Web Services Transactions (WS-Tx)

Rami Järvinen

Helsinki 17.11.2009

Seminar paper

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Rami Järvinen

Työn nimi — Arbetets titel — Title

Web Services Transactions (WS-Tx)

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminar paper	17.11.2009	

Tiivistelmä — Referat — Abstract

The traditional ACID-properties of transactions are too strong withing the web service environment, as web service transactions can be part of long running business processes. Strict ACID-transactions require the data to be locked during a transaction in order to assure atomicity and isolation. To provide a remedy for this problem, we need protocols that relax the traditional ACID-properties by focusing to data-centric consistency, process consistency and to enabling long-running transactions by using standardized XML messages that are coordinated in order to execute transactions in an organized way.

The Web Services Transactions (WS-Tx) specifications describe a one approach for transactional interoperability between web services. The aim of these specifications is to provide reliable and consistent execution of business transactions using heterogeneous interconnected web services.

WS-Tx specifications describe an extensible coordination framework (WS-Coordination) that is focused on outcome determination and processing. Also specific coordination types for short duration ACID like transactions (WS-AtomicTransaction) and long running loosely coupled business transactions (WS-BusinessActivity) are defined.

This seminar paper will first give out reasons why we need this kind of specifications: the need for more relaxed forms of transactions in web services environment. After that an high level introduction to WS-C, WS-AT and WS-BA is presented. Finally we will point out a few problems with regarding to protocols in WS-Tx specifications.

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Transactions have their roots in the business-domain. Exchanges of different kinds of items are represented by business transactions. Computer systems working with automated transaction processes access shared database data. In order to access the data in a way that the data stays consistent, transactions must fulfil the ACID properties [FS02].

- Atomicity - no operations within a transaction will happen if one operation fails
- Consistency - valid state transitions
- Isolation - the effects of the operations are not shared outside the transaction before successful completion
- Durability - successfull data changes survive future failures

Should the concept of atomic transaction not exist, the application would have to determine the exact point of failure where exception occured before getting system back to a consistent state. ACID properties do greatly simplify the exception recovery process. Transaction shields the application from system-generated exceptions like network errors.

Two phase commit protocol (2PC) is often used to ensure atomicity across different participants of a transaction. Further, the transaction is usually managed by multiple resource managers. Co-operation between those resource managers is enabled by the 2PC protocol.

In the first phase of a 2PC protocol, every participant in the transaction save the state changes for possible roll-backs or commits in the future, but doesn't commit the transaction at this point. After the first phase is complete, all participants are prepared for the second phase. The transaction is actually committed during the second phase and the original data state is updated with the stored changes. Both phases are executed by the coordinator of the transaction. The time between the two phases should be quite short, no more than a few milliseconds to prevent unnecessary locking of resources. As ACID properties require, the transaction is either commited at all participants or at none.

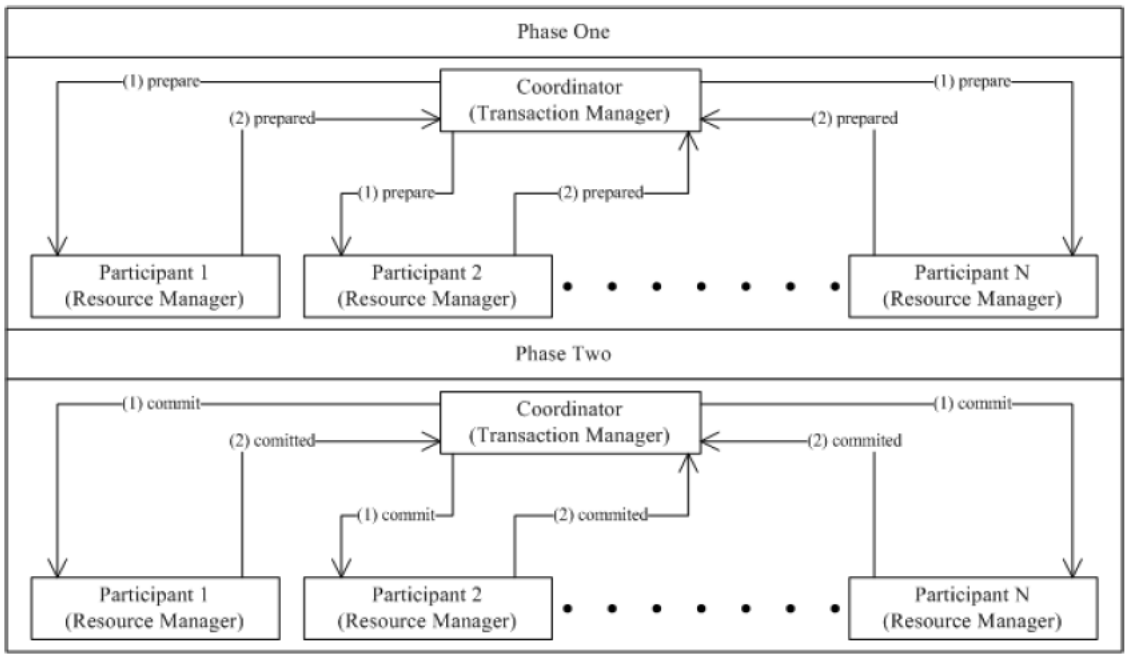

Figure 1: 2PC

The ACID properties and 2PC protocol hold for distributed transactions spanning over multiple computer systems. To execute distributed transactions over boundaries of multiple organisations, platform-dependent workflow systems can be used to connect activities taking place at the different systems. In these cases, a tight coupling between the participating organisations is required. Also, this method can be used only withing long-term relationships between organizations.

Todays organisations, coping with a fast changing environment, require a loose coupling of systems to engage in short-term collaborations with multiple partners. Service oriented computing (SOC) paradigm can be used for connecting heterogeneous systems on an Ad Hoc basis. In SOC, organisation can expose resources grouped as a process to the network. Process in this context means a set of actions that an organization is prepared to execute. This exposed web service defines a set of messages in standardized XML and couples actions to the reception of messages, together with the return of a reply message. When different web services are coupled on an ad hoc basis, business processes spanning over multiple organisations can be executed in order to achieve a business task in a more dynamic fashion than it would happen in traditional workflow systems.

While in traditional workflow systems the coordination activities are centrally organized and opening up back-end systems is needed, web services can hide back-end implementations and the business functions can be kept in control of organizations by themselves. Also, web services are based on the Web Service stack, which consist of platform independent standards like WSDL and SOAP. The downside here is that the connecting multiple web services together increase the complexity of applications and some kind of coordination between these services is needed.

Spanning business process activities across different web services requires correlation and coordination mechanisms that not only keep track of the ongoing process and transactions involved in that process, but that also do not infringe the autonomy of the participating organizations.

The traditional ACID-properties of transactions are too strong withing the web service environment, as web service transactions can be part of long running business processes. Strict ACID-transactions require the data to be locked during a transaction in order to assure atomicity and isolation. In a long running transaction it's highly impractical to lock data as it potentially blocks resources for long periods of time and makes them not available for others (e.g., locking a a book-sales table blocks other potential customers resulting in lower turn-over). Furthermore, a long-running transaction requires participating organizations to negotiate commitments to the transaction.

To provide a remedy for abovementioned problems, we need protocols that relax the traditional ACID-properties by focusing to data-centric consistency, process consistency and to enabling long-running transactions by using standardized XML messages that are coordinated in order to execute transactions in an organized way.

By definition, any Web Services transaction model should support the following functionality

- Relaxation of ACID properties in a structured, well-defined manner; strict ACID properties, especially atomicity are not appropriate for all applications. Most of the long-duration activities do not need all-or-nothing paradigm. Also the isolation must be relaxed, so that the results of tasks are exposed before whole activity has terminated.

- Flexible outcomes for consensus groups. They might be formed as open-flat, where the participants in a transaction are exposed to the business logic allowing it to define the relationships of the individual units of work to the

transaction task. Also open-nested formation would be possible, where there are tasks within an activity forming a parent-child relationship of consensus groups.

- Consensus groups participation must be flexible; a task may leave an activity prior to outcome processing if it decides it does not affect that processing. Then it will not expose results or cause side-effects happening in other tasks.
- Activities and tasks should be defined as individual scopes (consensus groups), with clear relationships between them so that the service can explicitly define responsibilities. Scopes allow the work performed by services or a long-running activity to be clearly demarcated by the application or the service. In addition, termination of scopes resides in the domain of the application and it driven from top-down.

2 The Web Services Transactions specifications

The Web Services Transactions specifications include WS-Coordination (WS-C) which is an extensible coordinator that can be used to coordinate virtually any kind of activity, e.g. atomic transactions, long-running transactions and workflow processes. As being an abstract part of a coordinator framework it is not able to coordinate any activity by itself. To do so, the coordinator has to be extended. Protocols WS-AtomicTransaction and WS-BusinessActivity are extensions to the WS-C protocol.

The Web Service Transaction was originally a specification of two transaction protocols - atomic transaction and business activities. The specification was split in two separate specifications containing updated versions of the original WS-Tx protocols. Web Service Atomic Transaction (WS-AT) specification is a protocol for short-lived activities and Web Service Business Activity (WS-BA) specification introduces a protocol for handling long-lived activities. Typically a long running business activity contains atomic transactions, so these transaction protocols can be combined.

2.1 WS-Coordination

The Web Services Transactions protocol defines a separate protocol focussing solely on outcome determination and processing: Web Services Coordination. The fun-

damental idea underpinning WS-Coordination is that there is a generic need for a coordination infrastructure in a Web services environment. The WS-Coordination specification defines a framework that allows specific coordination protocols (e.g. for security or transaction handling) to be plugged-in to coordinate work among clients, services, and participants.

Coordination is a requirement in a variety of different aspects of distributed applications, such as workflow, security, atomic transactions, caching and replication, security, auctioning, and business-to-business activities. For example, coordination of multiple Web services in choreography may be required to ensure the correct result of a series of operations comprising a single business transaction.

Coordination is the act of one agent (the coordinator) disseminating information to a number of participants to guarantee that all participants obtain a specific message. A coordinator can also be a participant (concept of interposition), creating a tree of sub-coordinators or peer-coordinators that cooperate to further propagate the result.

The coordination framework consist of three elements representing the basic responsibilities of all different kinds of coordination protocols [CCF+05a].

- Activation service - responsible for the creation of a new activity coordinator for a particular application instance.
- Registry service - responsible for the enrolment of registration of new web services and the coordination protocol selection
- Coordination service - responsible for ensuring that the registered web services are driven through completion by using the selected protocol

The coordination service does the actual controlling of the activity completion processing for Web Services that have been registered. Controlling is done using the selected coordination protocol (e.g. WS-AT or WS-BA). The particular coordination protocol then defines the operations needed for completion processing. The coordinator usually provides interfaces for commits and rollbacks and participant provides an interface for agreements like prepare, commit and rollback.

WS-Coordination does not define the entity that drives the coordination protocol from start to completion. This entity is usually a client application. Protocols based on the coordination framework (i.e. actual implementations) define the interface for the client.

Context information is required to flow between the co-ordinated participants. In WS-C, the context contains *correlation identifier* which is globally unique. This identifier tells which tasks belong to same activity. Context contains the location or endpoint address of the coordinator to enable new participants to join in. Activation service returns a CoordinationContext for each newly created activity.

Context information flows implicitly (transparently to the application) within normal messages sent to the participants. Usually context is exchanged by using the header of SOAP messages. This information is specific to the type of coordination being performed, for example to identify the coordinator(s), the other participants in an activity, recovery information in the event of a failure, etc.

2.2 WS-AtomicTransaction

Despite transactions usually being long-lasting in Web services, there's a need for having support for short-duration interactions. With the AtomicTransaction (WS-AT) protocol, consensus groups can be established and strict atomicity can be forced among their participants. Traditional transaction systems used in legacy systems can be adopted to the web service environment by wrapping them with Atomic Transaction. WS-AT is specifically useful in intra-domain environments where operations in number of internal applications need to be consolidated.

WS-AtomicTransaction specifies following protocols for atomic transactions [CCF+05b].

- Completion - Commitment processing initiator protocol
- Two-Phase Commit (2PC) - a Coordination protocol defining how multiple participant reach agreement on the outcome of an atomic transaction
  - Volatile 2PC - used by participants managing volatile resources
  - Durable 2PC - used by participants managing durable resources

Application uses The Completion protocol to tell the coordinator to either try to commit or abort a transaction. Status, either a `Commit` or `Rollback`, will be returned to the application once internal processing in the application has completed. After that, coordinator continues with the prepare phase for the Volatile 2PC protocol. All participants registered for Volatile 2PC are expected to respond to `Prepare` message sent by the coordinator by sending `Prepared` or `Aborted` back to the coordinator.

Volatile recipient might not receive a notification of the transaction's outcome, since by registering for using volatile protocol means that the participant isn't interested in the final state.

After the Prepare phase for Volatile 2PC participants is succesfully completed, the coordinator starts with the Prepare phase for Durable 2PC. Any participants registered for Durable 2PC must respond `Prepared` or `ReadOnly` before coordinator issues a `Commit` notification for registered participant. By sending `ReadOnly`, the participant states that its voting to commit the transaction but won't participate in phase 2. It's also possible for the participant to ask for a `Replay`. Then the coordinator may send the last approriate protocol notification again, because it will be assumed that the participant has suffered a recoverable failure. (Figure 2)

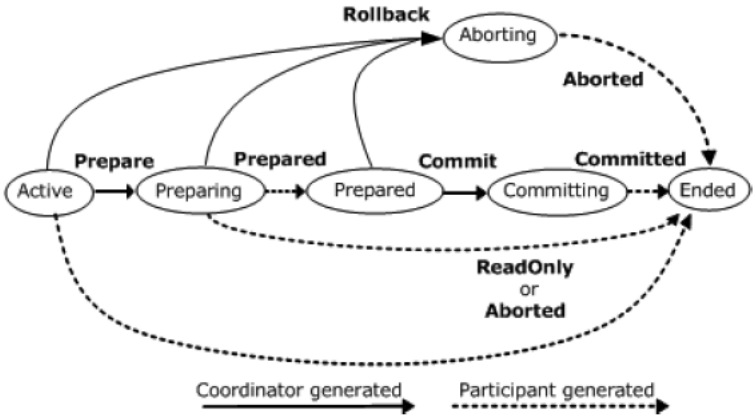

Figure 2: 2PC state diagram

All participants registered to use AtomicTransaction must implement 2PC protocol and thus be supporting following requirements:

- Isolated updates until commit
- Single all-or-nothing decision for all participants
- Entire transaction can be aborted by any participant

2.3 WS-BusinessActivity

The WS-BusinessActivity (WS-BA) specification provides protocols with relaxed transaction properties and is designed for long-duration interactions. These protocols enable existing business processes and workflow systems to interoperate. An

application coordinating the business activity (a long running transaction), won't care about system generated exceptions, but instead focuses on the handling of application generated exceptions. Application generated exceptions are here called business exceptions. To handle business exceptions, we need a business logic to drive the overall business transaction. BusinessActivity definitions are of help here by providing mechanisms to reach overall agreement. Atomic transaction is used for preserving the autonomy of organisations which are participating in the business activity.

The full ACID semantics are not maintained by BusinessActivity, but it the data consistency isn't still affected. Consistency is maintained through compensation. Services under control of BusinessActivity provide compensating actions for any error situations that may occur in their systems. A business activity is usually sliced into business tasks or units of work using a collection of Web Services. These slices are called scopes. Scopes can be nested to tree like branches that form parent and child relationships. Interesting thing here is that such compensations will typically use forward error recovery instead of backward one. A business activity is usually sliced into business tasks or units of work using a collection of Web Services. These slices are called scopes. Scopes can be nested to tree like branches that form parent and child relationships. Compensating actions are registered with the parent scope in order to undo completed child tasks. Exception handlers in the parent scopes deal with exceptions by doing forward recovery so the overall business activity can continue despite exceptions happen. Scope nesting provides *fault-isolation*. Is child scope fails, it does not require the enclosing scope to fail.

Results of completed transactions can be seen prior to the completion of the business activity. Tasks are therefore tentative and in the need of compensation, business logic is required to be implemented in a way that it will be possible. This relaxes the isolation property of ACID.

Characteristics of BusinessActivity also include following properties. It is ok for participants to delegate processing to the other scopes or exit without being interested of the outcome. A participant may also tell its own task outcome to coordinator without being asked for it. This notification should be used by the exception handler to re-evaluate the situation and continue processing. This is a good thing, since its said that a well-designed Business Activities should be proactive to be performant. It does not make any sense to wait until the end of the transaction for just to see that the transaction has failed if it can already be seen in the middle of the transac-

tion execution. Last important property is the the recording of all state transitions, including application state and coordination metadata.

Like AtomicTransaction, the BusinessActivity specification defines a couple of protocols for coordination [CCF+05c]. First one is BusinessAgreementWithParticipantCompletion (BAWPC). In this protocol a participant must know by itself when all work related to a given business activity has completed. After a participant has done its job, like completed some atomic transactions, it sends to the coordinator a `Completed` message. The the coordinator replies the final outcome of the protocol instance to the participant, being either a `Close` or `Compensate` notification. Upon receiving a `Close` notification a participant recognizes that the protocol instance will be complted succesfully and acknowledges that by responding with `Closed` message. If coordinator sends a `Compensate` message to the participant, he should return a `Compensated` message and start actions for executing compensation for the work performed previously. The coordinator will never know if the compensation was really performed or not. Actual compensation implementation is a back-end specific matter.

The coordinator may also send `Cancel` message to an participant to indicate that participant should cancel the work being done. Participants tell the coordinator about leaving the current business activity by sending `Exit` message. Failures are notified to the coordinator by sending `Fault` message containing an identifier of the cause of the fault. BusinessActivity specification does not specify how the coordinator should report the knowledge of failures or exits back to application. Most probably there is some kind of application-specific protocol used to handle communication from the coordinator to the application and vice versa. The message flow of BAWPC protocol is shown in the state diagram (Figure 3).

Furthermore the coordinator keeps track of the business activity state so that the state between coordinator itself and a participant stays the same. The coordinator may go backwards in states if it notices that some participant in one protocol instance is a state behind. When receiving inconsistent messages from the coordinator, participant should just ignore them. State tables in the protocol specification define what messages should be regarded as being consistent.

The second one of the protocols included in BusinessActivity specification is the BusinessAgreementWithCoordinatorCompletion (BAWCC) protocol. It is fundamentally the same as the BAWPC protocol but instead of the participant autonomously deciding to end its participation in the business activity he relies on

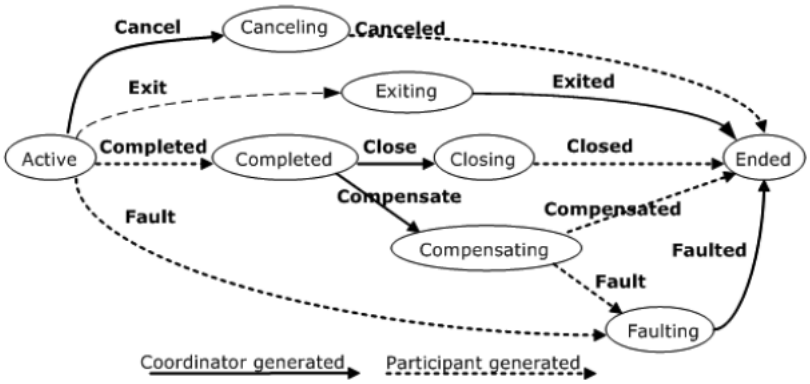

Figure 3: BusinessAgreementWithParticipantCompletion state diagram

its coordinator to tell when it has received all requests to do work within business activity.

If participant receives a `Complete` message from the coordinator it should complete its work. After receiving the message, the participant may issue a `Completed`, `Exit` or `Fault` message to the coordinator depending on the outcome. The coordinator reacts to these messages in the same manner as defined in the BAWPC protocol. The message flow of BAWCC is presented in the state diagram (Figure 4).

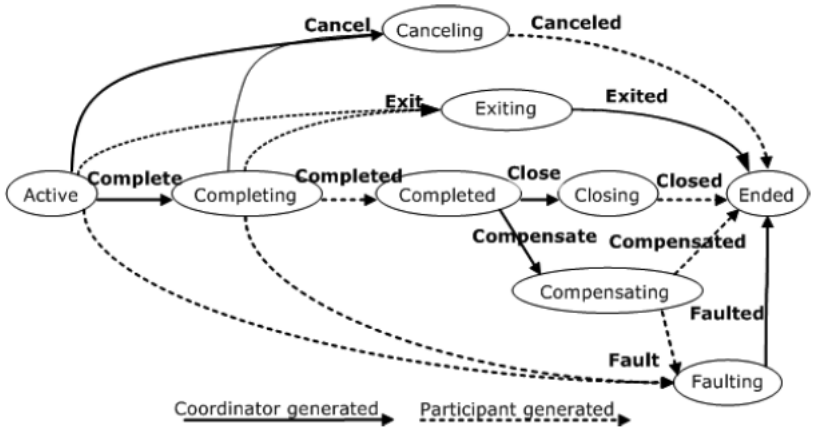

Figure 4: BusinessAgreementWithCoordinatorCompletion

3 Issues in WS-Tx specifications

Neto and Reverbel [NR08] noted one issue when doing an implementation of WS-C and WS-AT. WS-AT does not support the one-phase commit (1PC) optimization.

It requires the execution of the full 2PC protocol even if a distributed transaction involves only a single participant.

It has been also stated that WS-C and WS-BA violate the SOA paradigm by making assumptions about the internal structures of the participants. Proof-of-concept implementation of WS-C and WS-BA show that there is a tight coupling between coordinator, initiator and participant. A middleware required to deal with business messages prohibits a complete separation of concerns as it requires proprietary protocols between initiator and the coordinator. Erven et al [EHHZ07] define the Web Services-BusinessActivity-Initiator (WS-BA-I) Protocol as a remedy to this issue. WS-BA-I explicitly defines an interface between initiator and coordinator.

Choi et al [SHH+05] argue that commonly relaxed isolation in WS-Transactions is harmful. Resource locks in one participant are released once its jobs are completed without waiting for the completion of other operations. Those early unlocked resources may be accessed by other transactions, which may lead to spoiled data integrity and cause incorrect outcomes. In their paper, Choi et al present a mechanism for ensuring the consistent execution of isolation-relaxed WS transactions. The mechanism identifies transaction in inconsistent state with a notion of a completion dependency.

4 Conclusions

The strictness of ACID properties has been an emerging research topic for quite a long time. Different solutions to relax those properties have been presented during the years. Most of the solutions are overlapping in various areas. Web Service Transactions specifications is a one of the most promising solutions to the presented problem.

As being extensible, WS-Coordination can be used in virtually any kind of systems that need interoperability and coordination of different collaborators. I can see WS-AT and WS-BA as a good base for future developments in the area of transaction management. As being pointed out in a few papers, both transaction protocols have some flaws here and there but those problems will be tackled when implementations of these protocols are created.

References

CCF+05a Cabrera, L. F., Copeland, G., Feingold, M., Freund, R. W., Freund, T., Johnson, J., Joyce, S., Kaler, C., Klein, J., Langworthy, D., Little, M., Nadalin, A., Newcomer, E., Orchard, D., Robinson, I., Shewchuk, J. and Storey, T., Web service coordination (ws-coordination), August 2005. URL `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-tx/WS-Coordination.pdf`.

CCF+05b Cabrera, L. F., Copeland, G., Feingold, M., Freund, R. W., Freund, T., Johnson, J., Joyce, S., Kaler, C., Klein, J., Langworthy, D., Little, M., Nadalin, A., Newcomer, E., Orchard, D., Robinson, I., Storey, T. and Thatte, S., Web services atomic transaction (ws-atomictransaction), August 2005. URL `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-tx/WS-AtomicTransaction.pdf`.

CCF+05c Cabrera, L. F., Copeland, G., Feingold, M., Freund, R. W., Freund, T., Joyce, S., Klein, J., Langworthy, D., Little, M., Leymann, F., Newcomer, E., Orchard, D., Robinson, I., Storey, T. and Thatte, S., Web services business activity framework (ws-businessactivity), August 2005. URL `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-tx/WS-BusinessActivity.pdf`.

EHHZ07 Erven, H., Hicker, G., Huemer, C. and Zaptletal, M., The web services-businessactivity-initiator (ws-ba-i) protocol: an extension to the web services-businessactivity specification. *Web Services, IEEE International Conference*, Los Alamitos, CA, USA, 2007, IEEE Computer Society, pages 216–224.

FS02 Freund, T. and Storey, T., Transactions in the world of web services, part 1. web article., August 2002. URL `http://www.ibm.com/developerworks/webservices/library/ws-wstx1/`.

Gun03 Guntzel, K., Web services-based transactional workflows - advanced transaction concepts. *On The Move to Meaningful Internet Systems 2003: OTM 2003Workshops*. Springer Berlin / Heidelberg, 2003, pages 70–82.

NBCT06 Nezhad, H. R. M., Benatallah, B., Casati, F. and Toumani, F., Web services interoperability specifications. *Computer*, 39,5(2006), pages 24–32.

NR08 Neto, I. S. and Reverbel, F., Lessons learned from implementing ws-coordination and ws-atomictransaction. *ICIS '08: Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*, Washington, DC, USA, 2008, IEEE Computer Society, pages 367–372.

SHH+05 Seunglak, C., Hyukjae, J., Hangkyu, K., Jungsook, K., Su Myeon, K., Junehwa, S. and Yoon-Joon, L., Maintaining consistency under isolation relaxation of web services transactions. *Web Information Systems Engineering - WISE 2005*. Springer Berlin / Heidelberg, 2005, pages 245–257.