F A C H H O C H S C H U L E W E D E L

S E M I N A R A R B E I T

in der Fachrichtung

Wirtschaftsinformatik

Transaktionen in WebServices

Eingereicht von:	Tobias Ramin (Mat.-Nr. 2208) Launitzweg 7 20535 Hamburg Tel. (0 40) 22690642, Mobil: 0162-1313455 Email: wi2208@fh-wedel.de
Erarbeitet im:	7. Semester
Abgegeben am:	06. Dezember 2006
Referent (FH Wedel):	Prof. Dr. Sebastian Iwanowski Fachhochschule Wedel Feldstraße 143 22880 Wedel Tel. (04103) 8048-63 Email: iw@fh-wedel.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

1.1 Motivation

Web Services bieten eine Umgebung für verteilte Anwendungen über beliebige Netzwerke, vorzugsweise über das Internet. Sie verbinden zunehmend große Zahlen von Teilnehmern zu einer komplexen Struktur mit durchaus vielschichtigen Beziehungen untereinander. Streben mehrere Web Services die Erreichung eines gemeinsam anerkannten Ergebnisses an, welches auf jeden Fall konsistent ist, dann benötigen sie dazu transaktionelles Verhalten. Im Bereich von Web-Services-Architekturen kann dieses Verhalten besonders komplex werden, da hier die Teilnehmer unter Umständen nur über (SOAP-)Nachrichten miteinander kommunizieren können. In diesem Fall könnte der Nachrichtenaustausch, bedingt durch Netzwerkausfälle, jederzeit ausfallen.

1.2 Zielstellung

Das Ziel dieser Seminararbeit ist es einen Überblick über die existierenden Spezifikationen, die zur Umsetzung von transaktionellem Verhalten in Web Services dienen, zu geben. Die Ausarbeitung setzt auf den Seminarvortrag „WS-Coordination“ von Sven Bohnsack auf und geht nur kurz auf dessen wesentlichen Punkte ein.
Im Abschnitt 2 wird zunächst der Begriff Transaktion definiert und auf grundlegende Konzepte eingegangen. Dabei ist das ACID-Paradigma zu erläutern und das Konzept der verteilten Transaktionen vorzustellen. Weiterhin wird die Rolle von Transaktionen und die Transaktionsarten in Web Services nahe gebracht.
Der 3. Abschnitt beinhaltet die Spezifikationsfamilie Web Service Transaction Framework (WSTF). In den Unterkapiteln wird detailliert auf die 3 enthalten Spezifikationen eingegangen. Insbesondere auf die Spezifikationen WS-AtomicTransaction und WS-BusinessActivity zur Realisierung von Transaktionen.
Die Zusammenfassung der Seminararbeit und Empfehlungen im Umgang mit den vorgestellten Spezifikationen beschließen die Arbeit im Abschnitt 4.

2. Transaktionen

Transaktionen sind ein grundlegender Bestandteil beim Aufbau von zuverlässigen und verteilten Anwendungen. Sie versuchen stets ein Informationssystem von einem konsistenten Zustand in einen anderen, ebenfalls konsistenten Zustand zu überführen. Im Kontext von Web Services kann man Transaktionen als einen Mechanismus definieren, der sicherstellt, dass alle Teilnehmer zu einem gemeinsam anerkannten Resultat gelangen[1].
Die folgenden Unterkapitel befassen sich zunächst mit existierenden Transaktionskonzepten und der theoretische Überführung in Web-Services-Umgebungen.

2.1 Existierende Konzepte

2.1.1 ACID-Paradigma

Das ACID-Paradigma wurde ursprünglich für Transaktionen in lokalen Systemen, insbesondere im Datenbank-Umfeld, entwickelt. Derzeit erfüllen nahezu alle Datenbank-Transaktionen die ACID-Eigenschaften, weshalb das Konzept als grundlegend für andere Transaktionen betrachtet werden kann[2]. Folgende 4 Eigenschaften werden von einer atomaren/ ACID-konformen Transaktion gefordert[3]:

Atomicity: Die Transaktion ist eine nach außen unteilbare Operation in der die Menge von Arbeitsschritten entweder ganz oder gar nicht ausgeführt werden.

Consistency: Übergang von einem konsistenten Zustand in einen anderen, ebenfalls konsistenten Zustand.

Isolation: Die Zwischenstände einer noch nicht abgeschlossenen Transaktion sind nicht sichtbar für andere Transaktionen. Dies wird durch Sperren von Ressourcen ermöglicht. Ist die Transaktion beendet, ist der Zugriff auf die gesperrten Daten wieder ermöglich.

Durability: Nach erfolgreichem Abschluss der Transaktion müssen die Ergebnisse beständig gemacht werden.

Bei allen wichtigen Datenbanksystemen wird die strikte Erfüllung der ACID-Eigenschaften gefordert, dennoch gibt es Anwendungsszenarien bei denen dies expliziert nicht erwünscht wird. Ein mögliches Szenario stellen die verteilten Transaktionen dar, auf die im nächsten Abschnitt näher eingegangen wird.

2.1.2 Verteilte Transaktionen

In verteilten und unabhängigen Systemen, die in einem Netzwerk lose miteinander gekoppelt sind, finden verteilte Transaktionen statt. Das heißt, dass mehrere Prozesse an einer Transaktion beteiligt sind, die ein gemeinsam anerkanntes Resultat erreichen wollen.

---

[1] Vgl. S.Weerawerana u.a. (2005, Seite 218.)
[2] Vgl. W.Dostal u.a. (2005, Seite 232)
[3] Vgl. Prof. C. Eckert (Seite 2 f.)

Die Teilnahme mehrerer Anwendungen an einer Transaktion erfordert zwei Arten von Sub-Systemen: mehrere Teilnehmer (Participants) und ein Koordinator (Coordinator). Der Teilnehmer implementiert einen Teil der Geschäftslogik und führt eine Untermenge von Operationen der Transaktion aus. Der Koordinator kontrolliert die Erzeugung einer neuen Transaktion, sowie den Betritt eines Beteiligten zu einer bereits laufenden Transaktion. Die Transaktions-Logik ist in der Regel bereits in dem Transaktions-Framework implementiert.[4]
Zur Koordination der verteilten Anwendungen werden Koordinationsprotokolle benutzt. Die Implementierung dieser Protokolle ist abhängig von der Art der verteilten Transaktion. Man unterscheidet zwischen kurz- und langlaufenden verteilen Transaktionen. Bei kurzlaufenden Transaktionen wird häufig das 2-Phase-Commit-Protokoll (2PC) benutzt. Das Protokoll kommt zum Ende einer Transaktion zum Einsatz und bietet den Teilnehmern die Möglichkeit dem Transaktions-Resultat zu zustimmen und über das Ergebnis instruiert zu werden. Bei der Implementation von langlaufenden und verteilten Transaktionen stehen mehrere „erweiterte Transaktionsmodelle“ zur Verfügung. Das Konzept der geschlossenen und offen verschachtelten Transaktionen. Eine verschachtelte Transaktion, wie sie in der Abbildung 1 zu sehen ist, besteht aus mehreren Untertransaktionen (scops) und ist wie eine baumartige Struktur aufgebaut. Eine Transaktion ist offen verschachtelt, sobald eine Untertransaktion einen Commit ausführen darf, ohne dass diese auf die Wurzel oder andere Untertransaktionen warten muss. Sie ist hingegen geschlossen, wenn die Untertransaktionen erst nach dem Abschluss der übergeordneten Transaktion ihre Teiloperation beständig machen darf[5].

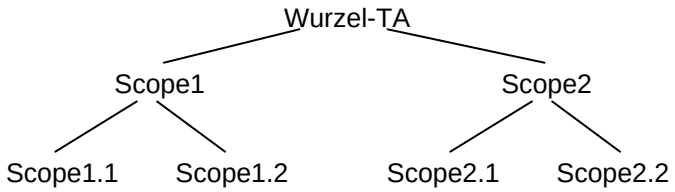


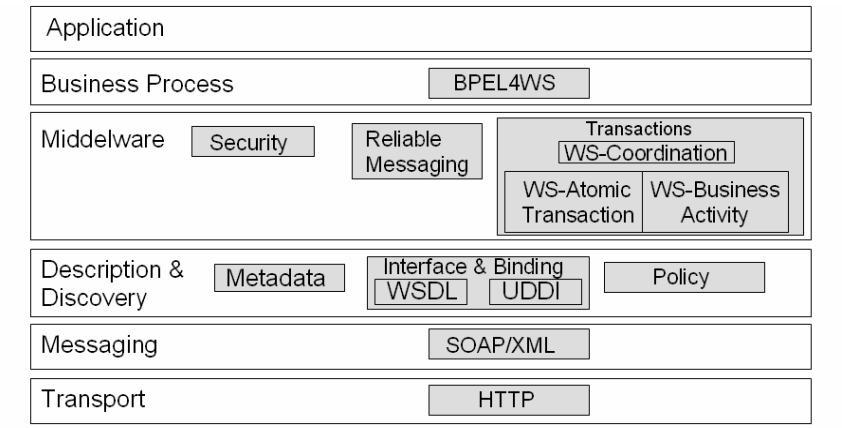
Abbildung 1 - Konzept der verschachtelten Transaktionen

Wie die Transaktionsmodelle für die Anforderungen für Web Services angepasst und welche Modelle verwendet werden, wird in den folgenden Kapiteln erläutert.

[4] Vgl. W.Dostal u.a. (2005, Seite 235 f.)
[5] Vgl. W.Dostal u.a. (2005, Seite 239 f.)

2.2 Transaktionen in Web Services

In einer Web-Service-Umgebung werden transaktionale Anwendungen durch die Verbindung von mehreren Web Services erstellt, die jeweils eigenständig Ressourcen manipulieren und Teilnehmer eines anerkannten koordinierten Ergebnisses sind können. Die Services, die i.d.R. Einzelteile einer Anwendung darstellen, sind lose gekoppelt und verteilt über unabhängige Systeme. Bedingt durch die Implementierung mehrerer Geschäftlogiken innerhalb eines Web Services und durch manuelle Eingriffe seitens der Benutzer werden vorwiegend langlaufende Transaktionen verwendet. Dennoch wird mit kurzlaufenden atomaren Transaktionen gearbeitet, wie im folgenden Unterkapitel näher erläutert wird.



Quelle: Weerawarana

Abbildung 2 - Web Service Modell[6]

Abbildung 2 zeigt die Einordnung von Transaktionen in das Web-Service-Modell. Transaktionen unterstützten hierbei den middelware-service, in dem es die Kern-Web-Service-Technologien wie SOAP, WSDL und UDDI unterstützt und dadurch eine vertrauensvolle Umgebung ermöglicht.

Ein wesentlicher Unterschied zwischen einem Web-Service- und Datenbank-Umfeld stellt die Definition von (Meta-)Datenstrukturen und die Festlegung von Standard-Operationen dar. In Web Services existieren keinen Operationen, wie z.B. ein Insert oder Delete, in denen entsprechende Abhängigkeiten der Daten berücksichtigt werden. Eine WSDL-Operation kann eine Datenbank-Operation aber auch eine Operation, bei der z.B. ein Brief an einen Kunden

[6] Übernommen aus S.Weerawerana u.a. (2005)

verschickt wird, sein.[7] Die im Folgenden erläuterten Transaktionen machen dennoch ein koordiniertes Arbeiten mit verteilten Web Services möglich.

2.2.1 Atomic Transaction

Atomic Transactions im Kontext von Web Services sind verteilte ACID-konforme Transaktionen die für kurzlebige in sicheren, vertrauensvollen und verteilten Umgebungen durchgeführte Aktivitäten geeignet sind. Diese kurzlaufenden Transaktionen werden in der Spezifikation WS-AtomicTransaction umgesetzt.[8]

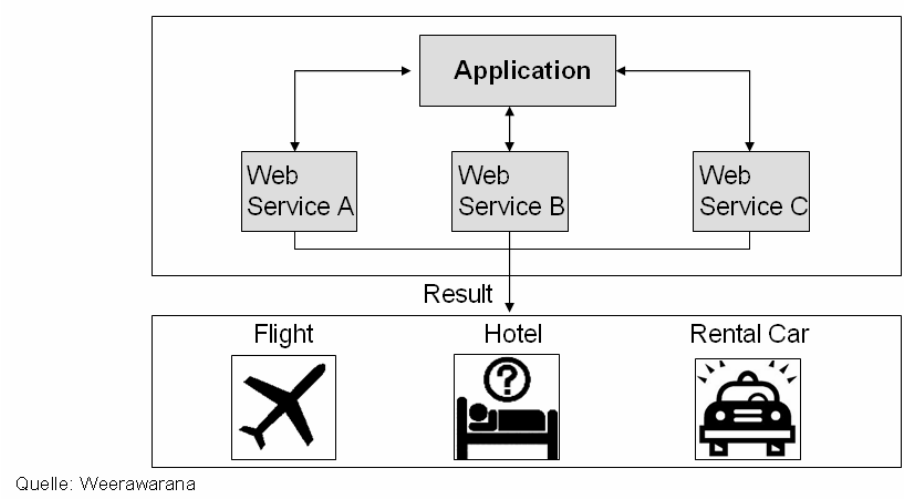


Abbildung 3 - Beispiel für eine Atomic Transaction[9]

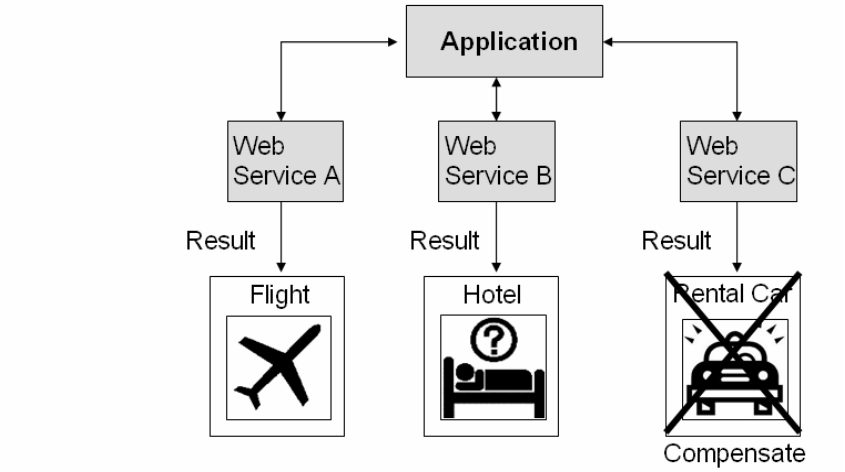
Abbildung 3 zeigt ein Beispiel für eine Atomic Transaction im Web-Service-Umfeld. Die Anwendung stellt eine Software dar, die über 3 Web Services Reservierungen durchführt. Bedingt durch die strikte Einhaltung der ACID-Eigenschaften ist eine Aufteilung der 3 Teiloperationen nicht möglich. Das führt dazu, dass nur bei erfolgreicher Durchführung aller Web Services die Ergebnisse beständig und sichtbar für andere Anwendungen gemacht werden.

2.2.2 Business Transaction

Lang laufende Prozesse, die - verteilt über ein Netzwerk - Web Services einbinden und womöglich in unsicheren Umgebungen ausgeführt werden, können sich nicht an die strikten ACID-Bedingungen von klassischen Transaktions-Protokollen halten. Geschäftanwendungen erfordern lange Bearbeitungszeiten, die Fähigkeit Fehlersituationen entsprechend zu behandeln und zu entscheiden, welche Web Services in das Gesamtergebnis mit einbezogen werden. Ein

[7] Vgl. G. Alonso (2004, Seite 226)
[8] Vgl. G. Alonso (2004, Seite 227)
[9] Übernommen aus S.Weerawerana u.a. (2005)

Ansatz, um diese Anforderungen zu befriedigen, ist die Auflockerung und Aufweichung der ACID-Eigenschaften (Atomicity und Isolation) und das Einführen von so genannten Kompensations-Mechanismen. Das heißt, dass alle Teilaktionen der teilnehmenden Web Services unabhängig von der Gesamtaktion sind und ihre Arbeitsschritte sofort und permanent durchführen können. Dadurch werden jegliche Teilergebnisse der Web Services sichtbar bevor die Gesamttransaktion abgeschlossen ist. Treten während der Verarbeitung Fehler auf und die Transaktion muss rückgängig gemacht werden, dann führt der Web Service eine Kompensations-Methode aus, die den Zustand, der vor dem Ausführen der Teilschritte bestand, wieder herstellt. Die Prozesse zum Wiederherstellen des alten konsistenten Zustandes sind abhängig von der Geschäftlogik und werden in der Kompensations-Methode des jeweiligen Web Service implementiert.
Unabhängig davon, wie Web Services Transaktionen oder Kompensations-Mechanismen implementieren, werden Standard-Protokolle gebraucht, die es den beteiligten Web Services ermöglichen, bei einem Fehler eines Web Services während der Verarbeitung sofort alle anderen zu determinieren und die Kompensations-Operationen aufzurufen. In dem folgenden Kapitel wird unter Anderem ein für diese Anforderungen spezifiziertes Protokoll mit dem Namen WS-BusinessActivity vorgestellt.



Quelle: Weerawarana

Abbildung 4 - Beispiel für eine Business Transaction[10]

Abbildung 4 zeigt das bereits in Abbildung 3 erläuterte Beispiel unter Verwendung einer Business Transaction. Es wird deutlich, dass die Teilaktionen unabhängig von der Gesamtaktion sind. Dadurch werden Änderungen sofort, permanent und sichtbar durchgeführt ohne auf das Ende der Gesamttransaktion zu warten. Entscheidet sich die Anwendung anhand der implementierten Geschäftlogik eine der bereits erfolgreich durchgeführten Reservierungen rückgängig zu machen, dann wird eine Kompensationsroutine aufgerufen.

[10] Übernommen aus S.Weerawerana u.a. (2005)

3. Web Service Transaction Framework

Das Web Service Transaction Framework ist eine Spezifikations-Familie, die von IBM, Microsoft und BEA entwickelt wurde. Sie bietet den Beteiligten einer verteilten Transaktion die Möglichkeit einer standardisierten Kommunikation untereinander. Das Framework besteht aus den Spezifikationen WS-Coordination, WS-AtomicTransaction und WS-BusinessActivity die in den folgenden Kapiteln genauer erläutert werden.

3.1 WS-Coordination

Die Spezifikation WS-Coordination liefert ein Framework, welches einen Rahmen zur Koordination von Web Services bereitstellt. Es unterstützt die Verwaltung, das Aktualisieren und die Verteilung von Context-Informationen an teilnehmende Web Services. Context-Informationen dienen der Identifikation von Teilnehmern und deren Operationen in einer Transaktion. Der Informationsgehalt einer Context-information hängt von der Komplexität einer Aktivität ab.[11]Eine Aktivität ist definiert als eine Berechnung, die als ein Prozess über einem oder mehrere Web Services durchgeführt wird.[12]
Die Hauptaufgabe des WS-Coordination Frameworks ist folglich die Erstellung von CoordinationContext-Elementen und die Weitervermittlung an partizipierende Web Services in der verteilten Transaktion. Für diese 2 Arten von Aufgaben stellt das Framework einen coordination service zur Verfügung, der 3 weitere Services, die im Folgenden genauer dargelegt werden, kontrolliert.

3.1.1 WS-Coordination Services

Activation Service: Der Service erstellt eine neue Aktivität und gibt den CoordinationContext zurück.
Registration Service: Der Registration Service ermöglicht jedem Teilnehmer einer Aktivität sich an einem unterstützten coordination protocol zu registrieren.
Protocol Services: Diese Services stellen die Implementierung der in den coordination types spezifiziert Protokolle dar und ermöglichen den Austausch von Nachrichten zwischen dem Koordinator und den Beteiligten.

3.1.2 Coordination Types

Jeder Koordinator bezieht sich auf einen Koordinations-Typen. Dieser Typ spezifiziert die grundlegende Logik einer Aktivität bzgl. der Verwaltung der Context-Informationen.

---

[11] Vgl. T. Erl (2005, Seite 177)
[12] Vgl. S.Weerawerana u.a. (2005, Seite 230)

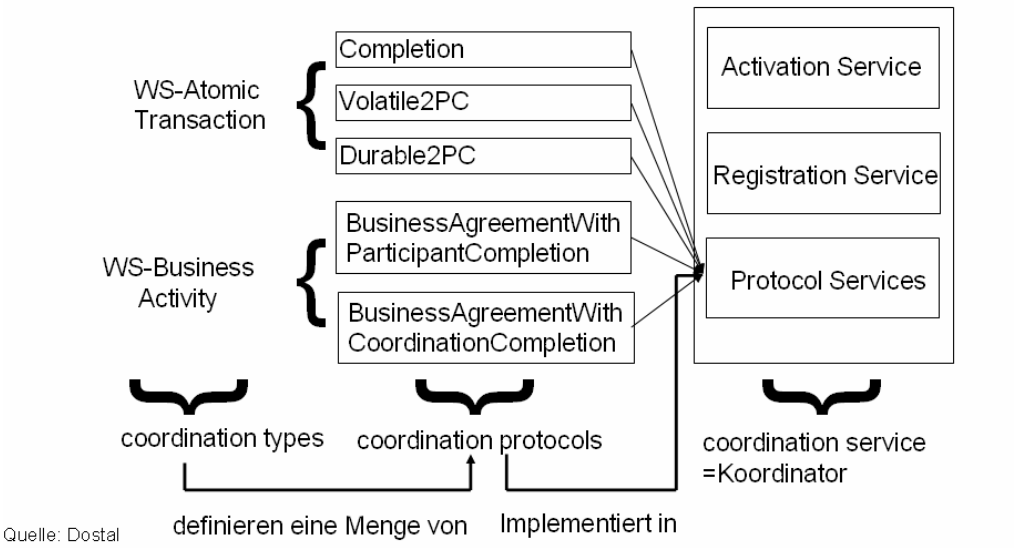


Abbildung 5 - WS-Coordination-Übersicht[13]

Abbildung 5 fasst die 3 Services der WS-Coordination-Spezifikation zusammen. Dabei wird deutlich, dass ein Coordination type eine Menge von coordination protocols definiert, die einen Typen einzigartig machen und aus bestimmten Verhaltensregeln bestehen.[14]Die definierten coordination protocols werden im Protocol Service implementiert.

In den folgenden beiden Abschnitten werden die beiden coordination types des WSTF vorgestellt – namentlich WS-AtomicTransaction und WS-BusinessActivity.

3.2 WS-AtomicTransaction

Die Spezifikation WS-AtomicTransaction ist ein coordination type, der in dem erweiterten Koordination Framework (spezifiziert im WS-Coordination) benutzt wird. Die Spezifikation besteht aus 3 coordination protocols[15]. Die Protokolle erlauben ein ACID-konformes Verhalten der Transaktion bedingt durch die Steuerung von kurz lebigen Aktivitäten in sicheren Umgebungen, den Einsatz des 2PC-Protokoll und der „all-or-nothing“-Eigenschaft.[16] Diese besagt, dass beim Auftreten eines oder mehrerer Fehler(-s) bei der Verarbeitung der teilnehmenden Web Services die gemachten Änderungen rückgängig gemacht werden (analog zur Atomaritäts-Bedingung). Updates sind außerhalb der Transaktion nicht sichtbar und die Ressourcen sind während der Bearbeitungszeit der Gesamttransaktion gesperrt.

[13] Übernommen aus W.Dostal u.a. (2005, Seite 245)
[14] Vgl. T. Erl (2005, Seite 180)
[15] Vgl. IBM
[16] Vgl. S.Weerawerana u.a. (2005, Seite 229)

WS-AtomicTransaction besteht aus den drei coordination protocols Completion, Durable2PC und Volatile2PC. Jedes Protokoll definiert eine Menge von Zuständen und Zustandsübergängen, die in den folgenden Unterkapiteln genauer untersucht werden.

3.2.1 Completion-Protokoll

Das Completion-Protokoll ist ein Steuerprotokoll und wird zum Beenden einer Transaktion verwendet. Es ermöglicht einer Anwendung dem Koordinator mitzuteilen, dass dieser die aktuelle Transaktion beenden (Commit- oder Rollback-Nachricht) und die Anwendung über das Ergebnis der Transaktion informieren soll[17]. Abbildung 6 zeigt das Zustandsdiagramm des Completion-Protokolls.
Ein möglicher Ablauf könnte folgendermaßen beschreiben werden:
Eine Anwendung registriert sich für das Completion-Protokoll beim registration service des Koordinators. Nach erfolgreicher Registrierung und Abschluss einer Menge von Teilprozessen, signalisiert die Anwendung den Abschluss der Transaktion entweder durch eine Commit- oder Rollback-Nachricht an den Koordinator.
Im Completing-Zustand entscheidet der Koordinator, ob die Transaktion mit einem commit oder rollback beendet werden soll und sendet der Anwendung entsprechend eine Commited- oder Aborted-Message.

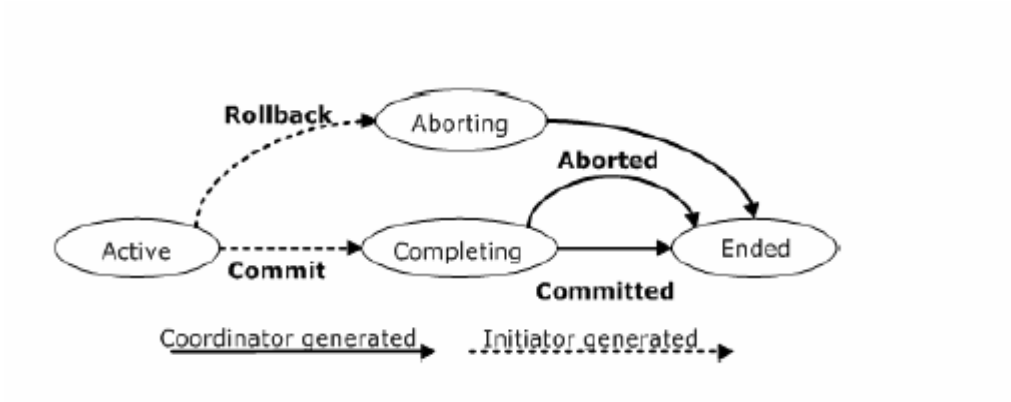


Abbildung 6 - Zustandsdiagramm des Completion-Protokolls[18]

[17] Vgl. L. F. Cabrera u.a. [1] (Seite 6)
[18] Übernommen aus L. F. Cabrera u.a. [1]

3.2.2 2PC-Phasen und Zustandsdiagram

Das 2PC-Protokoll ist Grundlage für die Koordinationsprotokolle Volatile2PC und Durable2PC. Aus diesem Grund wird in diesem Kapitel darauf eingegangen.
Der Verlauf des Protokolls ist in 2 Phasen aufgeteilt. Diese sind im Einzelnen:
Phase 1 - Vorbereitung: Jeder Teilnehmer wird durch die Prepare-Anfrage des Koordinators aufgefordert seinen Teil der Arbeitschritte zu beenden und eine Wahl (Vote) zum Transaktionsresultat abzugeben. Diese Wahl kann positiv (Prepared-Nachricht) bei erfolgreicher oder negativ (Aborted-Nachricht) bei fehlgeschlagener Verarbeitung ausfallen.
Phase 2 – Ergebnis: Nachdem der Koordinator alle Prepared- und möglicherweise Aborted-Nachrichten gesammelt hat, trifft er die Entscheidung für das Ergebnis der Gesamttransaktion. Sind in der ersten Phase keine Fehler aufgetreten, ist die Transaktion erfolgreich verlaufen und der Koordinator sendet eine Commit-Nachricht an alle Teilnehmer, damit diese ihre durchgeführten Prozesse persistieren können. Bei Abschluss des Persistierungsprozesses senden die beteiligten Web Services eine Commited-Nachricht zur Bestätigung. Treten während der ersten Phase Fehler auf, dann ist die Transaktion fehlgeschlagen und der Koordinator sendet eine Rollback-Nachricht an alle verbleibenden Teilnehmer der Transaktion.
Weiterhin werden folgende Nachrichten im Protokoll definiert:
ReadOnly: Antwortet ein Teilnehmer in der 1. Phase mit ReadOnly, somit spricht er sich für einen Commit der Transaktion aus und beendet die Teilnahme an der Transaktion (vergisst diese).
Aborted: Sendet ein Beteiligter im Zustand Preparing eine Aborted-Nachricht, voted er gegen einen Commit und verlässt die Transaktion[19].
Das 2PC-Protokoll dient als Sperr-Protokoll um die Isolation-Bedingung in ACID-Umgebungen zu erfüllen. Das heißt nach Abschluss der Phase 1 sperrt der Teilnehmer die Ressourcen solange bis die Phase-2-Nachricht des Koordinators empfangen wird, dadurch ist es für andere Transaktionen unmöglich diese Ressourcen zu benutzen.[20]

[19] Vgl. L. F. Cabrera u.a. [1] (Seite 7)
[20] Vgl. S.Weerawerana u.a. (2005, Seite 223)

Abbildung 7 zeigt das Zustandsdiagramm der Volatile2PC- und Durable2PC-Protokolle.

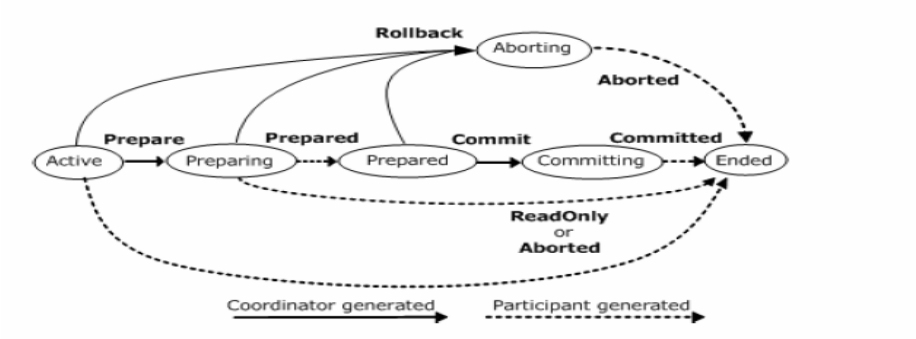


Abbildung 7 – Zustandsdiagramm des 2PC-Protokoll[21]

3.2.2.1 Volatile2PC-Protokoll

Web Services registrieren sich für das Koordinationsprotokoll Volatile2PC wenn diese flüchtige Daten verwalten[22]. Der Koordinator beginnt die Prepare-Phase mit allen registrierten Volatile2PC-Teilnehmern und signalisiert dadurch den Abschluss der Transaktion. In dieser Phase können die Web Services ausstehende Prozesse durchführen bevor sie die Phase mit einem Prepared, ReadOnly oder einer Fehlermeldung beenden[23].

3.2.2.2 Durable2PC-Protokoll

Zusätzlich zum Volatile2PC-Protokoll unterstützt WS-AtomicTransaction das Durable2PC-Protokoll. Das Protokoll ist definiert für Web Services, die mit permanenten Daten arbeiten. Nach dem die Anwendung eine Commit-Nachricht im Rahmen des Completion-Protokolls versendet hat und die Prepare-Phase der Volatile2PC-Teilnehmer erfolgreich abschlossen wurde, beginnt der Koordinator die Prepare-Phase für Durable2PC-Beteiligte.

[21] Übernommen aus L. F. Cabrera u.a. [1]
[22] Vgl. T. Erl (2005, Seite 188)
[23] Vgl. S.Weerawerana u.a. (2005, Seite 240 ff.)

3.3 WS-BusinessAcivity

Die Spezifikation WS-BusinessActivity ist der zweite coordination type, der auf dem WS-Coordination Framework aufbaut und definiert die 2 coordination protocols Business Agreement With Participant Completion und Business Agreement With Coordination Completion. BusinessActivity wird im Gegensatz zur AtomicTransaction hauptsächlich zur Abwicklung von langlaufenden, asynchronen Prozessen verwendet, ohne Ressourcen über einen längeren Zeitraum zu blockieren. Da BusinessActivities in den meisten Fällen AtomicTransactions zu einer Geschäftsaktivität aggregieren, darf eine Ressource innerhalb einer AtomicTransaction exklusiv für einen kurzen Moment gesperrt werden[24].
Ein weiterer signifikanter Unterschied zur WS-AtomicTransaction ist die Behandlung von Fehlern innerhalb einer Transaktion. Die Implementierung des Konzepts der open nested transactions ermöglicht der BusinessActivity alle Aktivitäten in einzelne Scops zu organisieren. Die dadurch entstehende Schachtelung und Hierarchisierung ermöglicht ein nicht-atomares Ergebnis, da ein Eltern-Scope auswählen kann welche Kinder (Kind-Scopes) in das Gesamtergebnis miteinbezogen werden.
Beendet ein Kind-Scope seinen Prozess, signalisiert es dem Eltern-Scope, dass die Effekte des bereits ausgeführten Prozesses später kompensiert werden könnten[25]. Obwohl die BusinessActivity die ACID-Bedingungen nicht strikt einhält, sorgen Kompensations-Mechanismen für eine System-Konsistenz[26]. Diese ermöglichen es einem Web Service die Auswirkungen einer bereits erfolgreichen Teiloperation (in den meisten Fällen eine AtomicTransaction, die sich im Ende-Zustand befindet) wieder rückgängig zu machen. Weiterhin ermöglichen die WS-BusinessActivity-Protokolle den Kind-Scopes das unverzügliche Melden von Fehlern ohne dass der Koordinator diese dazu explizit auffordert. Diese Besonderheit erlaubt es dem exception handler der Anwendung bei einer erfolglosen Operation den weiteren Prozessablauf anzupassen, ohne bis an das Transaktionsende warten zu müssen[27].

3.3.1 BusinessAgreementwithParticipantCompletion-Protokoll

Während der Lebensdauer einer BusinessActivity wechselt der Koordinator (in diesem Fall der business activity coordinator) und die Teilnehmer eine Reihe von Zuständen. Abb. 3 zeigt die Zustände und Zustandsübergänge des Protokolls, von denen im Folgenden die Wichtigsten genauer dargestellt werden.

[24] Vgl. W.Dostal u.a. (2005, Seite 247)
[25] Vgl. S.Weerawerana u.a. (2005, Seite 242 f)
[26] Vgl. S.Weerawerana u.a. (2005, Seite 242)
[27] Vgl. S.Weerawerana u.a. (2005, Seite 243)

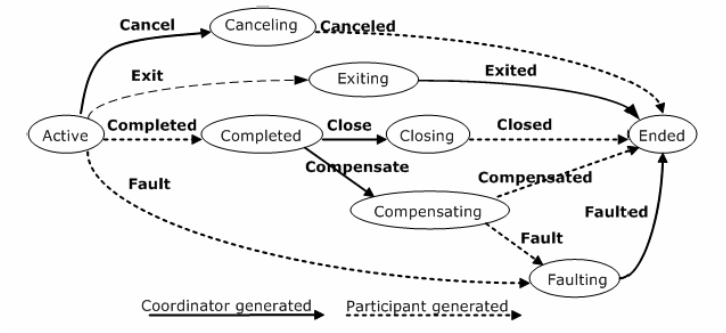


Abbildung 8 – Zustandsdiagramm BusinessAgreementwithParticipantCompletion[28]

Registriert sich eine Kind-Aktivität als ein Teilnehmer für das BusinessAgreementwithParticipantCompletion-Protokoll, befindet es sich im Active-Zustand. Durch eine Completed-Nachricht zeigt der Beteiligte an, dass er die Verarbeitung abgeschlossen hat und wechselt in den Completed-Zustand. In diesem Fall sollte der Beteiligte einen Kompensations-Mechanismus bereitstellen. Der Koordinator antwortet daraufhin mit einer Close-Nachricht, vorausgesetzt die Business-Aktivität war erfolgreich. Hingegen sendet er eine Compensate-Nachricht, wenn die erfolgreichen Prozesse durch eine Kompensations-Routine wieder rückgängig gemacht werden sollen.
Empfängt die Kind-Activity eine Compensate-Nachricht, dann ist dies nicht mit einem Abbruch oder Rollback einer AtomicTransaction zu vergleichen bei der bereits erfolgreich durchgeführte Schritte einfach abgebrochen werden, als wenn nie etwas passiert wäre. Es bedarf eines Kompensations-Mechanismus, der eine Reihe von weiteren Teilprozessen beinhalten kann. Die Stornierung einer Reservierung wäre ein Beispiel für eine Kompensations-Operation.
In den Zustandsübergängen Compensate und Compensated liegt ein wesentlicher Unterschied im Vergleich zu AtomicTransactions. Die Zustandsübergänge können auch noch nach einem erfolgreichen Completed der Transaktion ausgeführt werden. Befindet sich jedoch eine AtomicTransaction in dem gleichwertigen Zustand Commited, ist es nicht mehr möglichen diesen Zustand automatisch rückgängig zu machen[29].
Tritt während der Verarbeitung der Child-Activity oder der Kompensations-Methode ein Fehler auf, dann sendet die Aktivität eine Fault-Nachricht. Die nächste Protokoll-Nachricht des Koordinators ist eine Faulted-Nachricht, die dem Teilnehmer (Activity) signalisiert, dass der Koordinator den Fehler zur Kenntnis genommen hat und keine weiteren Aktionen mehr notwendig sind.
Soll die Verarbeitung der Child-Activity im Active-Zustand abgebrochen werden, dann sendet die Parent-Activity eine Cancel-Nachricht. Im Zustand Canceling bricht der Teilnehmer seine

[28] Übernommen aus L. F. Cabrera u.a. [2]
[29] Vgl. W.Dostal u.a. (2005, Seite 248)

gemachte Arbeit ab und sendet eine Canceled-Nachricht um die Protokoll-Instanz zu beenden[30].
Falls eine Child-Activity ihren Teilprozess beendet hat und nicht weiter am Protokoll teilnehmen muss (z.B. Read-only), dann sendet diese eine Exited-Nachricht und befindet sich im Exiting-Zustand. Mit einer Exited-Nachricht bestätigt der Koordinator das Verlasen des Teilnehmers.

3.3.2 BusinessAgreementwithCoordinatorCompletion-Protokoll

Das Business Agreement with Coordinator Completion Protokoll ist dem Business Agreement with Participant Completion Protokoll sehr ähnlich. Der einzige Unterschied besteht darin, dass die Child-Activity eine Complete-Nachricht vom Koordinator bekommen muss, bevor sie sich auf den Abschluss ihres Prozesses oder einer Kompensation vorbereitet. Dadurch hat die Parent-Activity die Möglichkeit dem Teilnehmer mitzuteilen, dass alle erwarteten Prozesse angefragt wurden[31]. Die Abb. 4 veranschaulicht das Business Agreement with Coordinator Completion Protokoll.

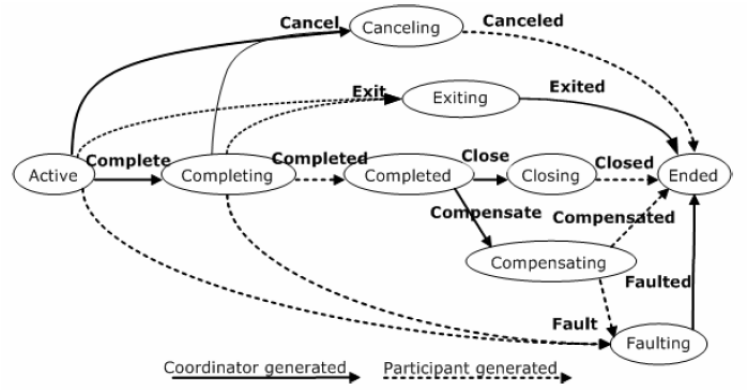


Abbildung 9 - Zustandsdiagramm BusinessAgreementwithCoordinationCompletion[32]

[30] Vgl. L. F. Cabrera u.a. [2] (Seite 7 f.)
[31] Vgl. G. Alonso u.a. (2004, Seite 232)
[32] Übernommen aus L. F. Cabrera u.a. [2] (Seite 9)

4. Schlussbemerkung

Die in der Seminararbeit vorgestellten Spezifikationen liefern Mechanismen zur Durchführung von Transaktionen im Web-Service-Umfeld. Sie ermöglichen das Einbinden von unabhängigen und verteilten Services und das Erreichen eines gemeinsam anerkannten Ergebnisses auch nach unvorhersehbaren Fehlern, wie beispielsweise Verbindungsabbrüche. Zur Koordination der eingebundenen Web Services werden entsprechende Koordinationsprotokolle verwendet, die in den Transaktionsspezifikationen WS-AtomicTransaction und WS-BusinessActivity spezifiziert werden. Die Verwendung der entsprechenden Spezifikationen richtet sich nach der Art der Aktivität. WS-AtomicTransaction wird demnach für kurz lebende und WS-BusinessActivity für lang lebende Aktivitäten benutzt.

Sowohl die Spezifikation WS-AtomicTransaction als auch WS-BusinessActivity enthalten keine Sicherheitsmechanismen, die Nachrichten vor dem Abfangen und Verfälschen schützen. Aus diesem Grund wird zu dem Einsatz der Spezifikation WS-Security, die im Verlauf der Vortragsreihe vorgestellt wird, geraten. Weiterhin sollten Sicherheitssysteme eingeführt werden, die unbefugten Zugriff auf Steuerprotokolle, wie beispielsweise das Completion-Protokoll, verwähren.

Literaturverzeichnis

Literaturverzeichnis

Wolfgang Dostal u.a. (2005): *Service-orientierte Architekturen mit Web Services, Konzepte – Standards - Praxis*, München: Spektrum

Gustavo Alonso u.a. (2004): *Web Services: Concepts, Architectures, and Applications*, Berlin, Heidelberg, New York: Springer.

Sanjiva Weerawarana u.a. (2005): *Web Services Platform Architecture*, Verlag: Prentice Hall PTR.

Thomas Erl (2005): *Service-Oriented Architecture, Concepts, Technology, and Design,* Verlag: Prentice Hall PTR.

Quellen im Internet:

IBM u.a.: *"Web Services Transactions specification"*, o.O., Internet http://www-128.ibm.com/developerworks/library/specification/ws-tx/, Stand 2005-08-16, Abruf 2006-11-12.

Luis Felipe Cabrera, Microsoft u.a. [1]: *"Web Service Atomic Transaction (WS-AtomicTransaction)"*,Spezifikation WS-AtomicTransaction, o.O., Internet: ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf, Stand August 2005 Version 1.0, Abruf 2006-11-12.

Luis Felipe Cabrera, Microsoft u.a. [2]: *"Web Service Business Activity Framework (WS-BusinessActivity)"*, Spezifikation WS-BusinessActivity, o.O., Internet: ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf, Stand August 2005 Version 1.0, Abruf 2006-11-12.

Prof. C. Eckert: *"Transaktionsverwaltung"*, Folie 5 aus der Lesung: *„FG Sicherheit in der Informationstechnik"*, TU Darmstadt, Internet: http://www.sec.informatik.tu-darmstadt.de/pages/lehre/SS02/bs2/folien/kapitel5.pdf
, Abruf 2006-11-17.