpose of our used Composition ontology is to specify transaction models, Web services, transaction coordinators, and their relationships. This ontology is described by OWL (Web Ontology Language) (OWL, 2005), and for illustrative purposes it is graphically presented in Figure 1, where ellipses represent classes and boxes represent properties.

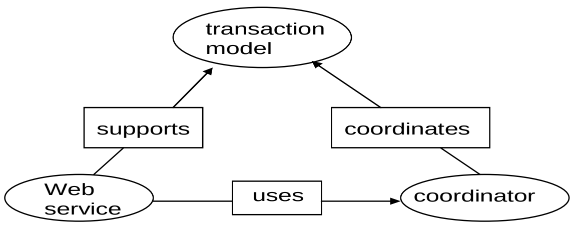

Figure 1: Composition ontology.

The instances of the ontology are defined by RDF-statements. The RDF (Resource Description Framework) model (RDFS, 2005) is called a triple because it has three parts: subject, predicate and object. Each triple is an RDF-statement. For example the statement "Hotel Lord's Web service supports BTP transaction model" is an RDF statement, where "Hotel Lord's Web Service" is the subject, "supports" is the predicate, and "BTP-transaction model" is the object.

In order that RDF-statements can be represented and transmitted it needs syntax. The syntax has been given in XML. So an RDF-statement can be represented as an XML-element. Further, an RDF-document is comprised of one ore more RDF-descriptions, and each RDF-description is comprised of one or more RDF-statement.

In Figure 2, an RDF description, which is comprised of the three RDF-statements, is presented.

```
<rdf:RDF
      xmlns : rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns : xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns : co="http://www.lut.fi/ontologies/composition-ontology#"
      <rdf:Description rdf:about="Lord's_hotel_web_service">
         <rdf:type rdf:resource="&co;web-service"/>
         <co : supprts>BTP-transaction</co : supports>
         <co : coordinates>Coordinator-123Y</co : coordinates>
      </rdf : Description>
</rdf:RDF
```

Figure 2: An RDF – descriptu in RDF/ XML serialization format.

Note that, in the above RDF-description XML namespace "xmlns:co" refers to the used composition–ontology. The first RDF-statement in the description states that "Lords-hotel-web-service" is an instance of the class web-service, which is a class in the Composition ontology.

2.3 Composition Server

Composition Server is a Web service, which function is to provide querying and update interface for the Composition ontology. It is used in constructing contexts for Web services' compositions. Each context includes context identifier, the services to be coordinated, the coordinators as well as the coordination type (transaction model).

Composition server has a publishing interface and an inquiry interface. Using the publishing interface new instances to the ontology can be inserted. Using the inquiry interface an application (or a human) can make for example the following queries:

- What are the transaction models that Hotel Lord's Web service supports?
- Give the information of the coordinator that is used by the Hotel Lord's Web service.
- What are the transaction models that are coordinated by the Coordinator C1.

3 COORDINATING WEB SERVICES

3.1 WS-Coordination

A way to coordinate the activities of Web services is to provide a web service which function is to do the coordination. In order to alleviate the development of such coordinators WS-Coordination provides a specification that can be utilized in developing the coordinator. As we use this specification in developing the coordinator, we first consider this specification and then illustrate the functionality of the fault tolerant atomicity coordinator by an example.

According to the WS-Coordination a coordinator is an aggregation of the following services (Singh and Huhns, 2005). As illustrated in Figure 3, the Activation service defines the operation that allows the required context to be created. In particular, a context identifier is created and passed to the Web services that participate to the same coordination. The Registration service defines the operation that allows a Web service to register its participation in a coordination protocol. A coordination protocol service coordinates a supported coordination type (transaction model).

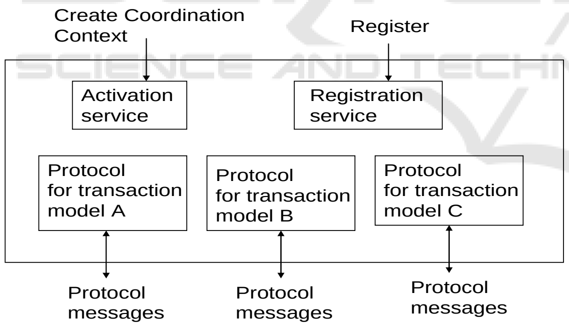
Create Coordination Context
Register
Activation service
Registration service
Protocol for transaction model A
Protocol for transaction model B
Protocol for transaction model C
Protocol messages
Protocol messages
Protocol messages

Figure 3: Coordinator for transaction models A, B and C.

The architecture (following the specification of WS-Coordination) of the coordinator that supports transaction models A, B and C is presented in Figure 3.

After an application has created a coordination context, it can send it to another application. The context contains the information required for the receiving application to register into the coordination. In principle, an application can choose either the registration service of the original application or use some other (own) coordinator. In the latter case the application forwards the context to the chosen coordinator.

3.2 Message Exchange in Fault Tolerant Coordination

In our used architecture we assume that each participating application use its own coordinator which have the role of participant in the coordination. The message exchange between the coordinator, participant and the applications (in the case of no failures) is illustrated in Figure 4. The figure illustrates the case where a travel agent tries to make reservation on a flight and a room reservation on a hotel. The used coordination type follows the 2PC-protocol. The protocol is fault tolerant in the sense that a termination protocol is invoked when a participant in the coordination has been waiting a predetermined time for a message. The functionality of the termination protocol is shortly presented in Section 3.3.

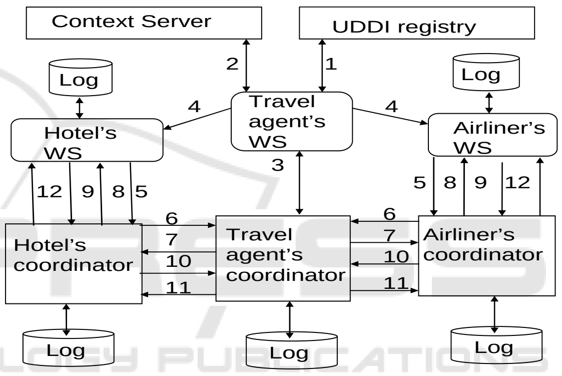

Figure 4: Communication structure.

The communication in the figure proceeds as follows:

1. Travel agent's Web service queries from the UDDI registry which airlines have flights, say from Amsterdam to London, and which Hotels are in centre of London. From the returned choices the travel agent's Web service decides to choose, say Lufthansa and Hotel Lord.

2. Travel agents Web service queries from the Context server which transaction models Lufthansa's and Hotel Lord's Web services support. Both Web services support fault tolerant atomic commitment protocol (FTACP), and so travel agent's Web service chooses the protocol.

3. Travel agent's Web service asks its coordinator to create a coordination context for FTACP-type coordination, and then the coordinator returns the context, which includes information where its registration service can be found.

4. Travel agent's Web service sends reservation messages to hotel's and airliner's Web services. Both messages include the context information.

5. Hotel's Web service and Airliner's Web service send the context information to their own coordinators.

6. Hotel's coordinator and airliner's coordinator register to travel agent's coordinator.

7. Travel agent's coordinator sends the request message to hotel's and airliner's coordinator and writes start-record in its log.

8. Hotel's coordinator and airliner's coordinator request whether their Web services are able to execute the reservation.

9. Hotel's Web service and airliner's Web service write their votes (Prepared or Aborted) into their logs and inform their coordinators about their votes.

10. Hotel's coordinator and airliner's coordinator write the vote in their logs, and then inform travel agent's coordinator about their vote.

11. If there were no Abort-message (i.e., airliner's and hotels Web services' votes were Prepared), then travel agent's coordinator sends the Commit-message to hotel's and airliner's coordinator; and otherwise it sends the Abort-message.

12. Hotel's coordinator writes the decision (commit-record or abort-record) in its log and informs hotels' Web service whether the transaction is committed or aborted, and respectively airliner's coordinator writes commit-record in its log and informs airliner's Web service.

A salient feature of the above protocol is that each Web service can unilatery decide Abort at any time, if it has not voted Prepared: After voting Prepared a Web service cannot take unilateral action. The period between the moment a Web service votes *Prepared* and the moment it has received sufficient information to know what the decision will be is called the *uncertainty period* for that Web service. A Web service is called *uncertain* while it is in its uncertainly period.

During this period (during the steps 8-12 in the previous example) the Web service does not know whether it will eventually decide Commit or Abort, nor can it unilaterally decide Abort. So, for example, during the uncertain period the Hotel is not allowed to reserve the room for any other customer. How these kinds of constraints are enforced is application or system dependent, e.g., by setting a write-lock on reservation data, or by using semantic information in the reservation application.

3.3 Managing System and Communication Failures

In the case of system or communication failure a service must await the repair of failures before proceeding, and so the service is blocked. Blocking is undesirable, since it can cause services to wait for an arbitrarily long period of time.

In order that a blocked service can proceed it must communicate with the coordinator (travel agent's coordinator in the case of previous example). This kind of communication is carried out in a termination protocol, which is activated by the participant (hotels or airliner's coordinator in the case of previous example) when it has been waiting a predetermined time for a message.

Our used termination protocol of the atomic commitment protocol goes as follows:

1. Hotel's coordinator (or airliner's coordinator) sends decision-request-message to travel agents coordinator.

2. Travel agent's coordinator sends the response-message to hotel's Coordinator (travel agent's coordinator).

Hotels coordinator (travel agent's coordinator) repeats the request,, if it has not receive the response in a predetermined time period. Note that travel agent's coordinator is always able to response to the request as it has no uncertainty period.

4 CONCLUSIONS

The full potential of Web services will be achieved only when applications and business process interactions are coordinated in a transactional way. The issue of Web services coordination is widely studied, and many transaction models suitable for composing Web services are proposed. However, the deployment of the transaction models is not straightforward as Web services' interfaces are not usually designed for compositions. This is regrettable, since it is obvious that by supporting one or more coordination types, a Web service can increase its usability, and more advanced composed Web services can be designed.

In order to simplify the composition of Web services we have introduced the Composition server, which can be used in publishing and querying Web services' abilities to participate on various coordination types.

In the future work we will also analyze the replacement of the Composition server by extending the UDDI registry by the information included in the Composition server. However, it is obvious that a drawback in this approach will be that the querying features of UDDI registry are restricted to keywords, and so we would loose the expression power that can be achieved by the ontology based Composition server.

REFERENCES

Alrifai, M., Dolog, P., Nejdl, W., 2006. Transactions concurrency control in web service environment. *Proceedings of the 4th European Conference on Web Services,* pages. 109-118.

BTP, 2002. BTP- Business Transaction Protocol, http://www.oasis-open.org/committees/business-transactions/documents/primer/

Choi, S., Jang, H., Kim, H., Kim, J., Kim, S., Song, J., Lee, Y., 2005. Maintaining consistency under isolation relaxation of web services transactions. *Proceedings of 6th International Conference on Web Information Systems Engineering,* In Lecture Notes in Computer Science, 245-257.

Davies, J., Fensel, D., Harmelen, F., 2002. *Towards the semantic web: ontology driven knowledge management.* West Sussex: John Wiley & Sons.

Ding, X.,Wei, J., Huang, T., 2006. User-defined atomicity constraint: A more flexible transaction model for reliable service composition. *Proceedings of 8th International Conference on Formal Engineering Methods,* In Lecture Notes in Computer Sciences, pages 168-184.

Fauvet, M., Duarte, H., Dumas, M., Benatallah, B., 2005. Handling transactional properties in web service composition. *Proceedings of 6th International Conference on Web Information Systems Engineering,* In Lecture Notes in Computer Sciences, pages 273-289.

Gray, J., Reuter A., 1993. *Transaction Processing: Concepts and Techniques*. Morgan Kaufman.

Gruber, T., 1993.Toward principles for the design of ontologies used for knowledge sharing. *Padua workshop on Formal Ontology.*

Guabtni, A., Charoy, F., Godart, C., 2006. Concurrency management in transactional web services coordination. *17th International Conference, DEXA.* In Lecture Notes in Computer Science, pages. 592-601.

IBM, Web Services Transactions specifications, 2008. http://www.ibm.com/developerworks/library/specification/ws-tx/

OWL–WEB Ontology Language, 2005. http://www.w3.org/TR/owl-ref/

Papazoglou, M., van den Heuvel, W., 2007. Service oriented architectures: approaches, technologies and research issues.*The International Journal on Very Large Data Bases*, 16, 3.

Puustjärvi, J., 2004. Concurrency control of Internet-based workflows. *In Proc of the Sixth International Conference on Information Integration and Web-based Applications & Services (iiWAS2004),* pages 647-654.

Puustjärvi, J., 2006. CWS-transactions: An approach for composing web services. *Second International Conference on Web Information Systems and Technologies (WEBIST),* pages, 69-74.

RDFS. 2005. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

Schmit, B. ja Dustdar, S., 2005. Towards transactional web services. Proceedings of the 7th IEEE *International Conference on E-Commerce Technology Workshops (CECW'05),* pages 12-20.

Singh, M., Huhns, M., 2005. *Service Oriented Computing: Semantics, Processes, Agents*. John Wiley &Sons, Ltd.

WSDL, 2001. WSDL- Web Services Description Language.http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

XLang–Web Services for Business Process Design. 2002.http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

Xu, W., Cheng, W., Liu, W., 2006. A transaction-aware coordination protocol for web services composition. *Proceedings of 7th International Conference on Web Information Systems Engineering,* In Lecture notes in Computer Science. Springer, pages126-131.

Younas, M., Chao, K., 2006. A tentative commit protocol for composite web services*. Journal of Computer and System Sciences.*

Zhao, W., Moser, L., Melliar-Smith, P., 2005. A reservation-based coordination protocol for web services". *Proceedings of the IEEE International Conference on Web Services (ICWS'05),* pages, 49-56.