15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Alles Oder Nichts

Verteilte Transaktionen im heterogenen Umfeld

Thomas Haug

MATHEMA Software GmbH

Transaktionen – Agenda

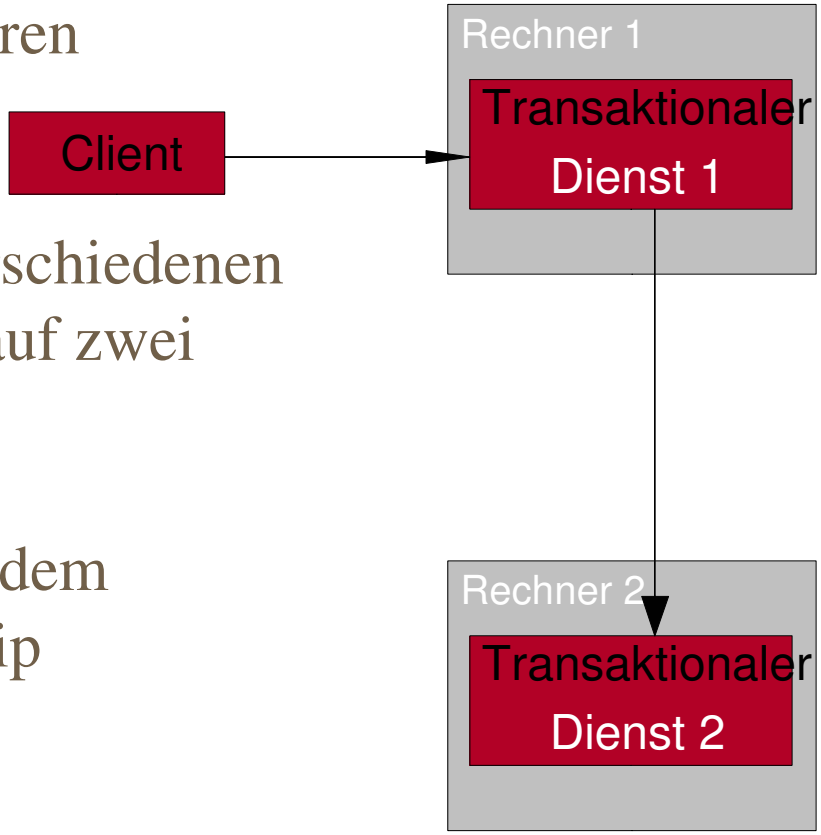
- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – Agenda

- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – Motivation

- Ein Client ruft einen Dienst auf, der seinerseits einen weiteren Dienst aufruft
- Die Dienste laufen auf verschiedenen Rechnern und „arbeiten“ auf zwei getrennten Datenbanken
- Beide Aufrufe sollen nachdem „Alles-oder-Nichts“ Prinzip ausgeführt werden



Transaktionen – Agenda

- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – ACID Eigenschaften (I)

- **Atomicity (Atomarität)**
  - Eine Transaktion wird entweder ganz oder gar nicht ausgeführt („Alles-oder-Nichts“)
  - Beispiel: Die Überweisung
    - Wenn ein Teil nicht durchgeführt werden kann, wie z.B. das Belasten des Kontos, dann soll auch die Gutschrift auf das Fremdkonto nicht möglich sein
- **Consistency (Konsistenz)**
  - Ein Transaktion arbeitet auf einem konsistenten Datenbestand und hinterlässt nach Abschluss der Transaktion wieder einen konsistenten Datenbestand

Transaktionen – ACID Eigenschaften (II)

- **Isolated (Isolation)**
  - Transaktionen sind isoliert (isolated), d.h. alle ihre Zwischenzustände sind für andere Transaktionen nicht sichtbar, d.h. konkurrierendes Lesen und Schreiben von ein und dem selben Datum durch mehrere Transaktionen wird verhindert
  - Aber in der (Datenbank) Praxis wird zu Gunsten der Lebendigkeit einer transaktionalen Anwendung diese Eigenschaft bewusst „aufgeweicht“

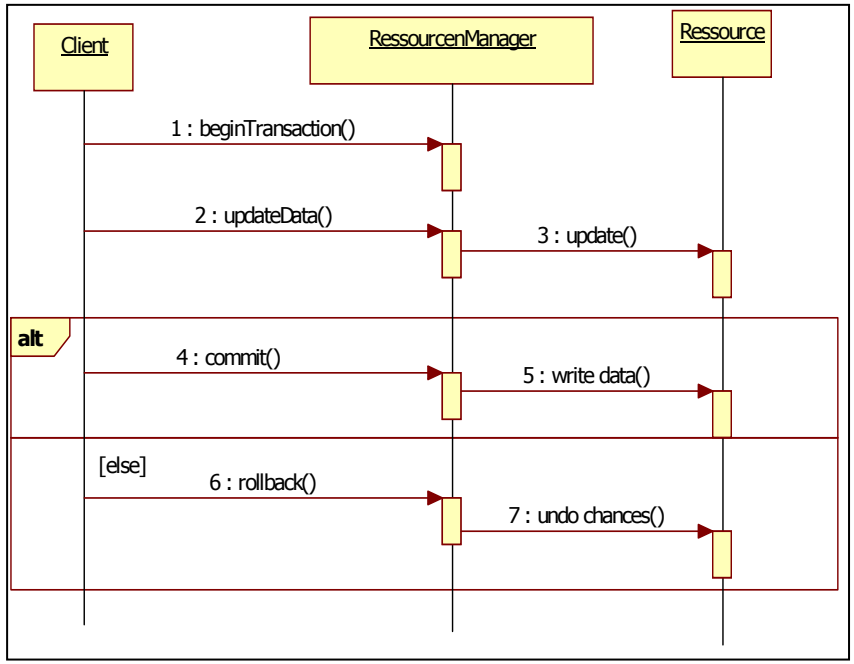
- **Durability (Dauerhaftigkeit)**:
  - Ergebnisse einer Transaktion sollen dauerhaft, z.B. auf einer Festplatte, gespeichert werden
  - Ergebnisse dürfen nicht durch nachfolgende Fehler (wie System crashes) verloren gehen

Transaktionen – Agenda

- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – lokale Transaktionen (1/2)

- Oftmals kommunizieren Anwendungen mit genau einer Ressource (z.B. Datenbank) und speichern dort ihre Daten ab.
- Hierbei kommen lokale ACID Transaktionen zum Einsatz
- In diesem Fall regelt üblicherweise die Ressource (eigentlich der zugehörige Ressourcenmanager) das Transaktionsmanagement

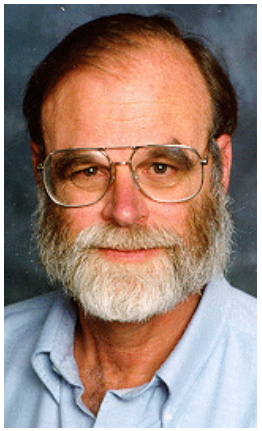


Transaktionen – lokale Transaktionen (2/2)

- Sollen mehrere Ressourcen innerhalb einer Transaktion Daten ändern bzw. speichern, so sind lokale Transaktionen nicht mehr ausreichend

Transaktionen – verteilte Transaktionen

- Sind mehr als eine Ressource in einer Transaktion beteiligt, so können die ACID Eigenschaften nur durch ein erweitertes Einigungsprotokoll (Commit Protokoll) ermöglicht werden
- Das Two-Phase-Commit (2PC) Protokoll ist das gängigste Commit Protokoll für verteilte Transaktionen
  - Wurde erstmalig von Dr. Jim Gray beschrieben
  - Wird bei XA, CORBA, Java, .Net und auch bei Web-Services eingesetzt
- Weitere Commit Protokolle
  - Three-Phase-Commit

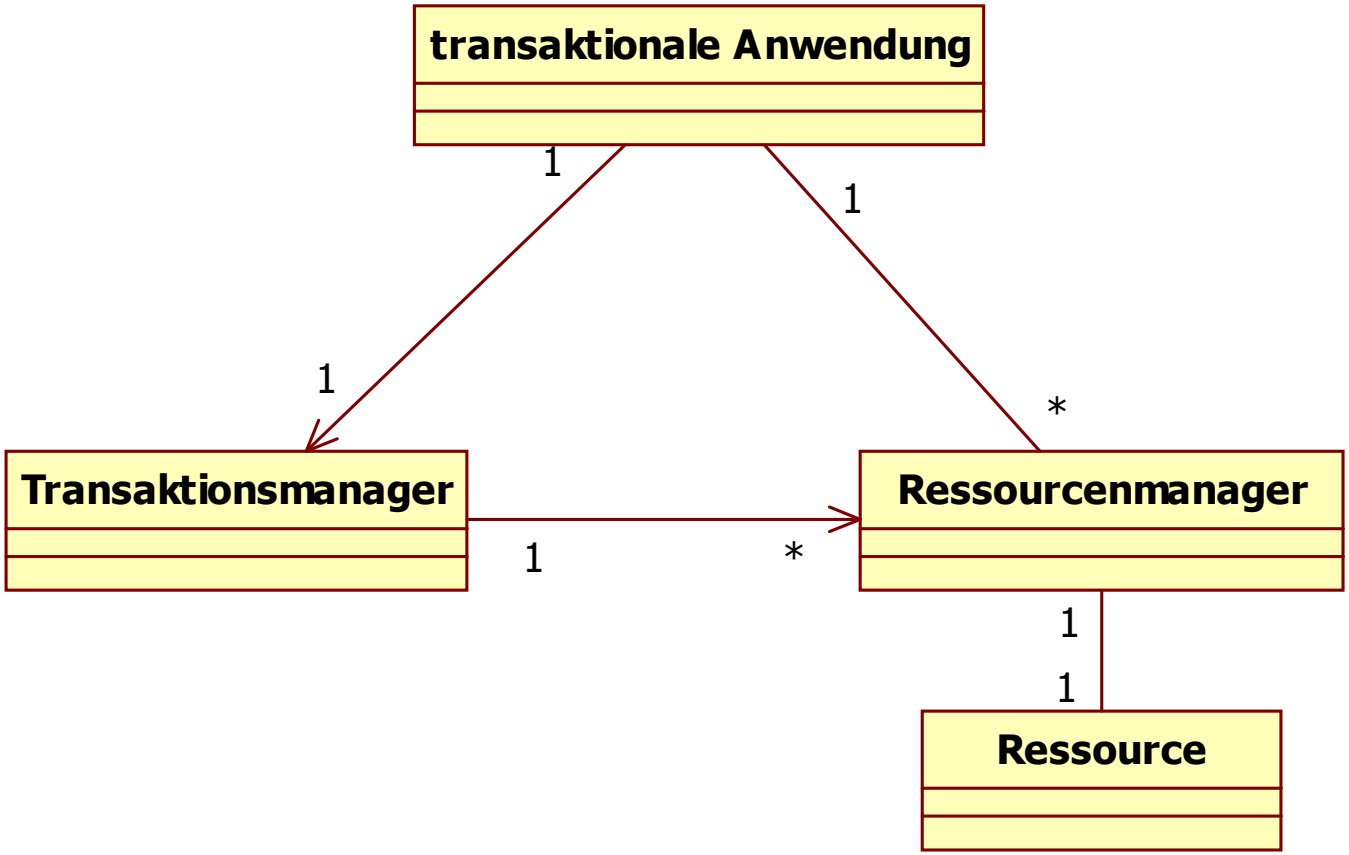


Transaktionen – Two-Phase-Commit

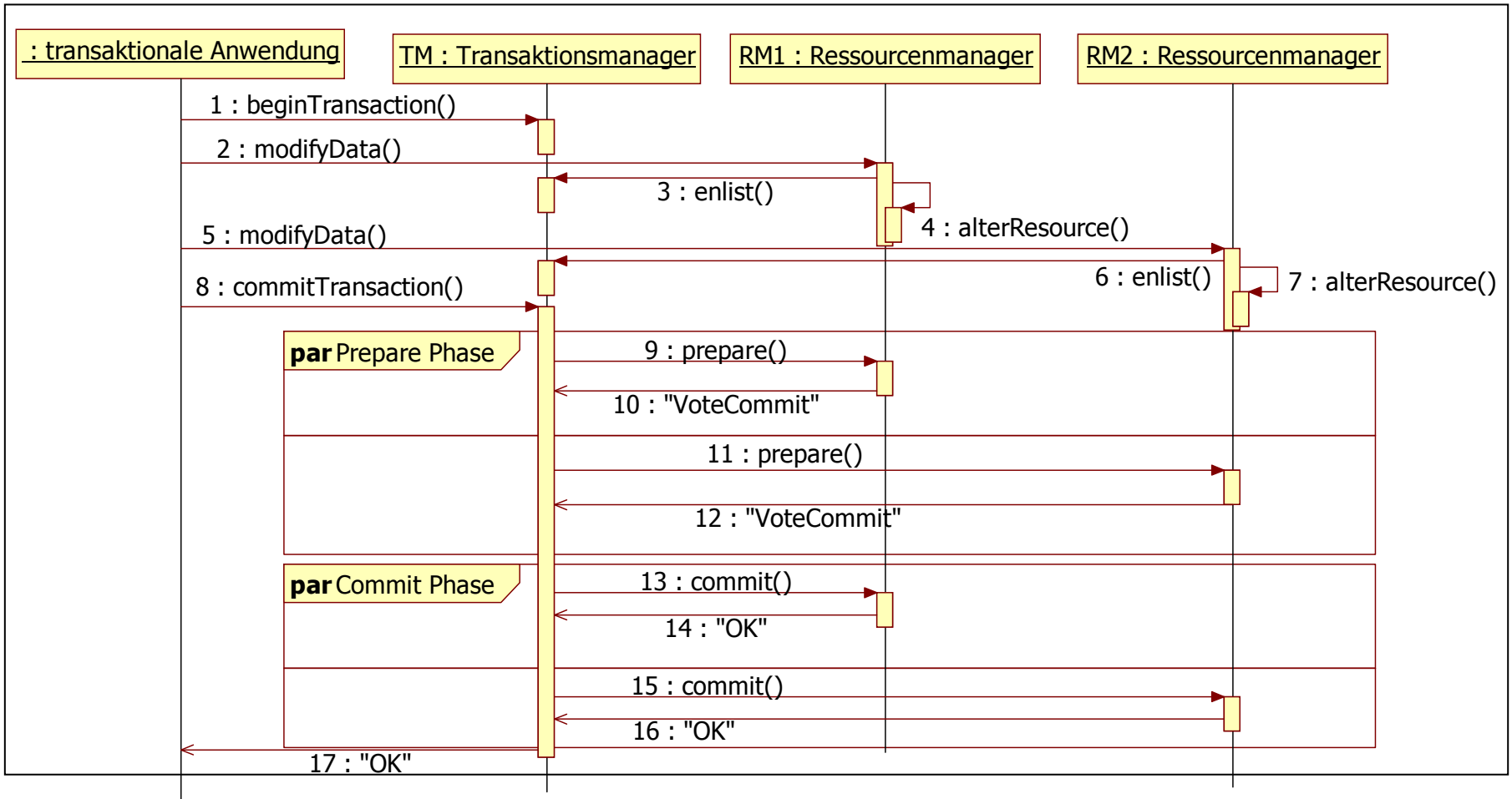
Teilnehmer

- Teilnehmer
  - **Transaktionale Anwendung**
    - Initiator der verteilten Transaktion
  - **Transaktionsmanager**
    - Koordiniert die verteilte Transaktion
    - auch als Transaktionskoordinator bezeichnet
  - **Ressourcenmanager**
    - verwaltet transaktionale Ressourcen und beteiligt sich an der verteilten Transaktion
    - Wird auch als Teilnehmer (Participant) bezeichnet

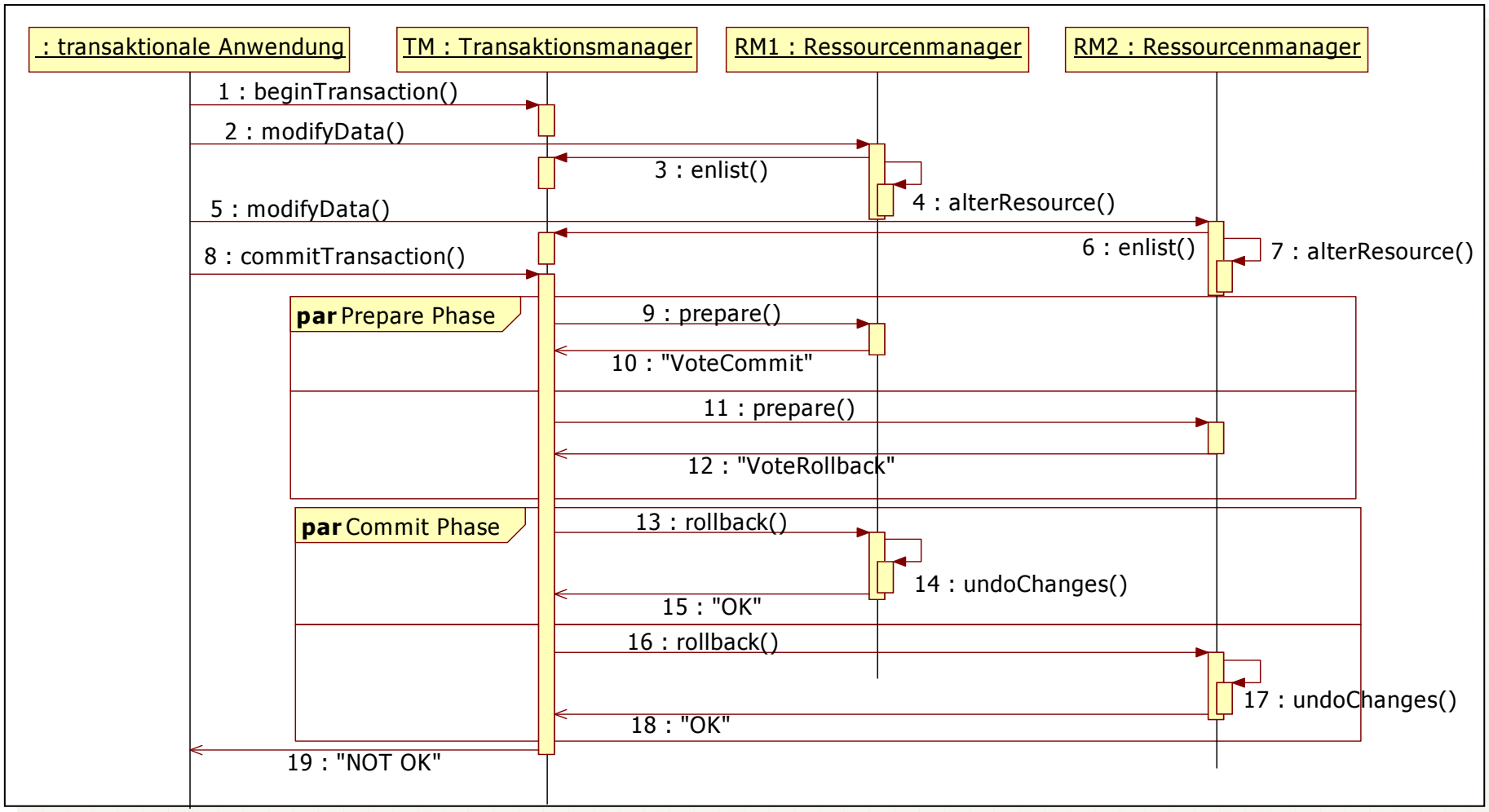
Transaktionen – Two-Phase-Commit Teilnehmer (2/2)



Transaktionen – Erfolgreiches Two-Phase-Commit



Transaktionen – Rollback mittels 2PC



Transaktionen – Fehlgeschlagenes 2PC (1/3)

- Ausfall eines Ressourcenmanagers
  - In „prepare“ Phase
    - Transaktionsmanager rollt die verteilte Transaktion zurück
  - In „commit“ Phase
    - Transaktionsmanager beendet Transaktion (Commit oder Rollback) mit übrigen Teilnehmern
    - Ausgefallener Ressourcenmanager muss den Ausgang der Transaktion bei Transaktionsmanager erfragen und Einigung „nachziehen“

Transaktionen – Fehlgeschlagenes 2PC (2/3)

- Ausfall des Transaktionsmanagers
  - Vor dem Schreiben der Commit Nachricht in stabilen Speicher
    - Transaktionsmanager rollt die verteilte Transaktion zurück
  - Nach dem Schreiben der Commit oder Rollback Nachricht in stabilen Speicher
    - Erneutes Versenden der globale Entscheidung nach Restart des Transaktionsmanagers an alle Teilnehmer
  - Bei einem Ausfall des Transaktionsmanagers müssen die Ressourcen Manager auf dessen Restart warten, um die lokalen Tranaktionen beenden zu können (klassisches 2PC Protokoll = **blockierendes Protokoll**)

Transaktionen – Fehlgeschlagenes 2PC (3/3)

- Heuristische Entscheidungen (bei fehlerhaften Koordinator)
  - Verhindert das Blockieren der Teilnehmer, die nun „eigenmächtig“ ihre lokalen Transaktionen beenden
  - Die Teilnehmer müssen ihre heuristischen Entscheidungen aufbewahren und dem Transaktionsmanager mitteilen
  - Der Transaktionsmanager teilt die heuristische Entscheidung dem Initiator der Transaktion mit (oft als Exception)
    1. Heuristischer Rollback
    2. Heuristisches Commit
    3. Heuristisches Mix
    4. Heuristisches „Hazard“

} Manueller Eingriff notwendig

Transaktionen – Agenda

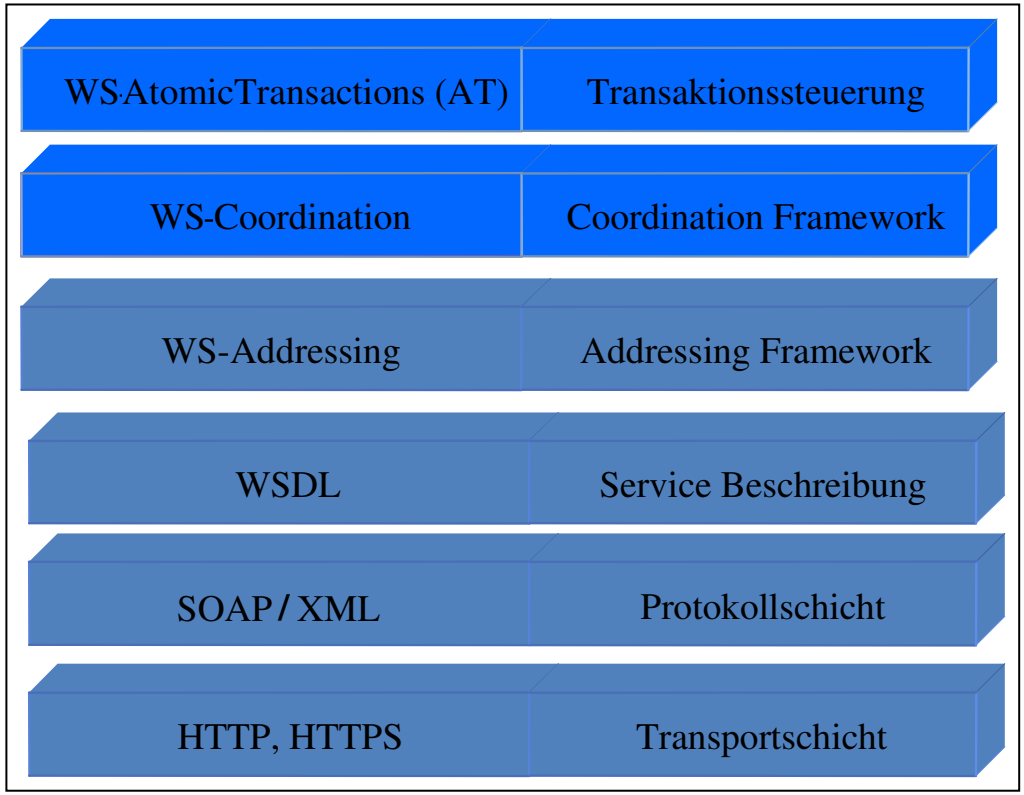
- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – WS AtomicTransaction

WS-AT (1/2)

- Abbildung des 2PC Protokolls in die Web Service Welt
- Wurde von Arjuna, BEA, IBM, IONA, Microsoft und weiteren Firmen definiert
- OASIS

  www.oasis-open.org
- Aktuelle Spezifikation 1.1



Transaktionen – WS AtomicTransaction

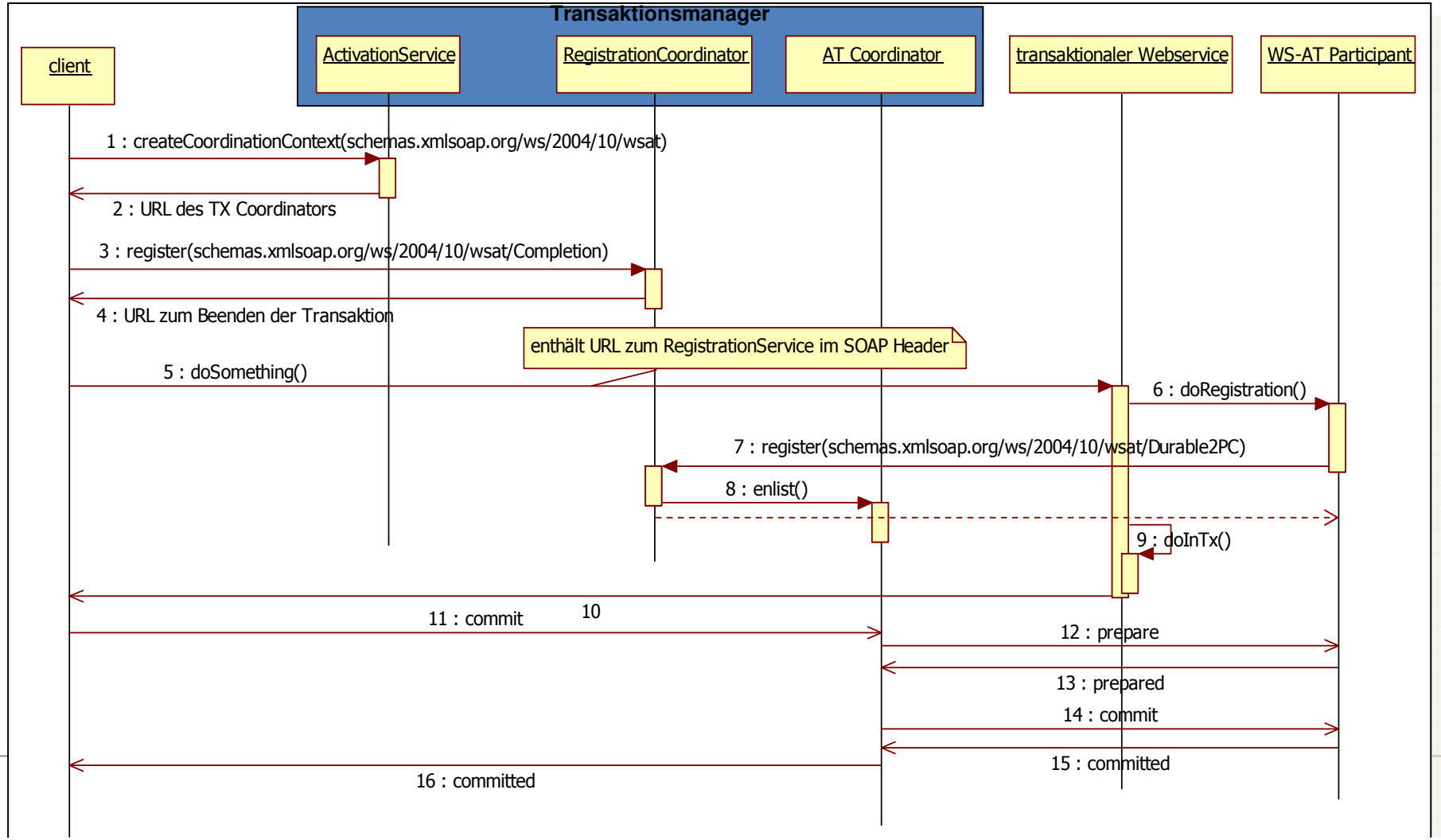
WS-AT (2/2)

- Baut auf der WS Coordination Spezifikation (WS-Coor) auf
  - Definiert Coordination contexts
  - Stellt Kontexterzeugung bereit
  - Ermöglicht die Protokoll Registierung

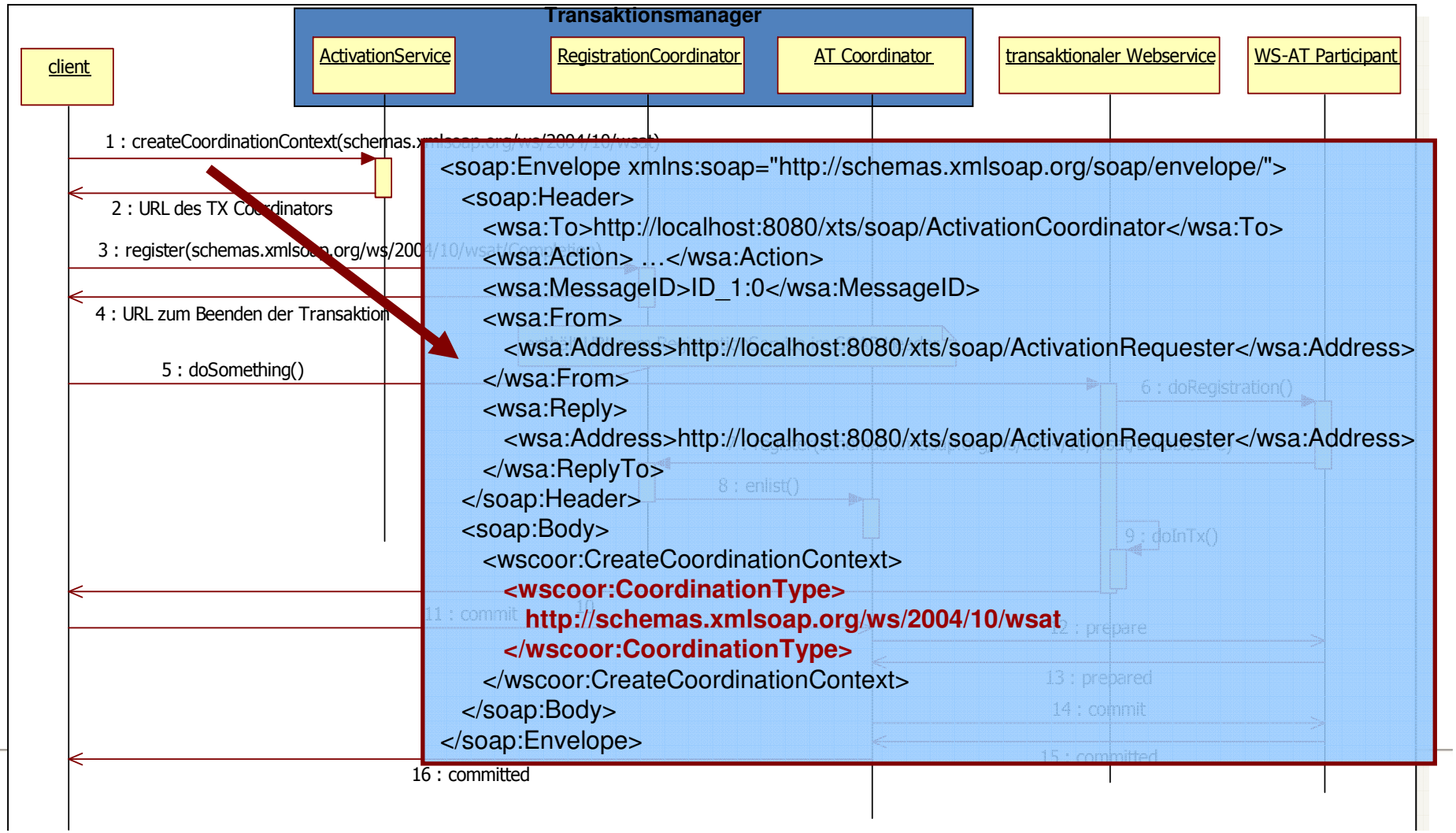
- WS-AT Coordination Type
  - **http://schemas.xmlsoap.org/ws/2003/09/wsat**

- WS-AT Coordination Protocols
  - http://schemas.xmlsoap.org/ws/2003/09/wsat/**Completion**
  - http://schemas.xmlsoap.org/ws/2003/09/wsat/**Volatile2PC**
  - http://schemas.xmlsoap.org/ws/2003/09/wsat/**Durable2PC**

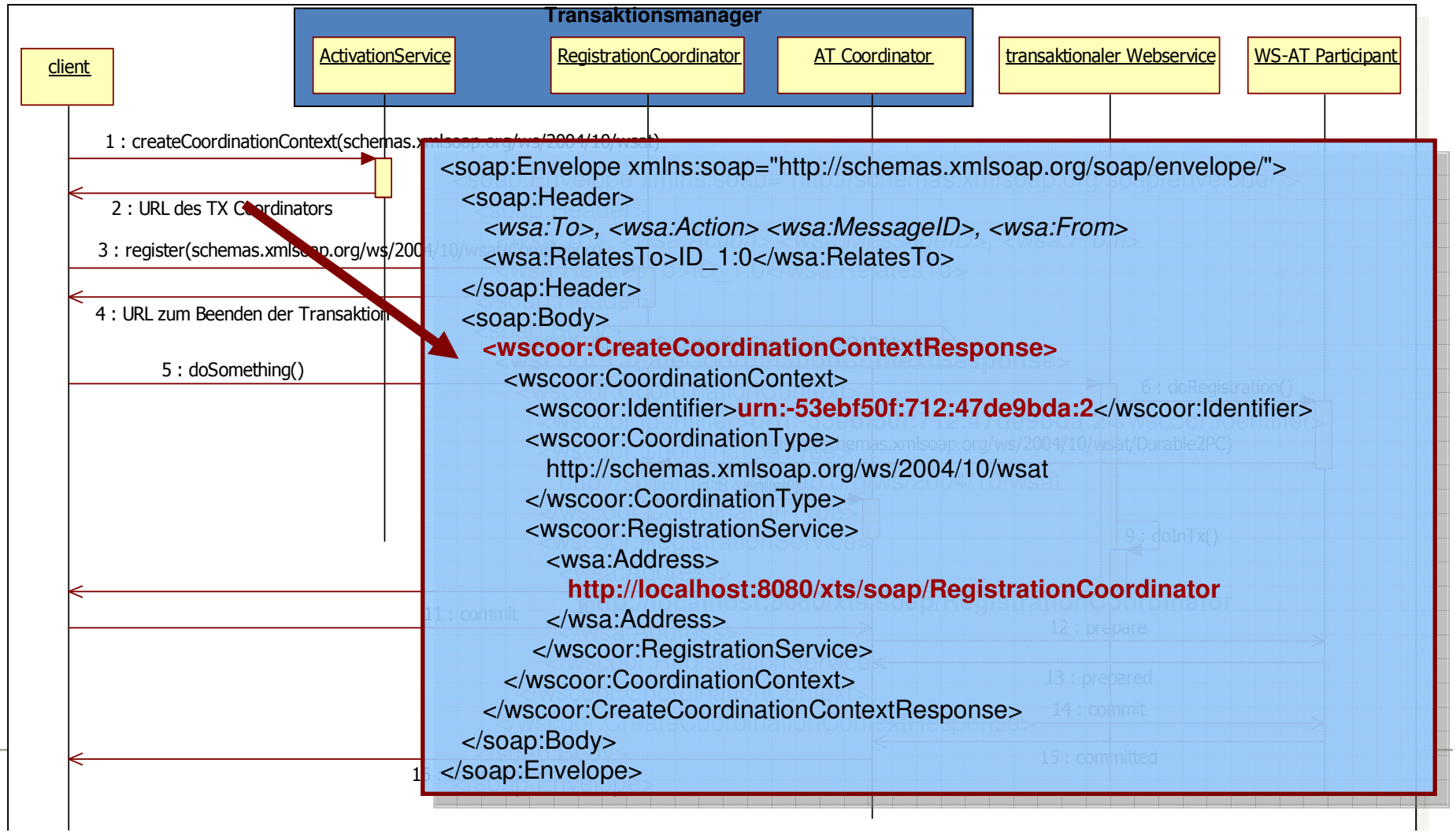
Transaktionen – WS-AT Ablauf (1/14)



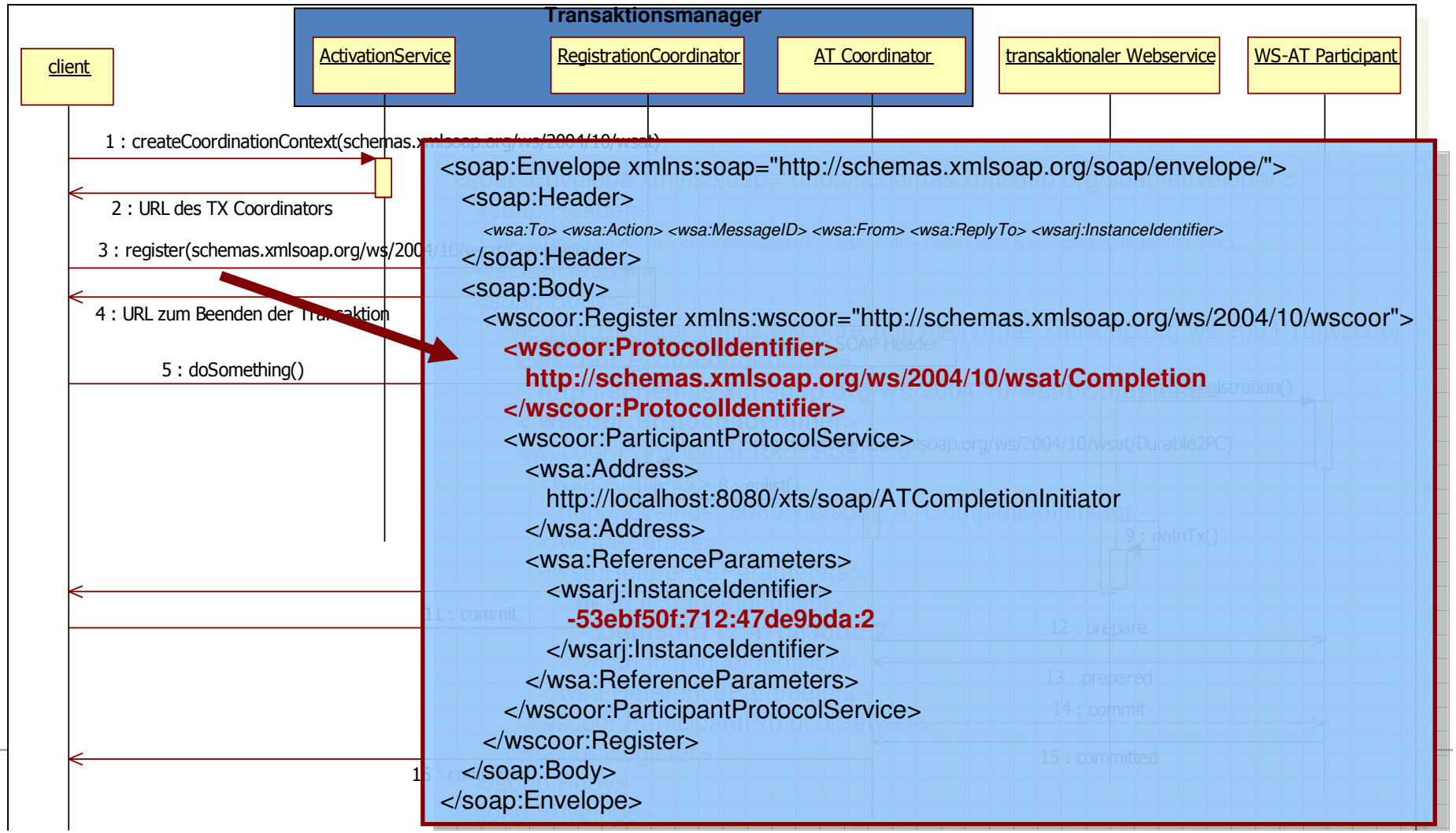
Transaktionen – WS-AT Ablauf (2/14)



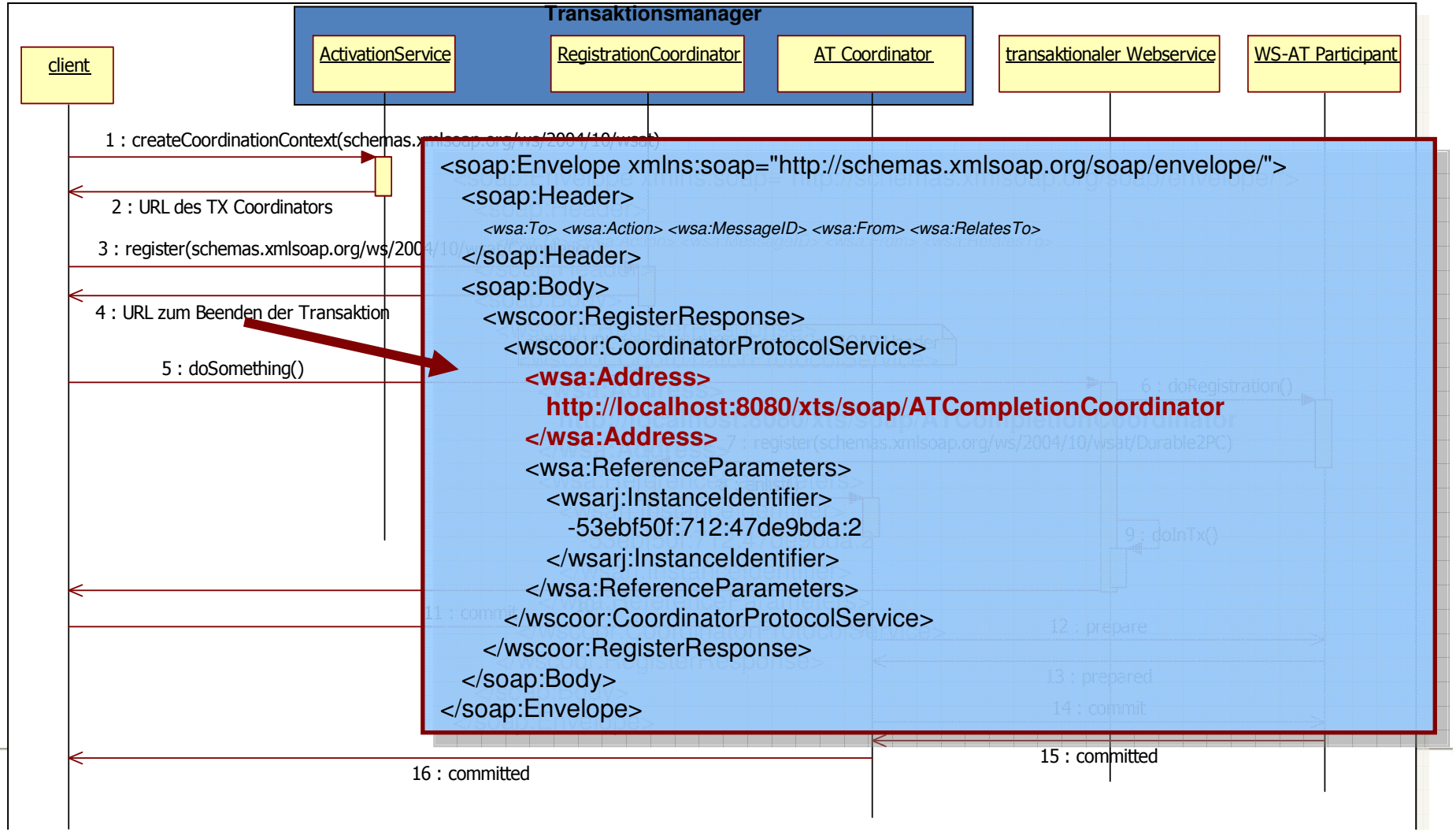
Transaktionen – WS-AT Ablauf (3/14)



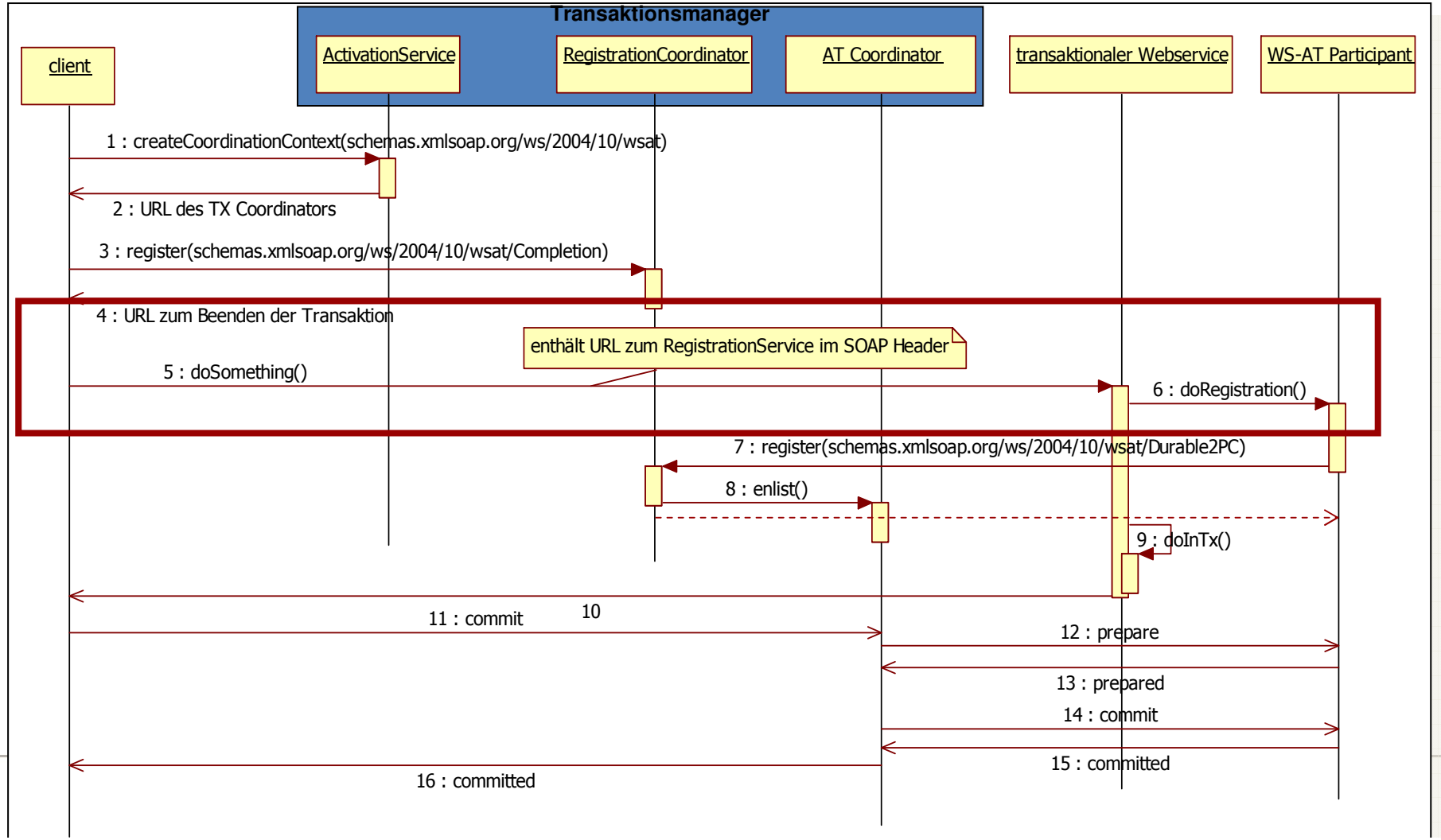
Transaktionen – WS-AT Ablauf (4/14)



Transaktionen – WS-AT Ablauf (5/14)

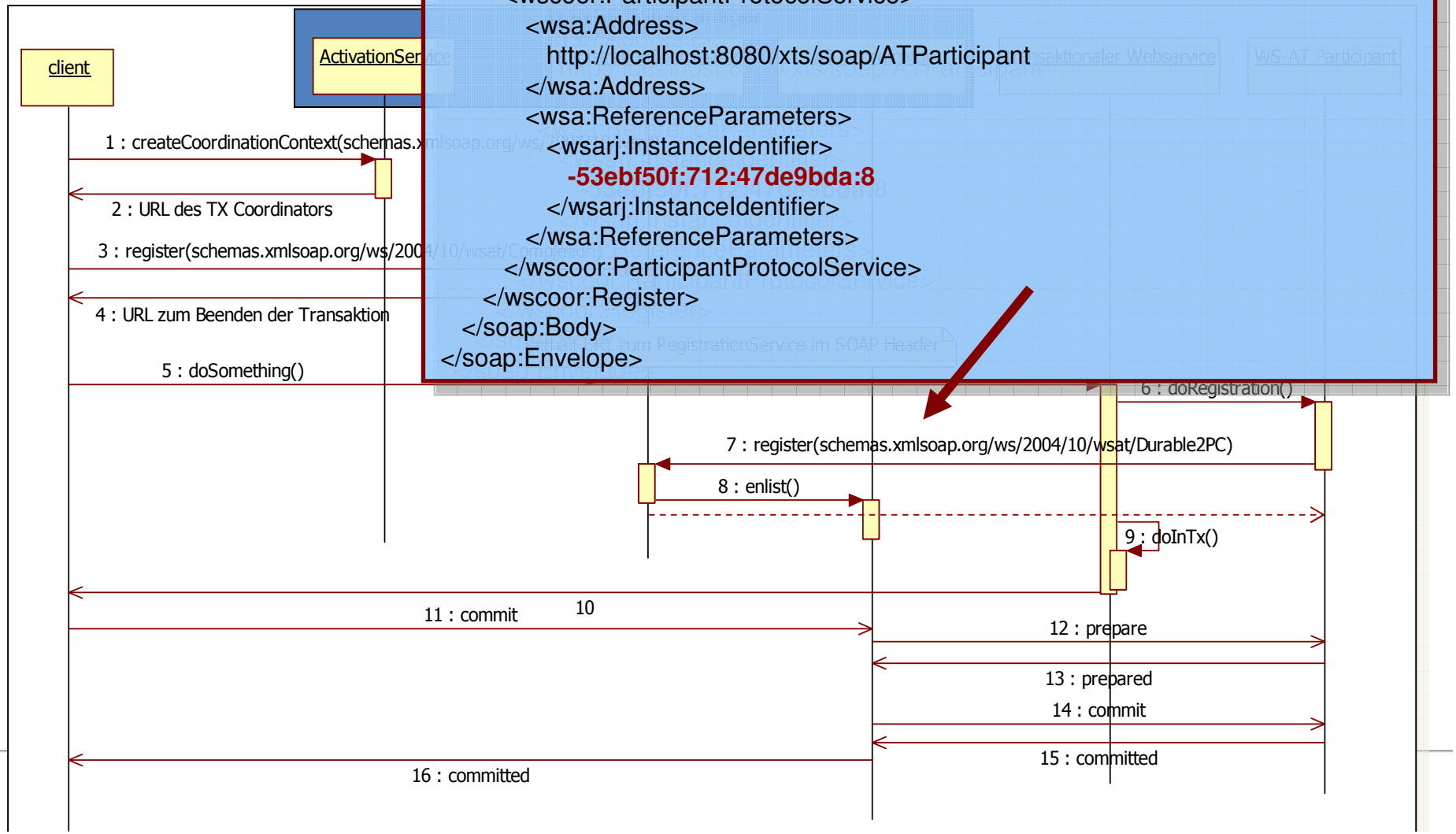


Transaktionen – WS-AT Ablauf (6/14)



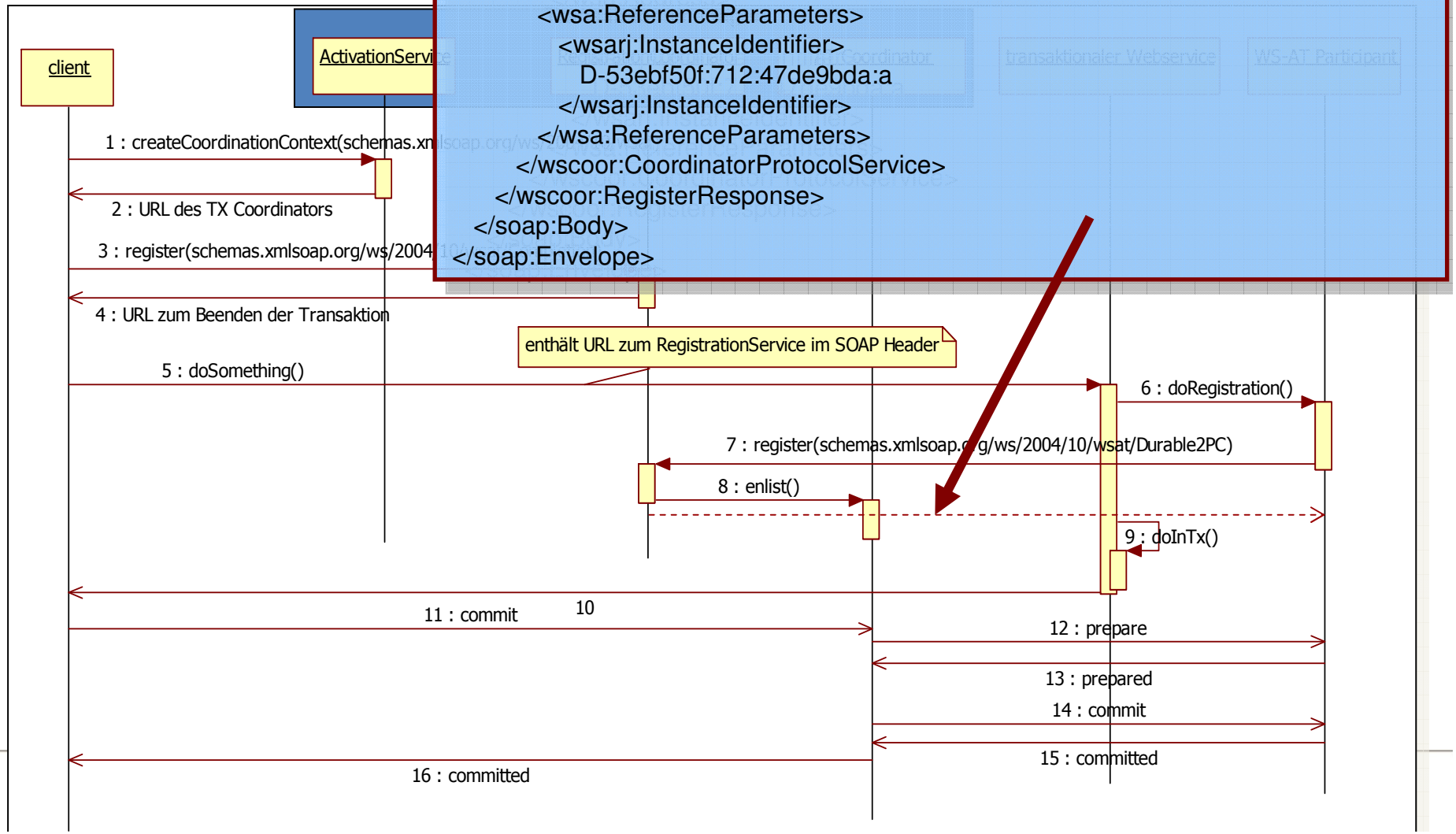
Transaktion

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsa:To> <wsa:Action> <wsa:MessageID> <wsa:From> <wsa:ReplyTo> <wsarj:InstanceIdentifier>
  </soap:Header>
  <soap:Body>
    <wscoor:Register>
      <wscoor:ProtocolIdentifier>
        http://schemas.xmlsoap.org/ws/2004/10/wsat/Durable2PC
      </wscoor:ProtocolIdentifier>
      <wscoor:ParticipantProtocolService>
        <wsa:Address>
          http://localhost:8080/xts/soap/ATParticipant
        </wsa:Address>
        <wsa:ReferenceParameters>
          <wsarj:InstanceIdentifier>
            -53ebf50f:712:47de9bda:8
          </wsarj:InstanceIdentifier>
        </wsa:ReferenceParameters>
      </wscoor:ParticipantProtocolService>
    </wscoor:Register>
  </soap:Body>
</soap:Envelope>
```

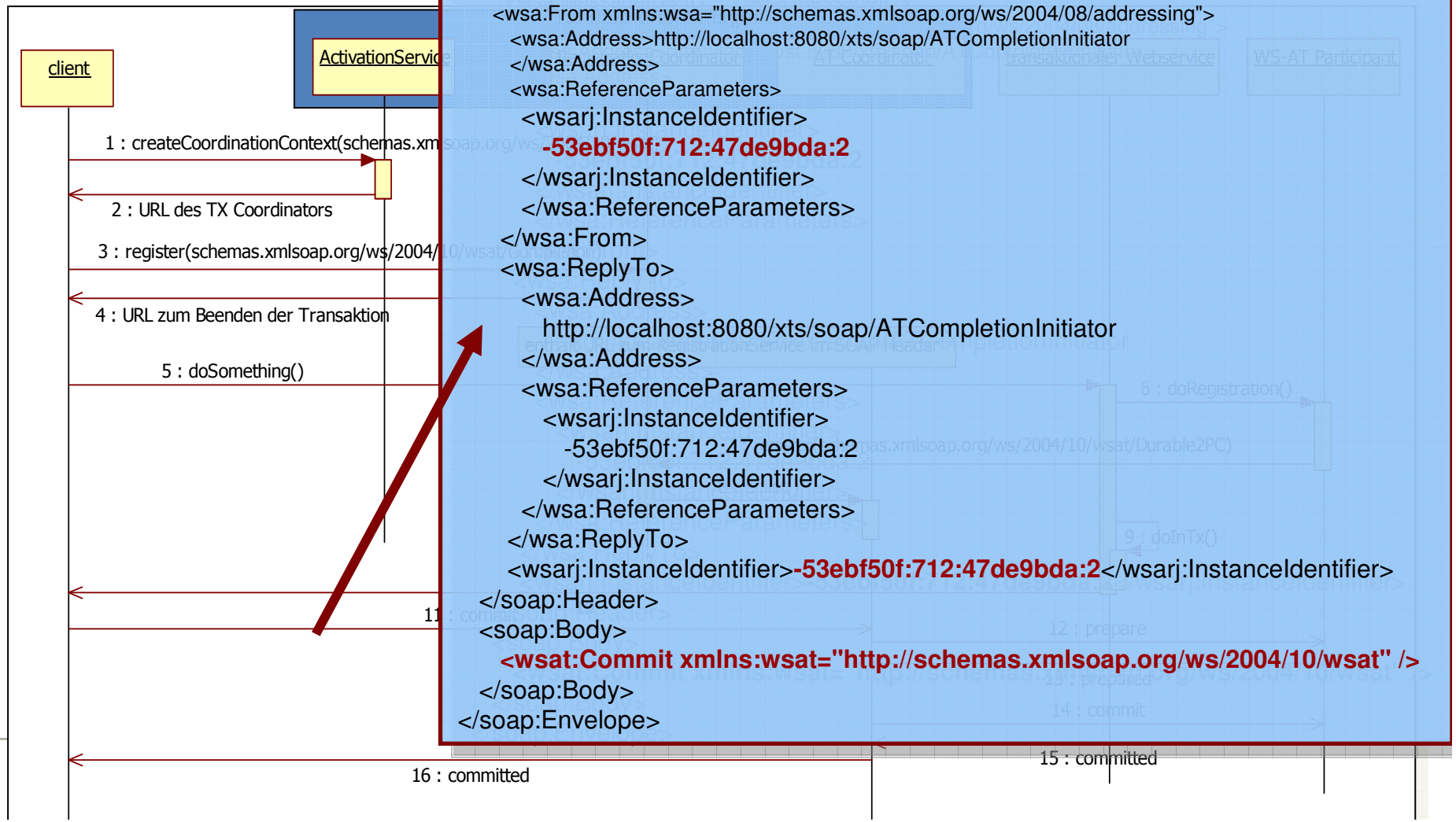


Transaktionen

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <wscoor:RegisterResponse >
      <wscoor:CoordinatorProtocolService>
        <wsa:Address>
          http://localhost:8080/xts/soap/ATCoordinator
        </wsa:Address>
        <wsa:ReferenceParameters>
          <wsarj:InstanceIdentifier>
            D-53ebf50f:712:47de9bda:a
          </wsarj:InstanceIdentifier>
        </wsa:ReferenceParameters>
      </wscoor:CoordinatorProtocolService>
    </wscoor:RegisterResponse>
  </soap:Body>
</soap:Envelope>
```

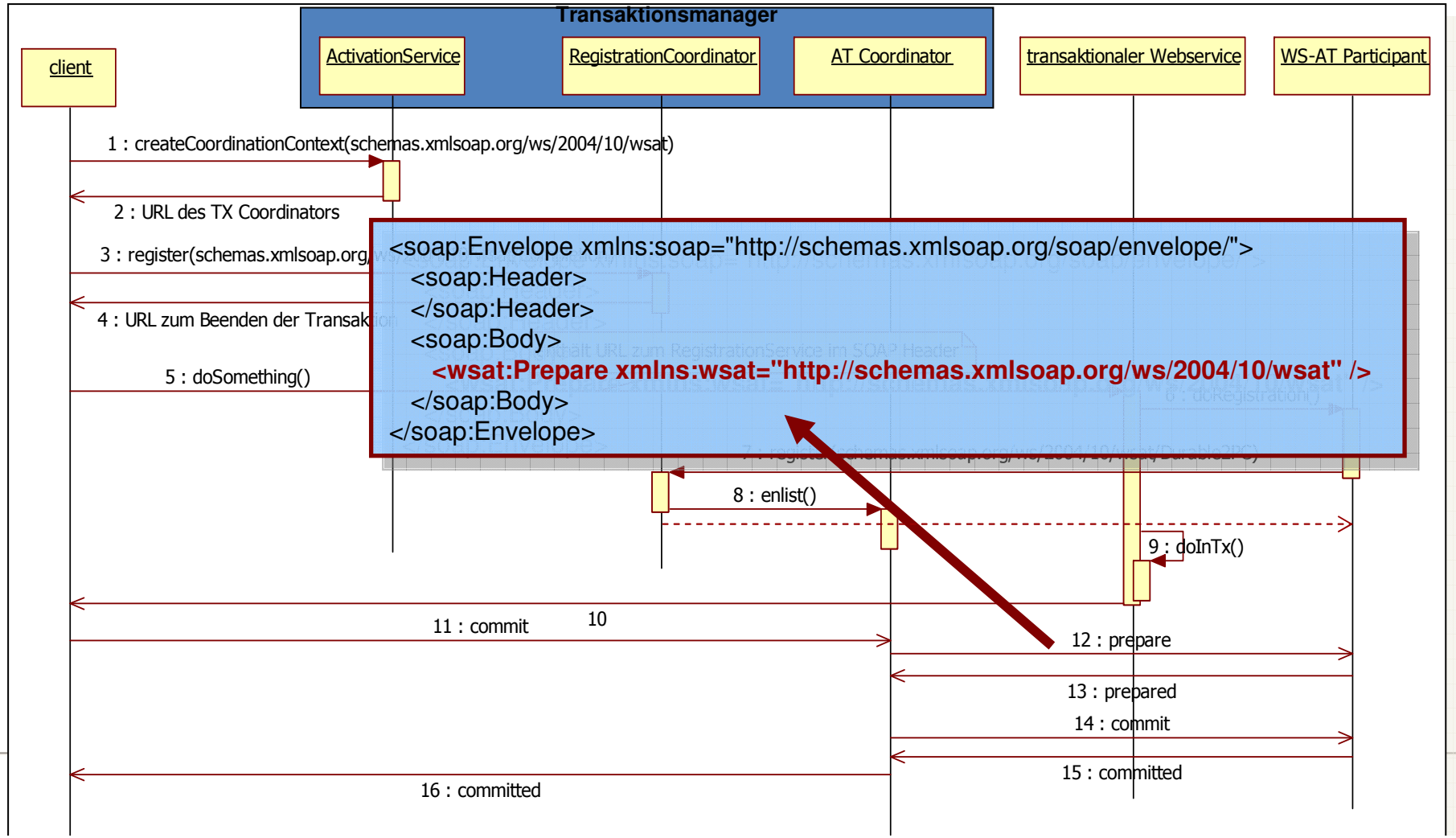


Transaktionen

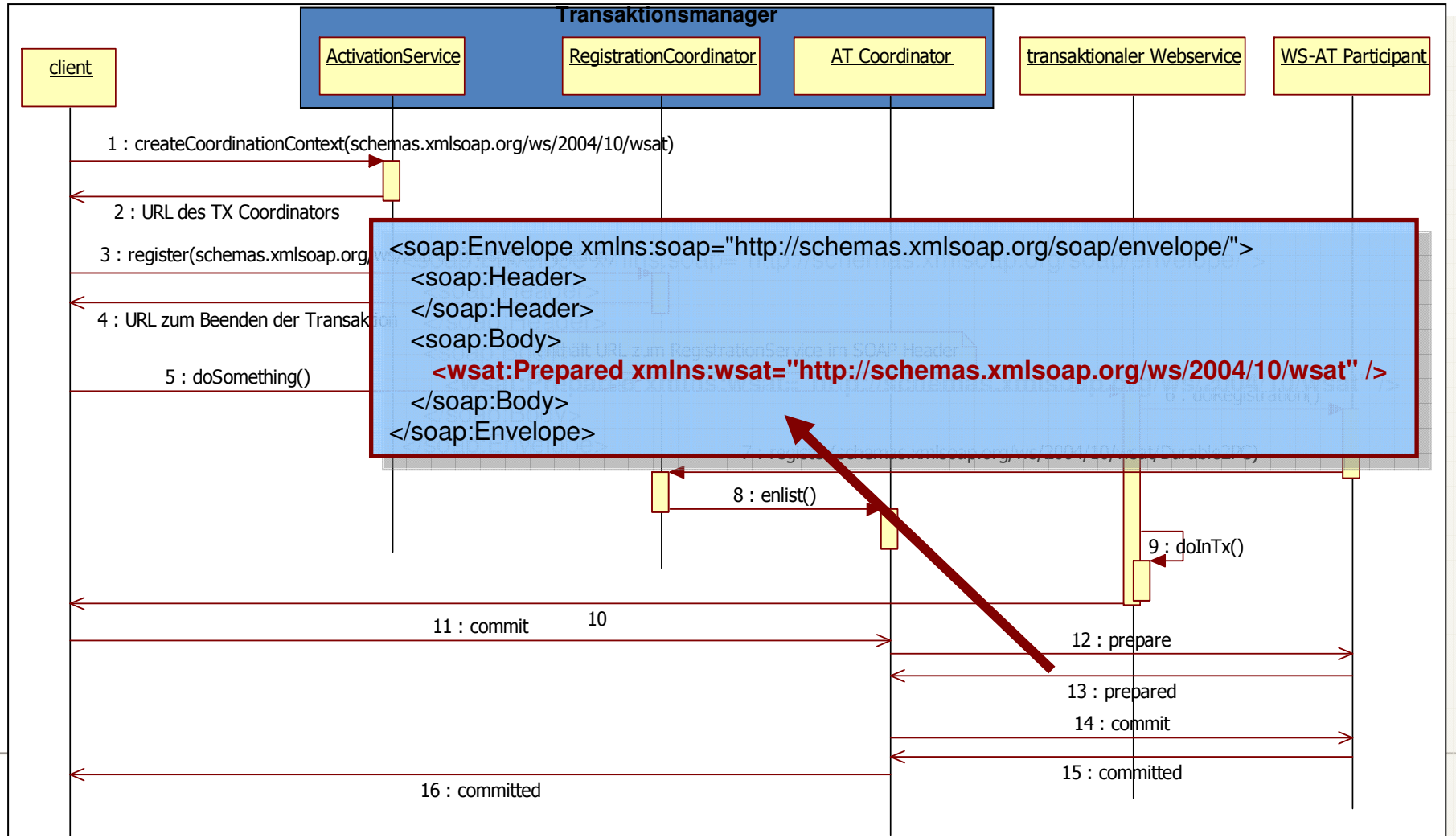



```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsa:To>
      http://localhost:8080/xts/soap/ATCompletionCoordinator
    </wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/10/wsat/Commit
    </wsa:Action>
    <wsa:MessageID> ID… </wsa:MessageID>
    <wsa:From xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
      <wsa:Address>http://localhost:8080/xts/soap/ATCompletionInitiator
      </wsa:Address>
      <wsa:ReferenceParameters>
        <wsarj:InstanceIdentifier>
          -53ebf50f:712:47de9bda:2
        </wsarj:InstanceIdentifier>
      </wsa:ReferenceParameters>
    </wsa:From>
    <wsa:ReplyTo>
      <wsa:Address>
        http://localhost:8080/xts/soap/ATCompletionInitiator
      </wsa:Address>
      <wsa:ReferenceParameters>
        <wsarj:InstanceIdentifier>
          -53ebf50f:712:47de9bda:2
        </wsarj:InstanceIdentifier>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsarj:InstanceIdentifier>-53ebf50f:712:47de9bda:2</wsarj:InstanceIdentifier>
  </soap:Header>
  <soap:Body>
    <wsat:Commit xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat" />
  </soap:Body>
</soap:Envelope>
```

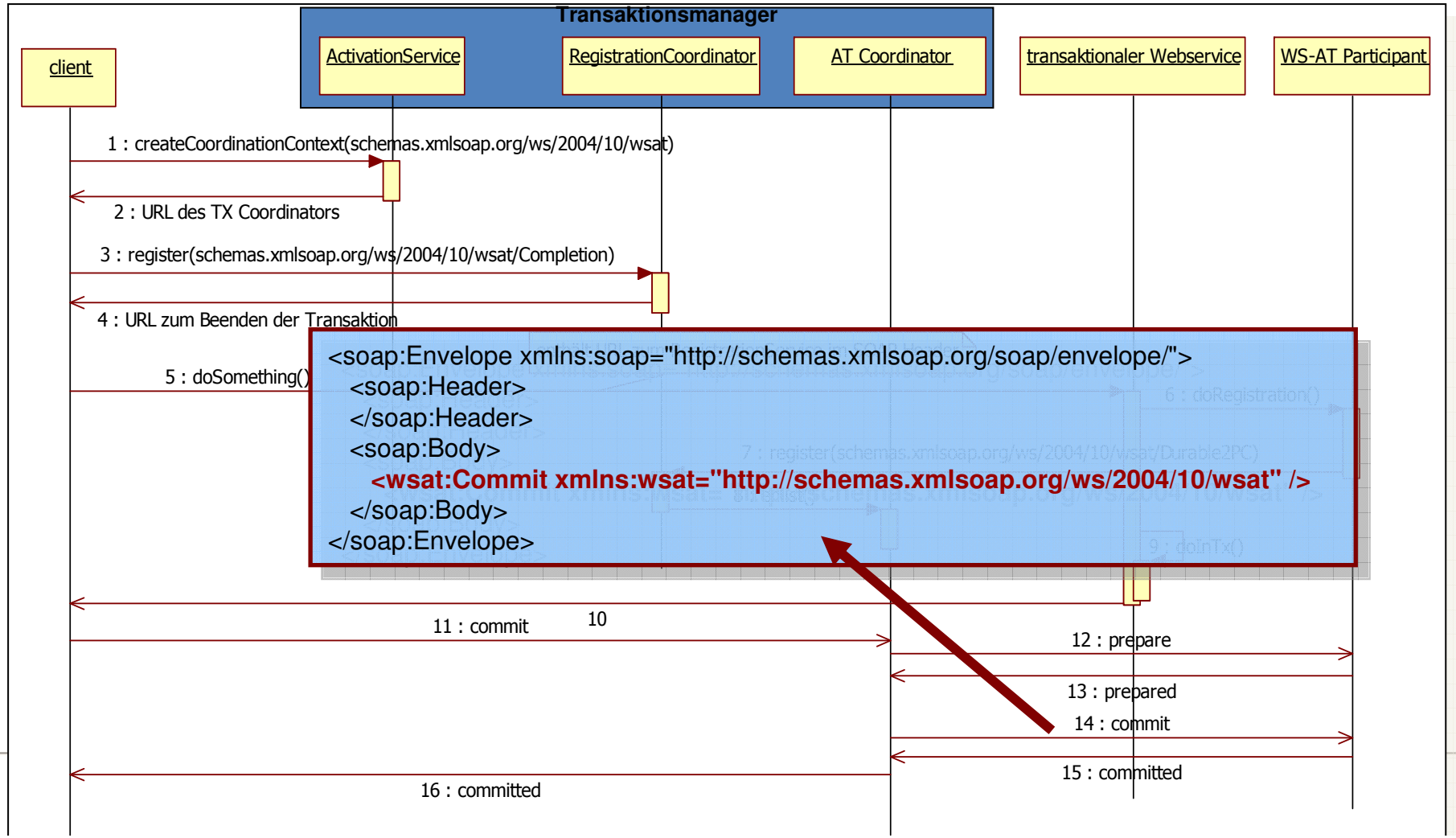
Transaktionen – WS-AT Ablauf (10/14)



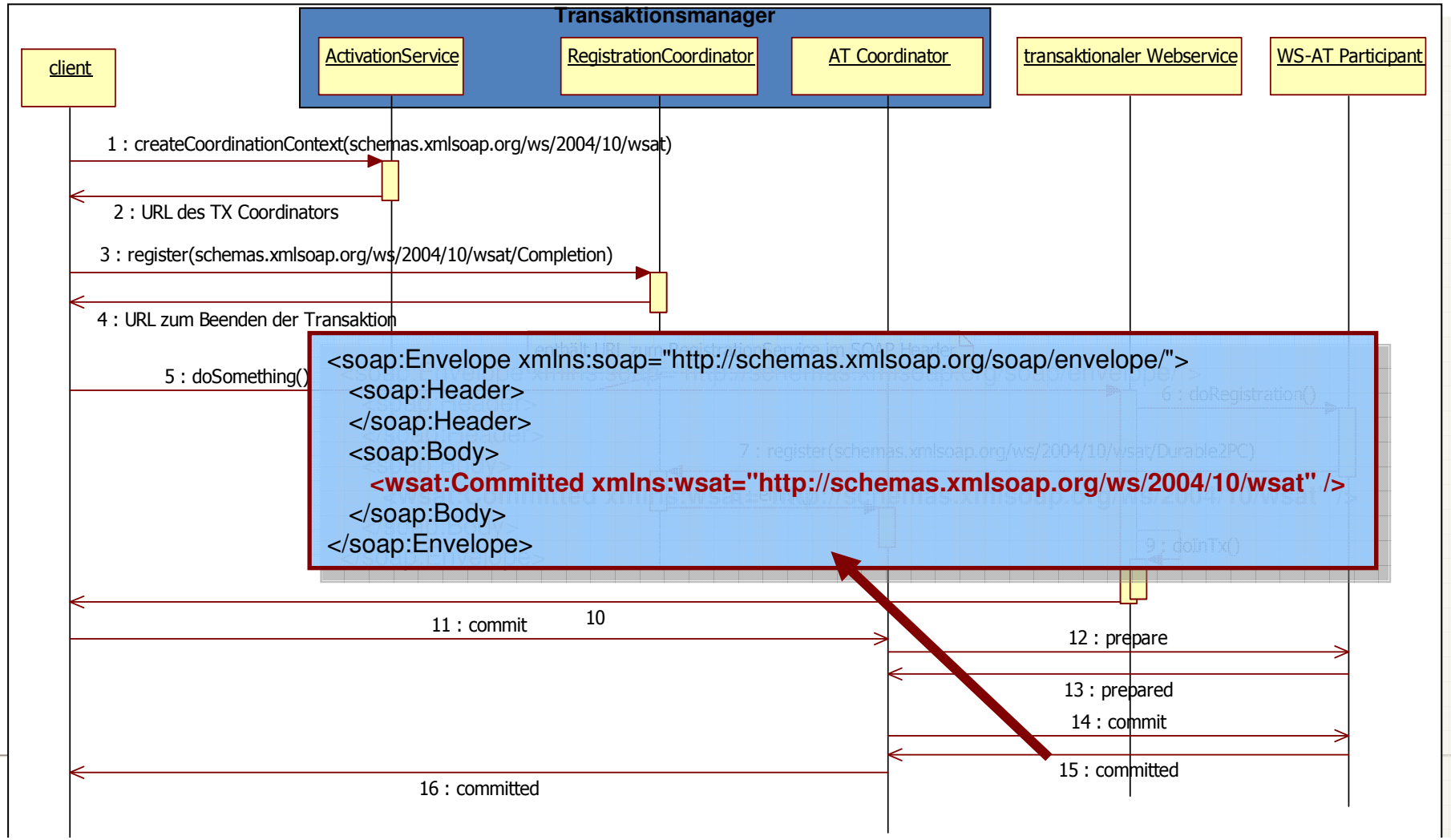
Transaktionen – WS-AT Ablauf (11/14)



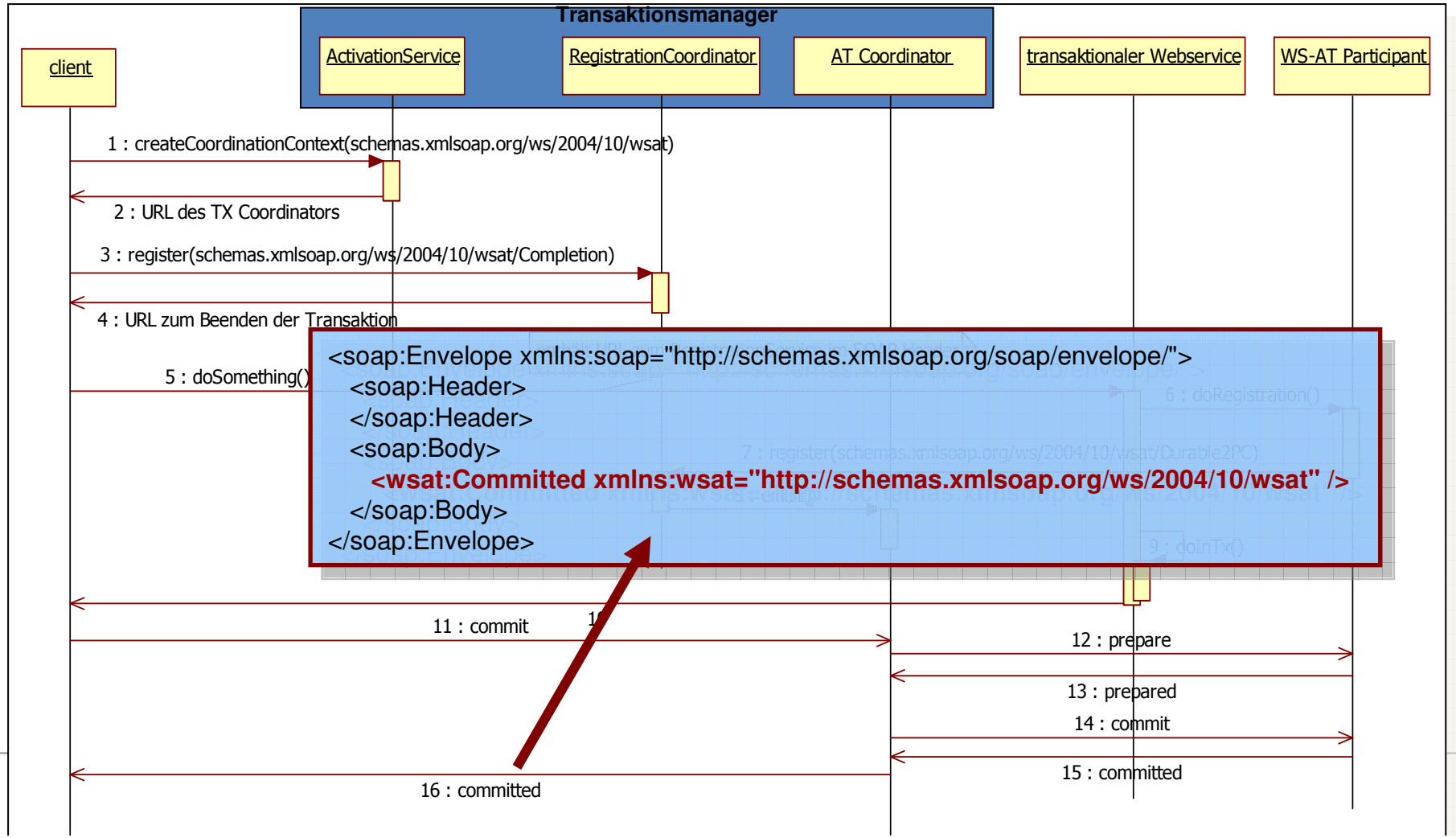
Transaktionen – WS-AT Ablauf (12/14)



Transaktionen – WS-AT Ablauf (13/14)



Transaktionen – WS-AT Ablauf (14/14)



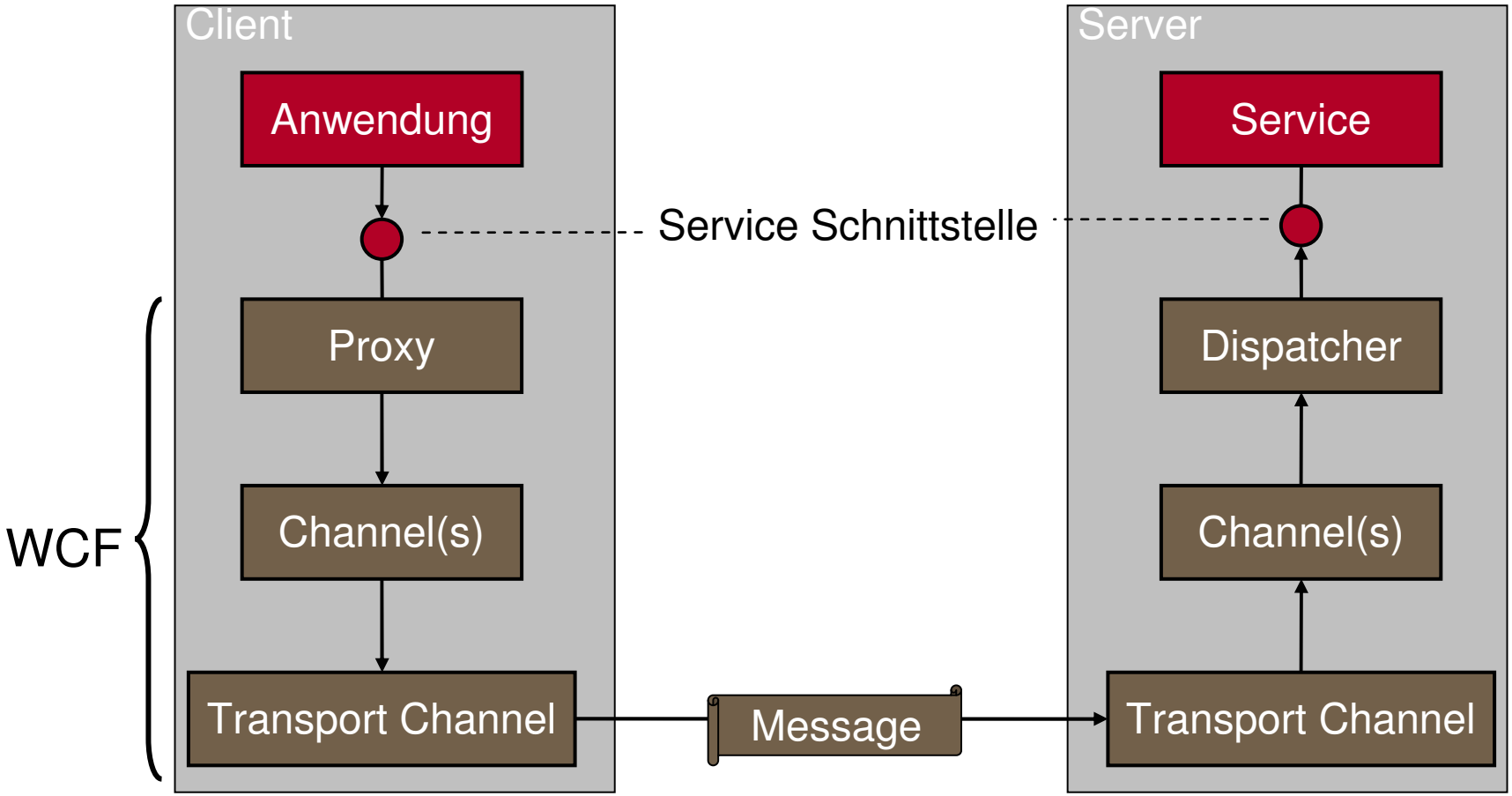
Transaktionen – Agenda

- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – Windows Communication Foundation

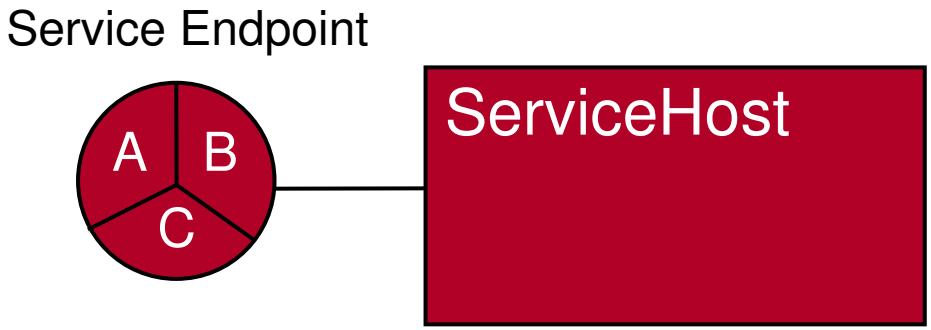
- Integraler Bestandteil seid .NET 3.0 (Release November 2006)
- Nachrichtenstruktur entspricht SOAP Struktur
- Implementiert WS-* Standards (wie **WS-AT**, WS-Security, WS-Reliable Messaging, WS-Addressing)
- Einheitliches Programmiermodell, vereinigt MS Technologien
  - .NET Remoting, ASP.NET Web Services Enterprise Services (COM+) und MSMQ
- Unterstützt Transaktionen über
  - OleTransactions (in Microsoft Umgebungen)
  - **WS-AT im heterogenen Umfeld**
  - Nutzt den Microsoft Distributed Transaction Coordinator (MSDTC)

WCF - Architektur Überblick (I)



WCF – Endpoints

- Services werden über sog. Endpoints bereitsgestellt
- Endpoints bestehen aus drei Bestandteile
  - Address Wo ist der Service zu finden
  - Binding Wie ist der Dienst anzusprechen
  - Contract Welche Operationen bietet der Dienst an



Transaktionen – WCF Endpoint Adressen

- Beschreibt den ‚Ort' eines Service
- Legt das Transportprotokoll fest, WCF unterstützt hierbei
  - **HTTP(S)**
  - TCP
  - IPC
  - MSMQ
  - Peer network
- Beispiel

```
http://localhost:8080/MyService
net.tcp://localhost:8888/Myservice
net.pipe://localhost/NamedPipe
net.msmq://hostName/ServiceName
```

Transaktionen – WCF Endpoint Bindings

- Legt das ‚Kommunikationsmuster' zwischen Client und Server fest
  - Nachrichten Enkodierung, Timeout Verhalten, Zuverlässigkeit , Security und **Transaktionhandhabung**

- WCF unterstützt unterschiedliche Typen
  - NetTcpBinding
  - NetNamedPipeBinding
  - NetMsmqBinding
  - MsmqIntegrationBinding
  - NetPeerTcpBinding
  - BasicHttpBinding
  - Ws(2007)HttpBinding
  - WsDualBinding
  - Ws(2007)FederationHttpBinding

Transaktionen – WCF Endpoint Contracts

- `ServiceContract`, `OperationContract`, `FaultContract`, `DataContract` und `DataMember` beschreiben einen WCF Vertrag

- Beispiel

```
 1:  using System.ServiceModel;
 2:  namespace WCFDemo {
 3:    [ServiceContract (Name= "..", Namespace = "..")]
 4:    public interface ISimpleDemoService {
 5:       [OperationContract (IsOneWay = false)]
 6:       [FaultContract(typeof(ArgumentException))]
 7:       void CallMe(string inputparam);
 8:       [OperationContract]
 9:       void StoreCustomer(Customer customer);
10:     }
11:  }
```

Transaktionen – Agenda

- Motivation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – WCF
Transaktionseinstellung (1/7)

- WCF Address Einstellung für WS-AT
  - HTTP(S)
  - TCP Einsatz von OleTransaction oder WS-AT

- WCF Binding Einstellung für WS-AT
  - `WSHttpBinding` für WS-AT 1.0
  - `WS2007HttpBinding` für WS-AT1.1 (.Net 3.5)
  - `TransactionFlow` Property muss auf `true` gesetzt werden

Transaktionen – WCF

Transaktionseinstellung (2/7)

- WCF Binding Einstellung für WS-AT (Fortsetzung)

Client

```
EndpointAddress address =
    new EndpointAddress("http://localhost:9081/services/RechungsService");
WSHttpBinding binding = new WSHttpBinding(SecurityMode.None);
binding.TransactionFlow = true;
```

Server

```
Uri address = new Uri("http:// localhost:9081/services/RechungsService");
WSHttpBinding binding = new WSHttpBinding(SecurityMode.None);
binding.TransactionFlow = true;
```

Transaktionen – WCF

Transaktionseinstellung (3/7)

- WCF Contract Attribut für WS-AT
  - `TransactionFlowOption` bestimmt, ob eingehende Transaktionen vom Service genutzt werden können
    - **`NotAllowed:`** Wenn Client eine Transaktion gestartet hat, wird diese stillschweigend ignoriert (Default)
    - **`Allowed`**: Der aufgerufene Service kann die Client Transaktion nutzen, muss aber nicht
    - **`Mandatory:`** Der Client muss eine offen Transaktion beim Betreten des Service bereitstellen. Der Service kann diese nutzen, muss aber nicht.

Transaktionen – WCF
Transaktionseinstellung (4/7)

- WCF Contract Attribute für WS-AT (Fortsetzung)
  - Beispiel

```
using System.ServiceModel;

namespace TransactionWCF {
    [ServiceContract]
    public interface IBestellService {
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Mandatory)]
        void BestellungAufnehmen(string kundenID);
    }
}
```

  - Bei generierten Schnittstellen (Clientseite) kann es notwendig sein, das `TransactionFlow` Attribut manuell einzufügen

Transaktionen – WCF

Transaktionseinstellung (5/7)

- WCF Contract Implementierung für WS-AT
  - `TransactionScopeRequired` Attribut an der Servicemethoden Implementierung bestimmt, ob die Client Transaktion genutzt wird
    - `true`: propagierte Transaktion wird als „*ambient transaction*“ genutzt
    - `false`: propagierte Transaktion wird ignoriert (**default**)

  - Mittels `TransactionAutoComplete=true` Attribute kann die Servicemethode bestimmen, ob sie automatisch in der prepare Phase des 2PC abstimmt (**default**)
    - Keine Exception → `vote Commit`
    - Exception → `vote Rollback`

Transaktionen – WCF

Transaktionseinstellung (6/7)

- WCF Contract Implementierung für WS-AT (Fortsetzung)
  - Beispiel

```
[OperationBehavior (TransactionScopeRequired=true,
                    TransactionAutoComplete=true)]
public void BestellungAufnehmen(string kundenID)  {
   Transaction tx = Transaction.Current;
   SqlConnection connection = new SqlConnection("server=HAUG\\sqlexpress;Integrated
                                                 Security=SSPI; database=WCF_DEMO_DB");
   SqlCommand cmd = new SqlCommand();
   connection.Open();
   connection.EnlistTransaction(tx);
   cmd.CommandText = "Insert into Bestellung (KundenID) VALUES '"+kundenID+"')";
   cmd.ExecuteNonQuery();
   connection.Close();
}
```

Transaktionen – WCF

Transaktionseinstellung (7/7)

- WCF Contract Implementierung für WS-AT (Fortsetzung)
  - Explizites „Abstimmen“

```
[OperationBehavior (TransactionScopeRequired=true,
                    TransactionAutoComplete=false)]
public void BestellungAufnehmen(string kundenID)  {
   Transaction tx = Transaction.Current;
   //… Arbeit in Transaktion verrichten
   OperationContext.Current.SetTransactionComplete();
}
```

  - Anschliessend darf keine Methode an der Transaktion oder transaktionalen Ressourcen aufgerufen werden, sonst wirft die WCF eine `InvalidOperationException`
  - Exception Handhabung
    - Soll eine Exception einen Rollback auslösen, so darf sie nicht von der Methode geschluckt werden

Transaktionen – Transaktion im Client starten

- Beispiel

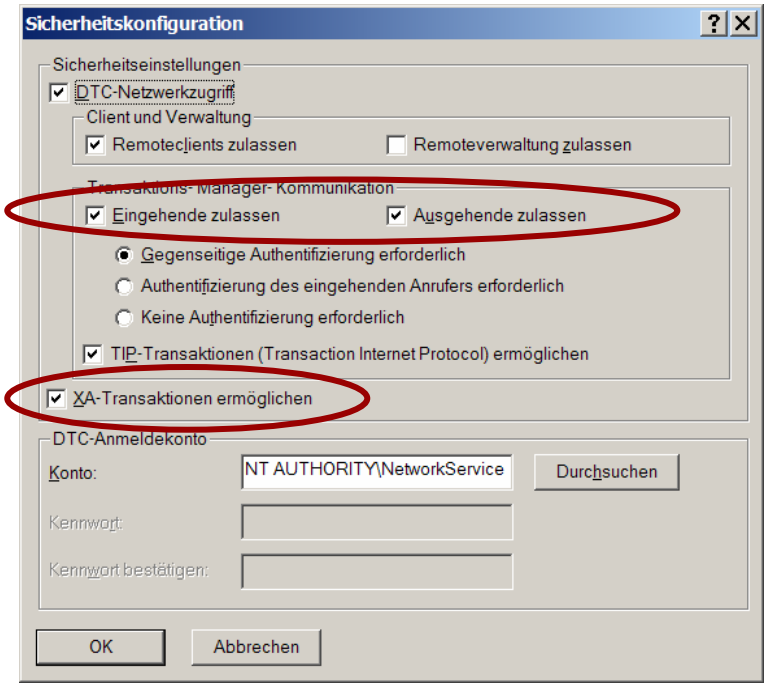
```
EndpointAddress address =
                new EndpointAddress("http://localhost:8181/BestellService");
WSHttpBinding binding = new WSHttpBinding(SecurityMode.None);
binding.TransactionFlow = true;
clientChannel =
                  ChannelFactory<TxBestellService.IBestellServiceChannel>.
                     CreateChannel(binding, address);
TransactionOptions txopt = new TransactionOptions();
txopt.Timeout = new TimeSpan(0,3,30);
using (TransactionScope txScope = new
          TransactionScope(TransactionScopeOption.Required,txopt)) {
   clientChannel.BestellungAufnehmen("Mueller ID");
   txScope.Complete();
}
```

Transaktionen – Schon fertig ?

- WCF WS-AT Beispiel wird in einer Microsoft Welt wunderbar funktionieren, im heterogen Umfeld nicht !
  - WS-AT ist per default deaktiviert, stattdessen wird ein optimiertes, aber proprietäres Protokoll verwendet
- Lösung
  - WCF und Windows OS benötigt Patch KB912817
  - Kommunikation zwischen MSDTC und Teilnehmern muss mittels Zertifikaten abgesichert werden

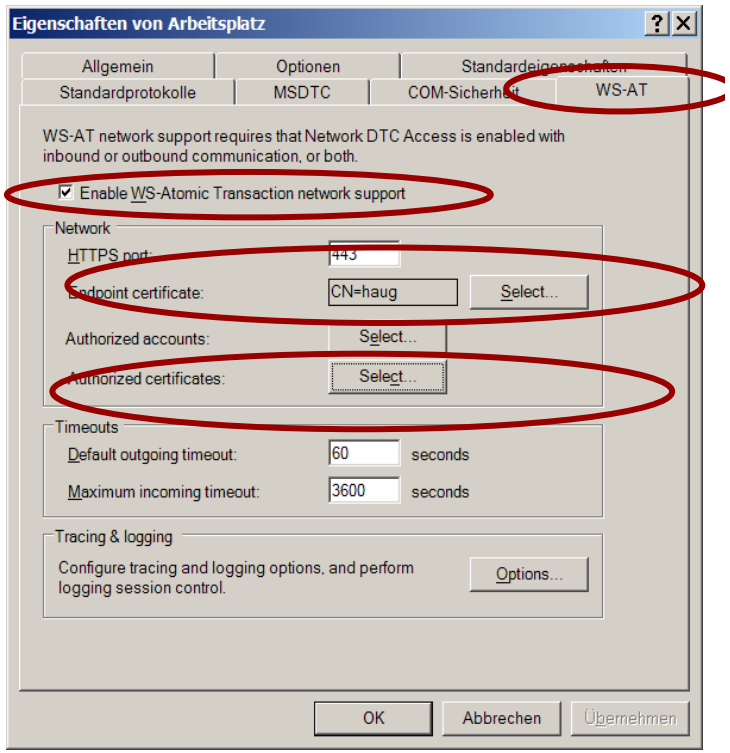
Transaktionen – WS-AT Konfiguration (1/2)

- Vorbereitung für WS-AT
- Registry-Eintrags
  `HKLM\SOFTWARE\Microsoft\WSAT\3.0\ServiceModel-DiagnosticTracing` auf den Wert „`1f`“ setzen



Transaktionen – WS-AT Konfiguration (2/2)

- WS-AT benötigt Zertifikate
- Zertifikat für MSDTC und Teilnehmer(!) notwendig
- Erstellen mit makecert (Windows SDK) und dem Keystore des Rechnern hinzufügen
  - `makecert -ss My -sr LocalMachine -n CN=%COMPUTERNAME% -sky exchange -ir LocalMachine -iv WSAT-CA.pvk -ic WSAT-CA.cer`
- Zertifikate für WS-AT nutzen
  - WsatConfig.exe
  - Snap-in (regasm.exe /codebase WsatUI.dll)

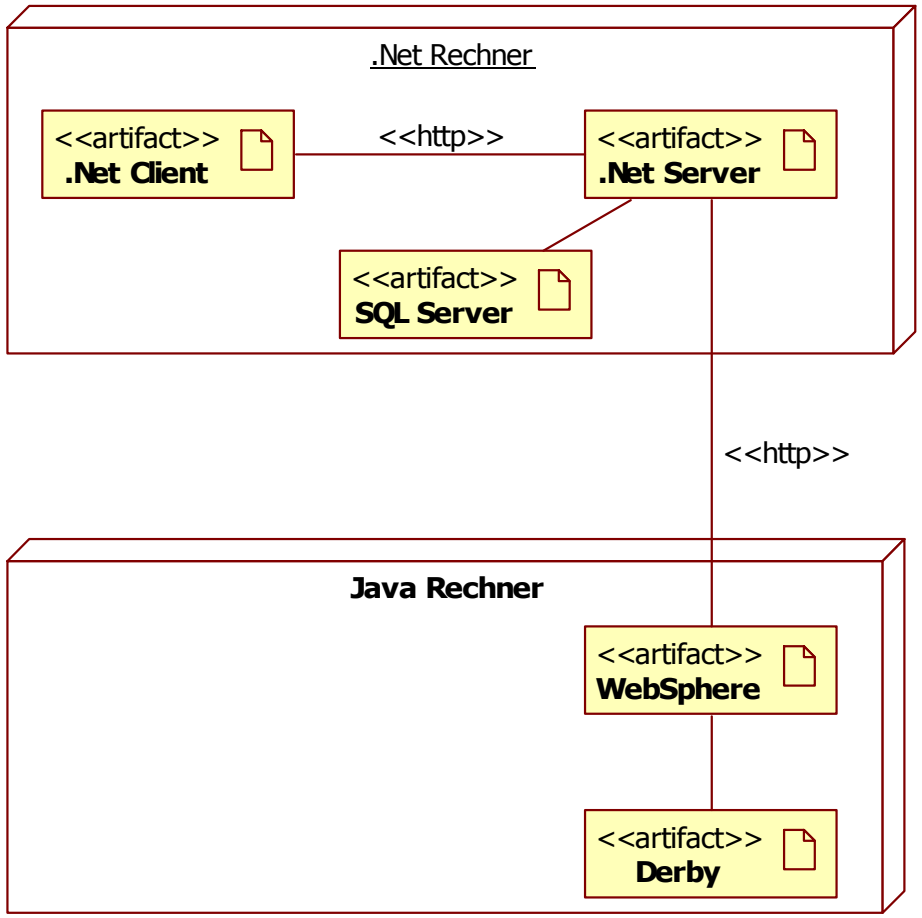


Transaktionen – Agenda

- Motiviation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – WCF mit Java Service (1/9)

- Szenario



Transaktionen – WCF mit Java Service (2/9)

- Das `WSHttpBinding` hat sowohl bei IBM Websphere 6.1 als auch JBoss 4.0.5GA nicht funktioniert
  - Content-Type und das Encoding sind unterschiedlich
- Einsatz eines CustomBindings ist notwendig

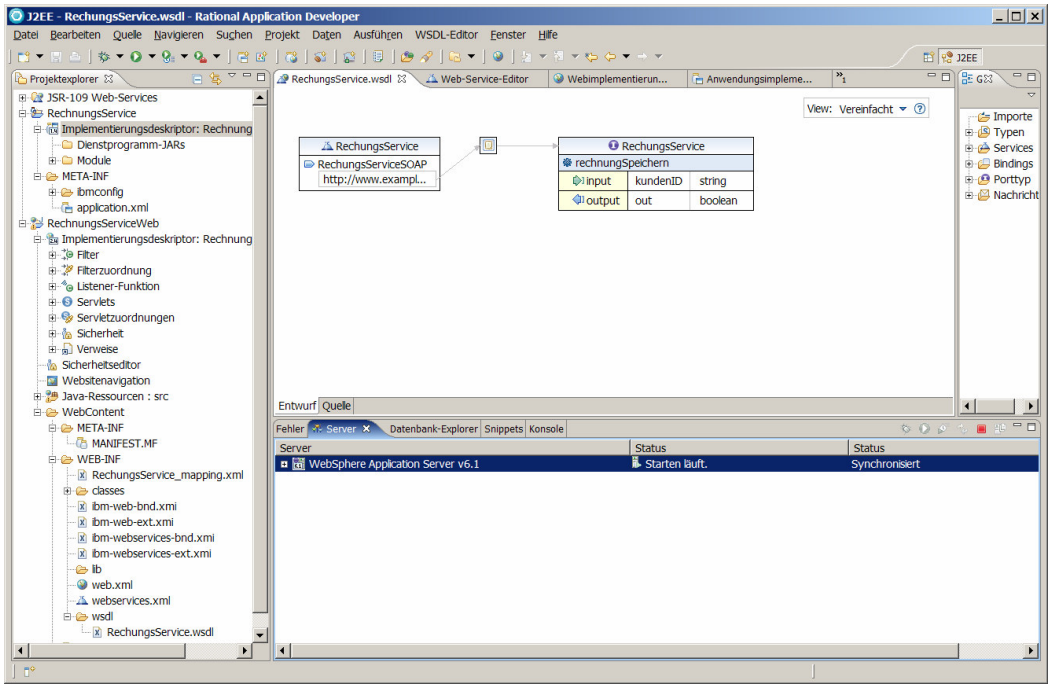
```
HttpTransportBindingElement http = new HttpTransportBindingElement();
CustomBinding bind = new CustomBinding();
TransactionFlowBindingElement txFlowBinding  = new TransactionFlowBindingElement();
txFlow.TransactionProtocol = TransactionProtocol.WSAtomicTransactionOctober2004;

bind.Elements.Add(txFlow);
bind.Elements.Add(new CustomTextMessageBindingElement("UTF-8", "text/xml",
                                                      MessageVersion.Soap11));
bind.Elements.Add(http);
```

- An `TransactionFlow` Attribut an generierter Schnittstelle denken

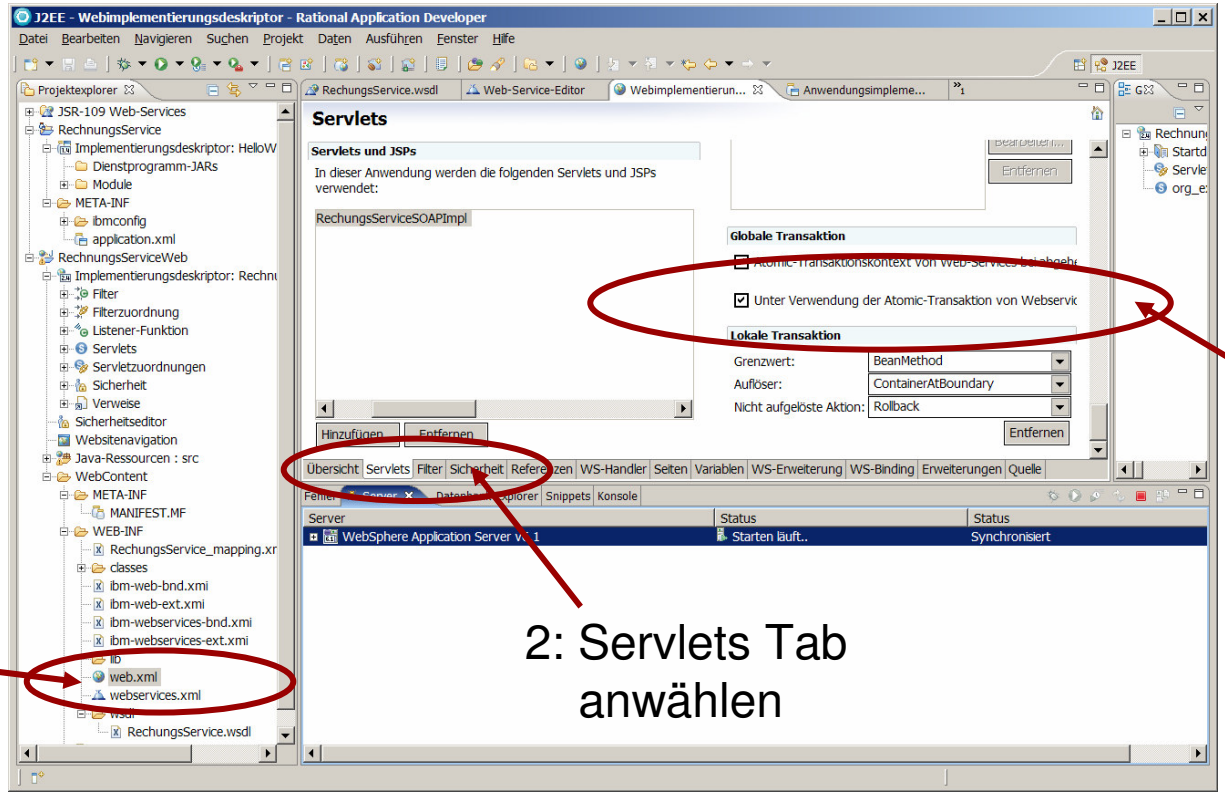
Transaktionen – WCF mit Java Service (3/9)

- Implementierung des Java Diensts mit IBM Rational Application Developer
- Service definieren (WSDL erzeugen)



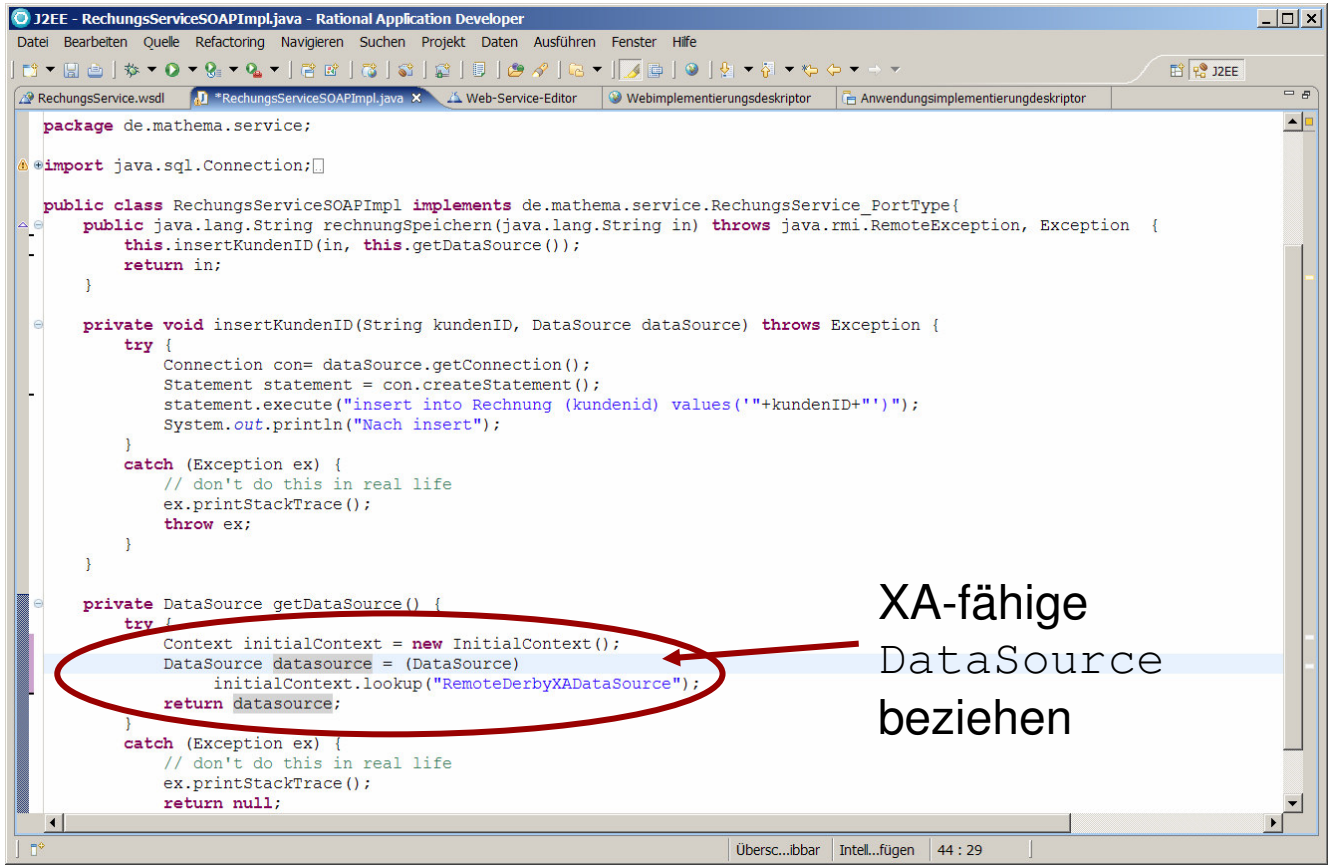
Transaktionen – WCF mit Java Service (4/9)

- WS-AT im Dienst aktivieren



Transaktionen – WCF mit Java Service (5/9)

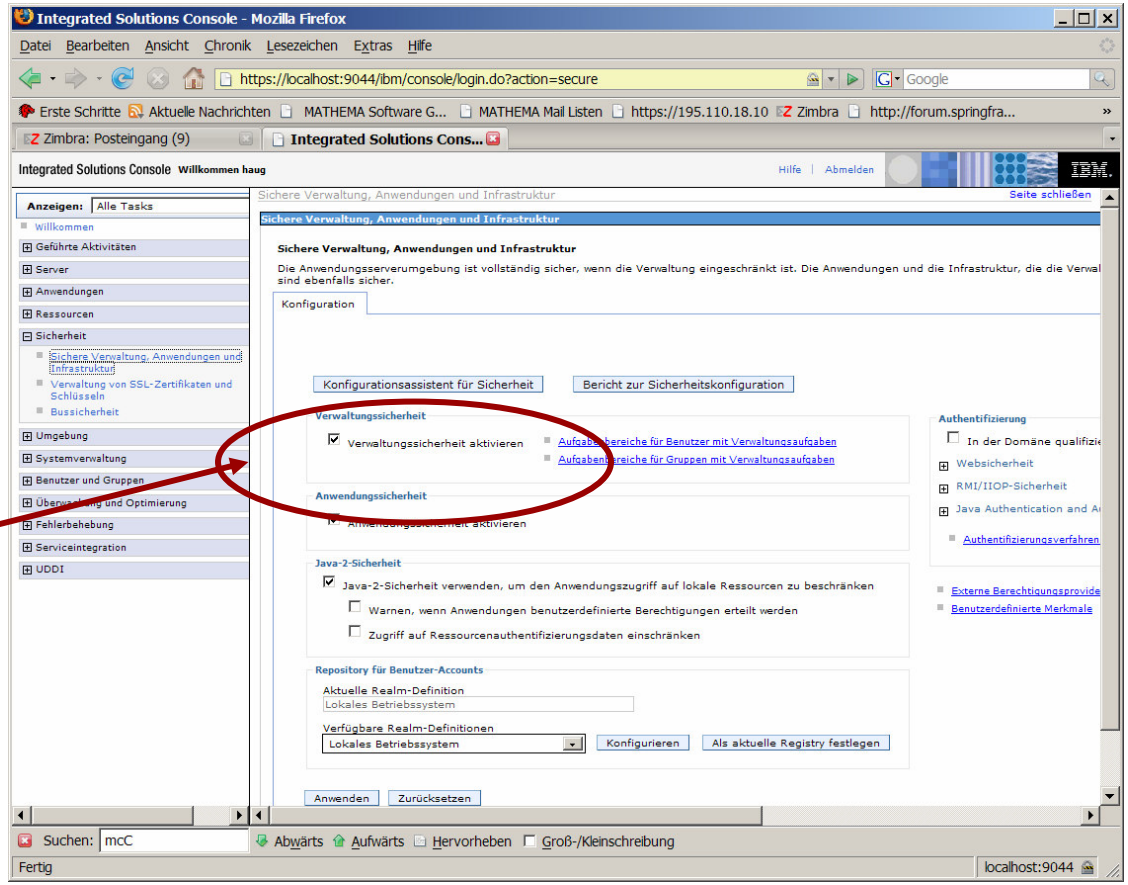
- Service implementieren



Transaktionen – WCF mit Java Service (6/9)

- Service definieren (WSDL erzeugen)

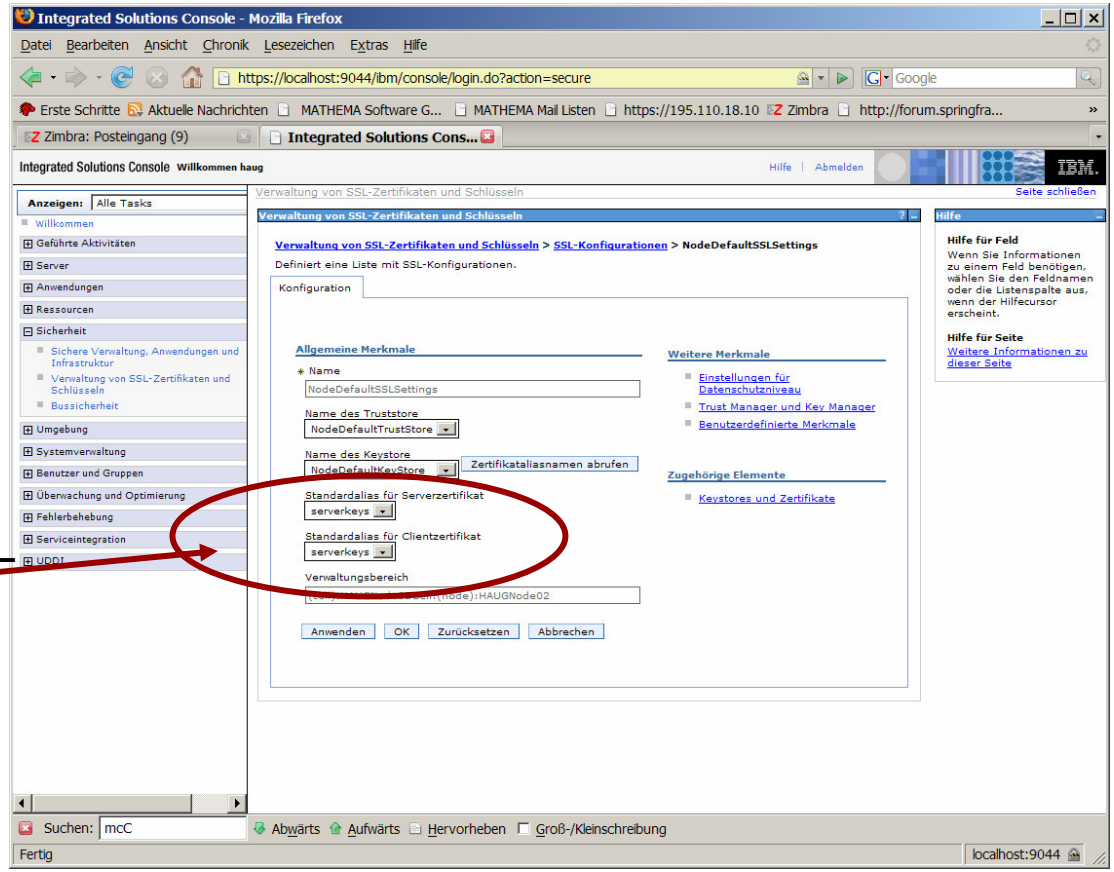
„Sicherheit“ aktivieren



Transaktionen – WCF mit Java Service (7/9)

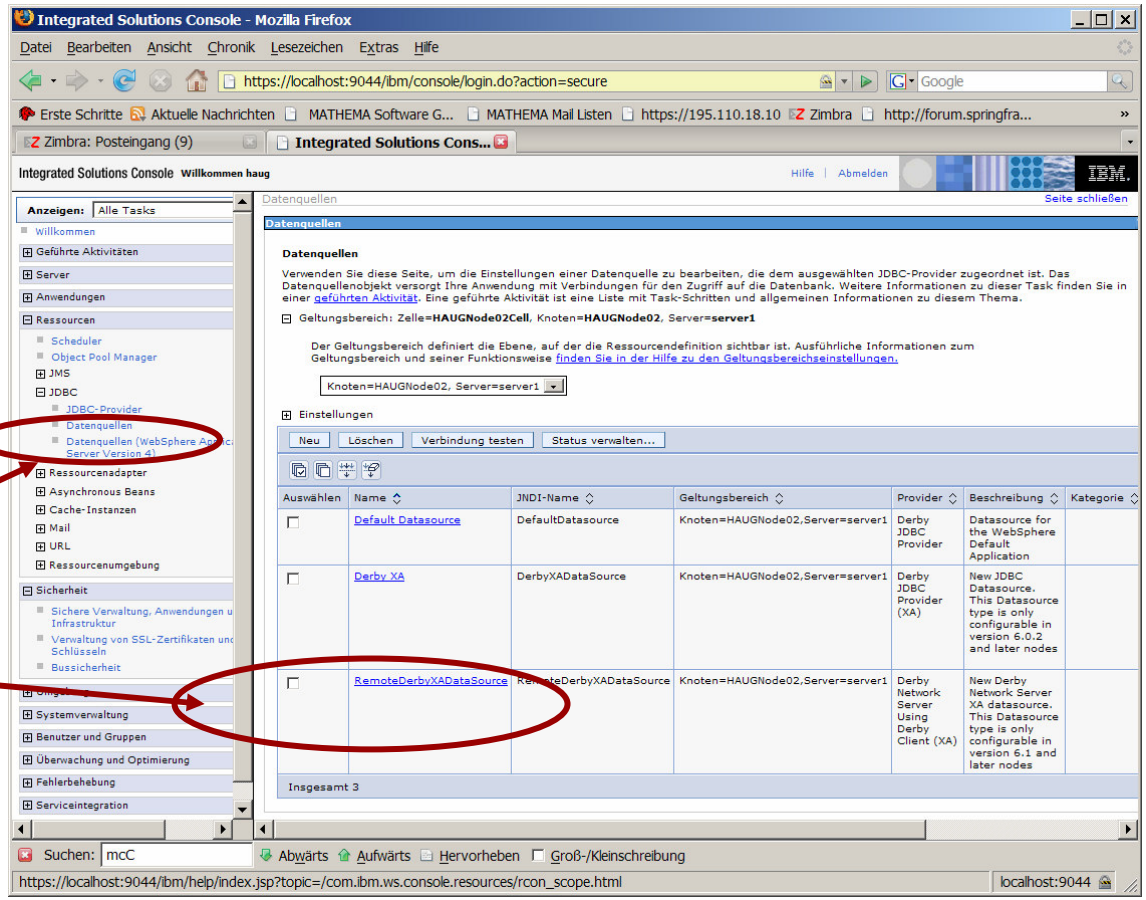
- Zertifikate erzeugen und einrichten

Zertifikate für SSL einrichten



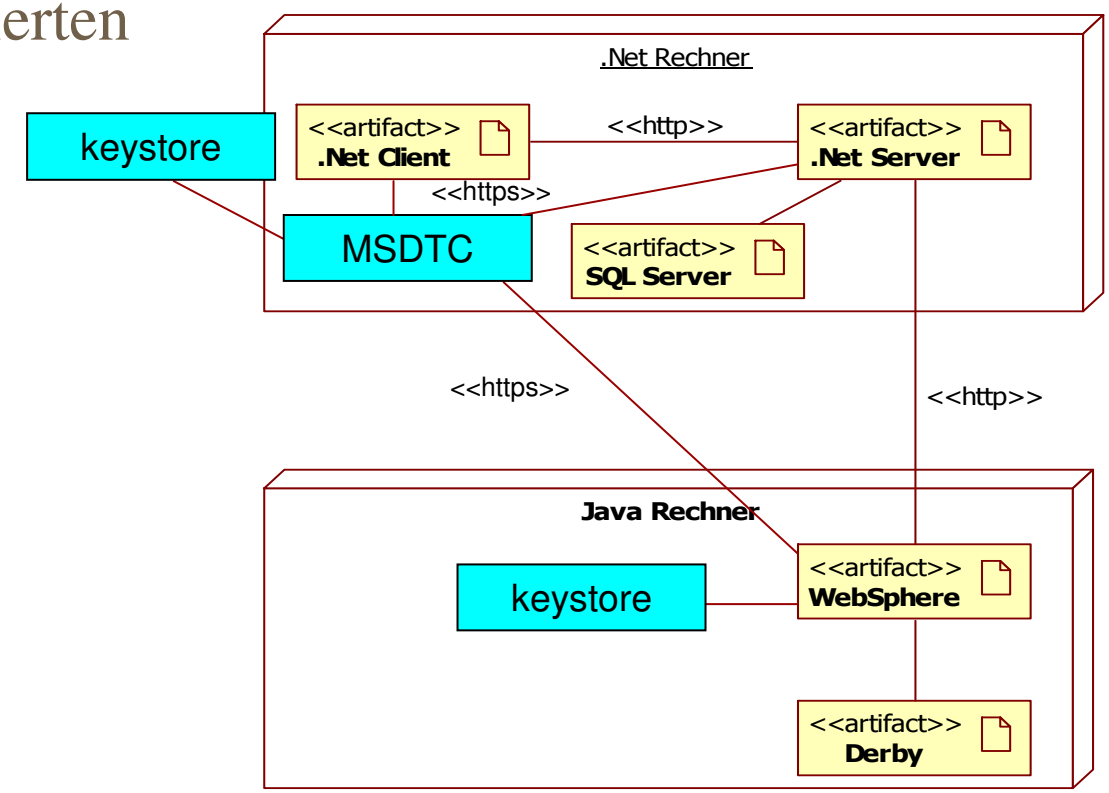
Transaktionen – WCF mit Java Service (8/9)

- XA DataSource



Transaktionen – WCF mit Java Service (9/9)

- Server starten und los geht's…
- Kommunikationspfade des implementierten Systems



Transaktionen – Agenda

- Motiviation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Kompensation - Motivation

- Das 2PC Protokoll erfordert eine enge Kopplung zwischen Teilnehmern und Transaktionsmanager
- Das 2PC Protokoll ist für „kurz-laufende“ Transaktionen geeignet
- Diese Anforderungen sind in einem SOA Umfeld schwer durchzusetzen
- Im SOA Umfeld soll durch lose Kopplung der Kommunikationspartner Robustheit geschaffen werden
- Üblicherweise sind Geschäftsprozesse „lang-laufende“ Transaktionen

Kompensation – Grundprinzipien I

- Eine Alternative zum 2PC Protokoll ist die Kompensation.
- Für Aktionen werden entsprechende Gegenaktionen definiert
- Es werden die ACID Prinzipien bewusst aufgeweicht
- Atomarität
  - Ggf. „Partielle Transaktionen“ durchführen
    - Z. B. Buchung eines Hotels ohne den zugehörigen Flug zu buchen
  - Ist ähnlich zu geschachtelten Transaktionen

Kompensation – Grundprinzipien II

- Isolation
  - Die Teilnehmer bestimmen, wann sie Commit ausführen
    - Möglichst frühzeitig
  - Die Teilnehmer definierten kompensierende Aktionen
    - Kann ein Rollback sein
    - Kann aber auch „etwas anderes“ sein, z.B. eine Stornierung
- Konsistenz
  - Da die Isolation aufgeweicht wird, wird zwangsläufig die Konsistenz „verletzt“.
  - Eine „laufende“ Transaktion kann „global gesehen“ einen inkonsistenten Zustand haben

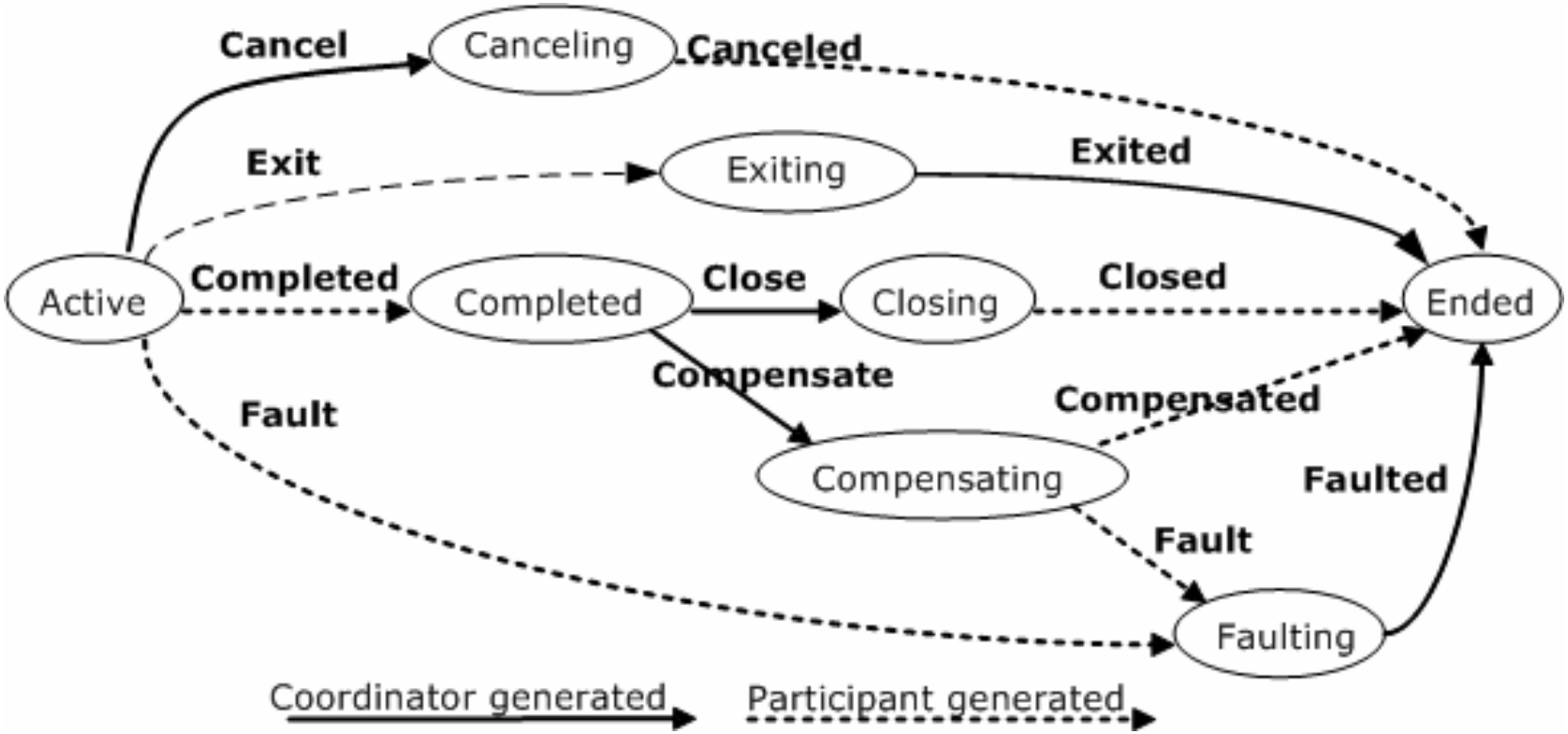
Kompensation – WS Business Activity I

- Definiert einen Standard für langlaufende Transaktionen mit Kompensation
- Standard der OASIS
- Aktuelle Version 1.1 (2007)
- Es werden zwei Modelle (Protokolle) unterstützt
  - BusinessAgreementWithParticipantCompletion
  - BusinessAgreementWithCoordinatorCompletion
- Keine direkte Unterstützung in WCF
- Beispiel Implementierung in JBOSS TS vorhanden

Kompensation – WS Business Activity II

- **BusinessAgreementWithParticipantCompletion Protocol**

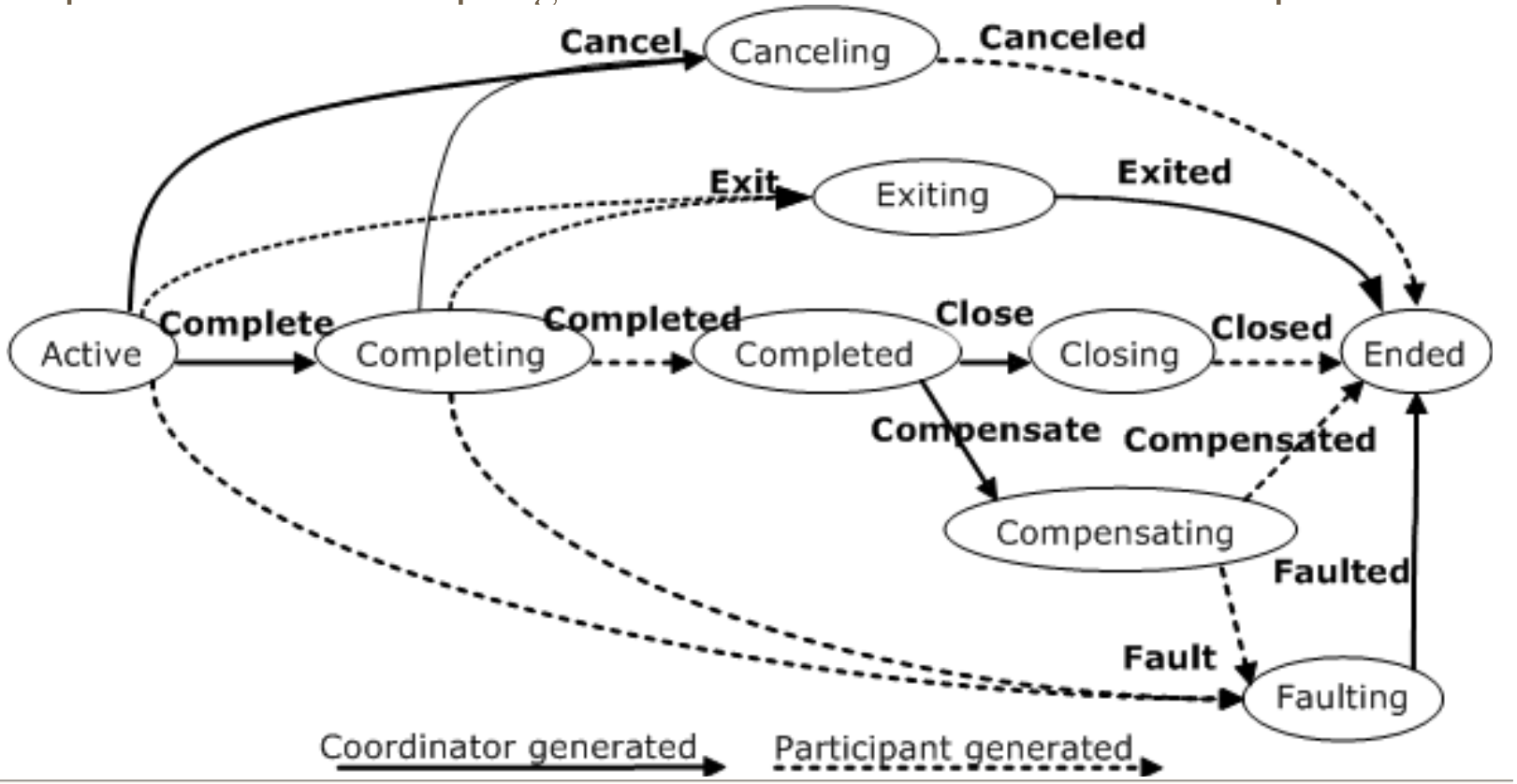
  http://schemas.xmlsoap.org/ws/2004/10/wsba/ParticipantCompletion



Kompensation – WS Business Activity III

- BusinessAgreementWithCoordinatorCompletion

  http://schemas.xmlsoap.org/ws/2004/10/wsba/CoordinatorCompletion



Transaktionen – Agenda

- Motiviation
- Theorie
  - Transaktionseigenschaften
  - lokale und verteilte Transaktionen
  - WS-AtomicTransaction (WS-AT)
- WS-AT mit Windows Communication Foundation (WCF)
  - WCF Überblick
  - WS-AT mit WCF
  - WS-AT Transaktionen mit WCF und Java
- Kompensation
- Fazit und Literaturhinweis

Transaktionen – Fazit (1/2)

- Die Transaktionshandhabung bettet sich nahtlos in die WCF ein
- Erstellen einer transaktionalen Anwendung mittels WCF ist relativ unkompliziert
- IBM WebSphere erleichtert das Erstellen Transaktionaler Services

Transaktionen – Fazit (2/2)

- Durch den Einsatz von WS-AT mit verschlüsselter Kommunikation zum MSDTC wird das Setup und der Betrieb (während der Implementierung) wesentlich komplizierter

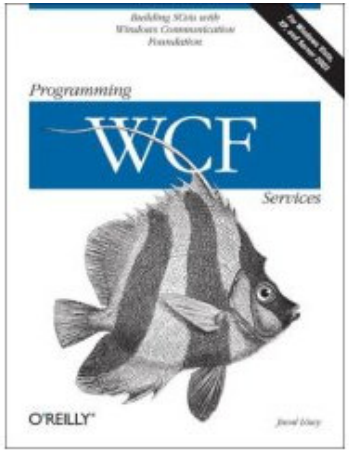
- Interoperabilität, Stabilität und Dokumentation lassen an manchen Stellen sowohl bei WCF, MSDTC, WebSphere, JBoss zu wünschen übrig

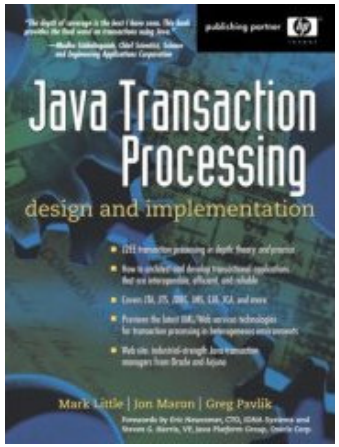
→ „Man sollte sich genau überlegen, ob WS-AT notwendig ist“
Alternativ kann Kompensation eingesetzt werden

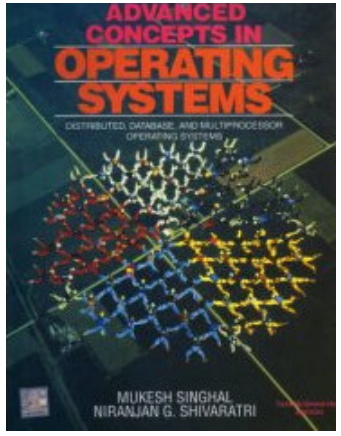
Transaktionen – Literatur

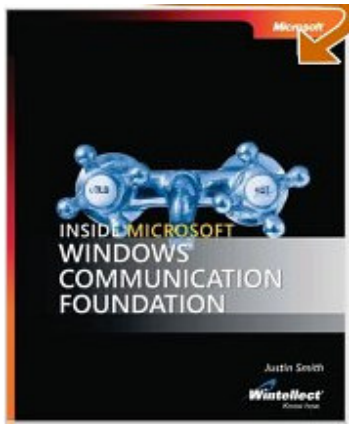
Webressourcen:

* http://research.microsoft.com/~Gray/papers/DBOS.pdf









15.–18. 09. 2008
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Thomas Haug
MATHEMA Software GmbH