

ASM-AG

**„ASMs und die Struktur und Dynamik von  
Web Services“**

- Studienarbeit -

Alexander Brade



Humboldt-Universität zu Berlin  
Institut für Informatik  
Unter den Linden 6  
D-10099 Berlin

# Inhaltsverzeichnis

Anmerkung: Alle mit \* markierten Einträge des Inhaltsverzeichnisses beziehen sich auf Darstellungen dynamischen Verhaltens, welches in Form einer ASM dargestellt wurde. Die Kapitel orientieren sich an denen in [WS]. Auf Beschreibungen der einzelnen Grafiken wird auf Grund des großen Umfanges verzichtet, die entsprechenden Erläuterungen sind dem Buch [WS] zu entnehmen. Diese Arbeit soll vielmehr zeigen, dass eine formale Darstellung der verschiedensten graphischen Darstellung, die in [WS] vertreten sind, mit Hilfe von algebraischen Spezifikationen und ASMs möglich ist.

<b>RELATIONEN UND ADTS ZUR DARSTELLUNG VON WEB-SERVICES.....</b>	<b>5</b>
<b>KAPITEL 1.....</b>	<b>16</b>
VERSCHIEDENE SCHICHTEN EINES INFORMATIONSSYSTEMS (GRAFIK SEITE 4/9): .....	16
ZUSAMMENFASSUNG VON SCHICHTEN IN DER 1-TIER ARCHITEKTUR (GRAFIK SEITE 11):.....	16
ZUSAMMENFASSUNG VON SCHICHTEN IN DER 2-TIER ARCHITEKTUR (GRAFIK SEITE 12):.....	16
INTERNE ORGANISATION DER ANWENDUNGSLOGIK-SCHICHT (GRAFIK SEITE 14):.....	17
CLIENTS ALS INTEGRATIONSWERKZEUG (GRAFIK SEITE 15): .....	17
3-TIER ARCHITEKTUREN UND MIDDLEWARE (GRAFIK SEITE 16):.....	18
INTEGRATION VON SYSTEMEN MIT UNTERSCHIEDLICHEN ARCHITEKTUREN (GRAFIK SEITE 17):.....	18
ERWEITERUNG EINES 3-TIER ZU EINEM N-TIER SYSTEM (GRAFIK SEITE 20): .....	19
PROZESSUNTERBRECHUNG AUF GRUND EINES SYNCHRONEN AUFRUFES (GRAFIK SEITE 23): .....	20
ASYNCHRONE AUFRUFE UND QUEUES (GRAFIK SEITE 25): .....	20
<b>KAPITEL 2.....</b>	<b>21</b>
ABSTRAKTION MIT RPC (GRAFIK SEITE 31):.....	21
ENTWICKLUNG VERTEILTER ANWENDUNGEN MIT RPC (GRAFIK SEITE 37): .....	21
RPC- GRUNDFUNKTIONEN (GRAFIK SEITE 38):.....	22
ERWEITERUNG DER GRUNDFUNKTIONEN UM DYNAMIC BINDING (GRAFIK SEITE 40):.....	23
DCE- ARCHITEKTUR (GRAFIK SEITE 44): .....	23
RESSOURCENVERWALTUNG MIT DBMS (GRAFIK SEITE 48A): .....	24
VERWALTUNG HETEROGENER RESSOURCEN MIT DBMS (GRAFIK SEITE 48B): .....	24
KAPSELUNG VON RPC-AUFRUFEN (GRAFIK SEITE 49)*: .....	25
GRUNDBESTANDTEILE EINES TP-MONITORS (GRAFIK SEITE 51): .....	28
HIGH-LEVEL SICHTWEISE AUF DIE CORBA-ARCHITEKTUR (GRAFIK SEITE 55):.....	29
ROLLE VON IDL-SPEZIFIKATIONEN BEI CLIENTS UND SERVERN (GRAFIK SEITE 56): .....	30
NACHRICHTENBASIERTE INTERAKTIONEN (GRAFIK SEITE 61A): .....	31
NACHRICHTENBASIERTE INTERAKTIONEN (GRAFIK SEITE 61B):.....	31
MESSAGE QUEUING MODEL (GRAFIK SEITE 62): .....	31
MESSAGE QUEUING MODEL WITH SHARED QUEUES (GRAFIK SEITE 63):.....	31
<b>KAPITEL 3.....</b>	<b>32</b>
NACHRICHTENBASIERTE INTERAKTION MIT NEUEN SYSTEMEN (GRAFIK SEITE 73): .....	32
MESSAGE BROKER UND BENUTZERDEFINIERTES ROUTING-LOGIK (GRAFIK SEITE 74): .....	32
MESSAGE BROKER UND DAS "PUBLISH/SUBSCRIBE"-MODELL (GRAFIK SEITE 76):.....	32
VERTEILTE MESSAGE-BROKER (GRAFIK SEITE 78 (3.5)): .....	33
HIGH-LEVEL ARCHITEKTUR VON EAI-SYSTEMEN (GRAFIK SEITE 78 (3.6)):.....	34
NACHRICHTENAUSTAUSCH UND PROZESSAUFRUFE (GRAFIK SEITE 80)*: .....	35
WORKFLOW-SPEZIFIKATION EINES RECHNUNGSPROZESSES (GRAFIK SEITE 85)*:.....	38
GRUNDAUFGABEN EINES WFMS (GRAFIK SEITE 86): .....	40
KOMBINATION VON WORKFLOW- UND EAI-TECHNIKEN (GRAFIK SEITE 90): .....	40
<b>KAPITEL 4.....</b>	<b>41</b>
CLIENT-/SERVER-INTERAKTION ÜBER ZWISCHENHÄNDLER (GRAFIK SEITE 96): .....	41
VERSCHLÜSSELUNG ZWISCHEN HTTPS SERVERN UND CLIENTS (GRAFIK SEITE 97):.....	41
ERWEITERUNG DER 3-TIER ARCHITEKTUR (GRAFIK SEITE 99): .....	41
APPLETS ALS MÖGLICHKEIT DER CLIENT-IMPLEMENTATION (GRAFIK SEITE 100): .....	42

CGI-PROGRAMME ALS INTERFACES (GRAFIK SEITE 101): .....	42
SERVLETS ZUR INTERAKTION MIT SERVERANWENDUNGEN (GRAFIK SEITE 103): .....	43
SCHICHTEN DER ANWENDUNGSSERVER (GRAFIK SEITE 104): .....	43
AUSZUG AUS DER J2EE-SPEZIFIKATION (GRAFIK SEITE 105): .....	43
ANWENDUNGSSERVER ZUR UNTERSTÜTZUNG DER ANWENDUNGSLOGIK (GRAFIK SEITE 106):.....	44
ANWENDUNGSSERVER ZUR UNTERSTÜTZUNG DES “PRESENTATION LAYERS” (GRAFIK SEITE 109):.....	45
MÖGLICHE VERBINDUNG ZWISCHEN ZWEI 3-TIER SYSTEMEN (GRAFIK SEITE 112): .....	46
INTEGRATION VON MIDDLEWARE-PLATTFORMEN (GRAFIK SEITE 114): .....	46
B2B INTERAKTION AUF MIDDLEWARE-LEVEL (GRAFIK SEITE 116): .....	47
<b>KAPITEL 5.....</b>	<b>47</b>
MANUELLE INTEGRATION FIRMENÜBERGREIFENDER PROZESSE (GRAFIK SEITE 126): .....	47
B2B INTEGRATION IM SINNE VON EAI (GRAFIK SEITE 127):.....	48
FIRMENÜBERGREIFENDE INTEGRATION (GRAFIK SEITE 128):.....	49
POINT-TO-POINT INTERAKTION (GRAFIK SEITE 129):.....	50
LÖSUNGSANSATZ VON WEBSERVICES (GRAFIK SEITE 134): .....	50
EINSTIEGSPUNKTE ZU LOKALEN SERVICES (GRAFIK SEITE 135):.....	51
ANWENDUNGSINTEGRATION MIT HILFE VON WEBSERVICES (GRAFIK SEITE 136):.....	52
GRUNDLAGE DER SERVICE-BESCHREIBUNGEN (GRAFIK SEITE 137): .....	52
GRUNDGERÜST DER SERVICE-INTERAKTIONEN (GRAFIK SEITE 139): .....	53
INTERNE UND EXTERNE ARCHITEKTUREN BEI WEBSERVICES (GRAFIK SEITE 142): .....	53
KONVENTIONELLE MIDDLEWARE ALS INTEGRATIONSPLATTFORM (GRAFIK SEITE 144): .....	54
IMPLEMENTATION VON WEBSERVICES IN VERBINDUNG MIT TIER-SYSTEMEN (GRAFIK SEITE 145):.....	55
EXTERNE ARCHITEKTUR VON WEBSERVICES (GRAFIK SEITE 146):.....	55
EXTERNE ARCHITEKTUR VON WEBSERVICES UND P2P-PROTOKOLLE (GRAFIK SEITE 148):.....	56
<b>KAPITEL 6.....</b>	<b>58</b>
AUFRUF VON WEBSERVICES MIT SOAP-NACHRICHTEN (GRAFIK SEITE 153): .....	58
WSDL-SPEZIFIKATIONEN ANALOG ZU IDL-SPEZIFIKATIONEN (GRAFIK SEITE 154):.....	58
UDDI-GRUNDIDEE (GRAFIK SEITE 155): .....	59
SCHEMATISCHE DARSTELLUNG EINER SOAP-NACHRICHT (GRAFIK SEITE 157): .....	59
DOKUMENTENBASIERTE INTERAKTION (GRAFIK SEITE 159): .....	60
RPC-BASIERTE INTERAKTION (GRAFIK SEITE 159):.....	60
BEISPIEL FÜR EINER SOAP-NACHRICHT (GRAFIK SEITE 160): .....	60
RPC-AUFRUFE VIA HTTP MIT SOAP (GRAFIK SEITE 162): .....	61
EINE EINFACHE SOAP-IMPLEMENTIERUNG (GRAFIK SEITE 163):.....	62
BEISPIEL FÜR EINE WSDL SERVICE-SPEZIFIKATION (GRAFIK SEITE 167):.....	63
WSDL-SPEZIFIKATION EINES SERVICES (GRAFIK SEITE 170):.....	63
ERZEUGUNG VON WSDL-DOKUMENTEN MIT HILFE VON APIS (GRAFIK SEITE 173): .....	64
SCHEMATISCHE DARSTELLUNG EINES UDDI-EINTRAGES (GRAFIK SEITE 177): .....	64
INTERAKTION ZWISCHEN UDDI-REGISTRIES (GRAFIK SEITE 181):.....	66
SCHEMATISCHER ZUSAMMENHANG ZWISCHEN UDDI UND WSDL (GRAFIK SEITE 183): .....	67
ANBIETEN EINES INTERNEN SERVICES ALS WEBSERVICE (GRAFIK SEITE 185):.....	67
<b>KAPITEL 7.....</b>	<b>68</b>
KOMMUNIKATION ZWISCHEN EINEM CLIENT UND EINEM WEBSERVICE (GRAFIK SEITE 198)*: .....	68
ZUSTANDSGRAPH EINER KONVERSATIONS-SPEZIFIKATION (GRAFIK SEITE 201): .....	69
BESTELLVORGANG ZWISCHEN ZWEI WEBSERVICES (GRAFIK SEITE 202 (7.3))*:.....	69
ALLGEMEINES PROTOKOLL FÜR BESTELLVORGÄNGE (GRAFIK SEITE 202 (7.4))*: .....	70
DAS BESTELL-PROTOKOLL ALS SEQUENZDIAGRAMM (GRAFIK SEITE 204):.....	72
DAS BESTELL-PROTOKOLL ALS AKTIVITÄTSDIAGRAMM (GRAFIK SEITE 205)*:.....	72
DAS BESTELL-PROTOKOLL AUS KUNDENSICHT (GRAFIK SEITE 207)*:.....	73
HORIZONTALE UND VERTIKALE WEBSERVICE-PROTOKOLLE (GRAFIK SEITE 208): .....	74
CLIENT – WEBSERVICE INTERAKTION (GRAFIK SEITE 209): .....	75
AUFGABE DES “CONVERSATION CONTROLLERS” (GRAFIK SEITE 210): .....	75
“CONVERSATION CONTROLLER” ALS TEIL EINES SOAP-ROUTERS (GRAFIK SEITE 211):.....	76
INFRASTRUKTUR ZUR IMPLEMENTIERUNG HORIZONTALER PROTOKOLLE (GRAFIK SEITE 214):.....	76
KOMMUNIKATION VON PORT-REFERENZEN UND ROLLENINFORMATIONEN (GRAFIK SEITE 215):.....	77
ZENTRALISIERTE WS-KOORDINATION (GRAFIK SEITE 216):.....	77
VERTEILTE WS-KOORDINATION (GRAFIK SEITE 216): .....	78
ERSTELLUNG / AKTIVIERUNG EINES KOORDINATIONSKONTEXTES (GRAFIK SEITE 218): .....	78

WS-REGISTRIERUNG BEI EINEM KOORDINATOR (GRAFIK SEITE 219): .....	79
KONVENTION ZUR DEFINITION VON PROTOKOLL-SPEZIFISCHEN SCHNITTSTELLEN (GRAFIK SEITE 220): .....	79
NACHRICHTENAUSTAUSCH ÜBER EINEN ZENTRALEN KOORDINATOR (GRAFIK SEITE 221)*: .....	80
NACHRICHTENAUSTAUSCH BEI VERTEILTER KOORDINATION (GRAFIK SEITE 222)*: .....	82
VERKNÜPFUNG VON KOORDINATOREN BEI VERTEILTER KOORDINATION (GRAFIK SEITE 224): .....	84
PORT-TYPEN BEI ATOMAREN TRANSAKTIONEN (GRAFIK SEITE 229): .....	84
BEISPIEL FÜR ATOMARE TRANSAKTIONEN (GRAFIK SEITE 231)*: .....	85
PORT-TYPEN BEI GESCHÄFTSPROZESSEN (GRAFIK SEITE 232): .....	89
BEISPIELHAFTER AUSFÜHRUNG DES GESCHÄFTSPROZESSPROTOKOLLES (GRAFIK SEITE 233)*: .....	89
ROSETTANET PIP 3A2 (GRAFIK SEITE 237): .....	91
ROSETTANET PROTOKOLL UND NACHRICHTENKOMPONENTEN (GRAFIK SEITE 239): .....	91
<b>KAPITEL 8.....</b>	<b>92</b>
CLIENT-KONVERSATION MIT MEHREREN WEBSERVICES (GRAFIK SEITE 246): .....	92
ZUSAMMENGESETZTE SERVICES (GRAFIK SEITE 247): .....	93
IMPLEMENTATION VON ZUSAMMENGESETZTEN WEBSERVICES (GRAFIK SEITE 249): .....	93
GRUNDLEGENDE BESTANDTEILE EINER WS-COMPOSITION MIDDLEWARE (GRAFIK SEITE 250): .....	94
IMPLEMENTATION ZUSAMMENGESETZTER WS ÜBER WS-MIDDLEWARE (GRAFIK SEITE 251): .....	95
EXTERNE INTERAKTION VS. EXTERNE IMPLEMENTATION (GRAFIK SEITE 252): .....	96
PROZESSDIAGRAMM EINES LIEFERANTEN-WEBSERVICES (GRAFIK SEITE 258)*: .....	96
STATECHART DES LIEFERANTEN-WEBSERVICES (GRAFIK SEITE 260)*: .....	98
PETRINETZDARSTELLUNG DES LIEFERANTENPROZESSES (GRAFIK SEITE 261): .....	99
PROZESSHIERARCHIE DES LIEFERANTEN-PROZESSES (GRAFIK SEITE 263): .....	100
DATENFLUSSANSATZ FÜR DIE DATENÜBERTRAGUNG ZWISCHEN KOMPONENTEN (GRAFIK SEITE 267): .....	100
SERVICEAUSWAHL ANHAND VON UDDI-EINTRÄGEN (GRAFIK SEITE 268)*: .....	101
DYNAMIC BINDING (GRAFIK SEITE 269): .....	102
BEISPIEL FÜR EINE ATOMARE REGION (GRAFIK SEITE 270): .....	103
ACID-TEILTRANSAKTIONEN (GRAFIK SEITE 272): .....	103
FEHLERBEHANDLUNG INNERHALB EINES PROZESSES (GRAFIK SEITE 274)*: .....	104
KOORDINATIONSprotokoll EINES BESTELLVORGANGES (GRAFIK SEITE 277): .....	105
HÄNDLERSICHT AUF DAS KOORDINATIONSprotokoll (GRAFIK SEITE 278)*: .....	105
BESCHREIBUNG DER ROLLE DES HÄNDLERS IM KOORDINATIONSprotokoll (GRAFIK SEITE 279)*: .....	106
IMPLEMENTATION DER LIEFERANTEN-ROLLE (GRAFIK SEITE 280)*: .....	108
ROUTING-PROBLEM DER KOMPOSITIONSKOMPONENTE (GRAFIK SEITE 283): .....	109
UMFANG DER BPEL-SPEZIFIKATION (GRAFIK SEITE 285)*: .....	110
PROZESSSTRUKTUREN IN BPEL (GRAFIK SEITE 287)*: .....	112
PARTNERDEFINITIONEN IN BPEL (GRAFIK SEITE 290): .....	113
FEHLERBEHANDLUNG IN BPEL (GRAFIK SEITE 291): .....	114
KORRELATION ZWISCHEN NACHRICHTEN (GRAFIK SEITE 293): .....	114
<b>KAPITEL 9.....</b>	<b>115</b>
CLOSED COMMUNITY WITH DIRECT INTERACTION (GRAFIK SEITE 304): .....	115
INTERAKTIONSVERMITTLUNG MIT HUBS (GRAFIK SEITE 306): .....	115
KONTROLLE VON MIDDLEWAREKOMPONENTEN DURCH EIN WSMS (GRAFIK SEITE 310): .....	116
EXTERNE ARCHITEKTUR DES WS-MANAGEMENTS (GRAFIK SEITE 315): .....	116
MANAGEMENT OUTSOURCING (GRAFIK SEITE 317): .....	117
WECHSELWIRKUNGEN ZWISCHEN SYSTEMEN VIA XMLCIM (GRAFIK SEITE 318): .....	117
AUSTAUSCH VON MANAGEMENTINFORMATIONEN MIT HILFE VON XMLCIM (GRAFIK SEITE 319): .....	117

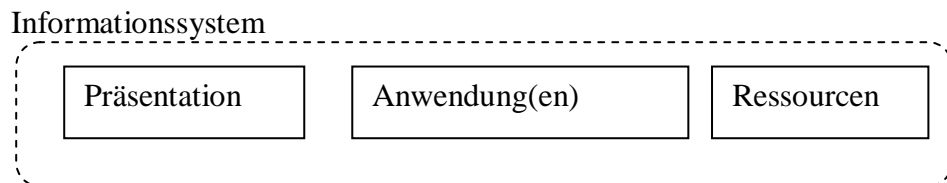
## Relationen und ADTs zur Darstellung von Web-Services

Bevor wir zur Erzeugung von ASM aus der graphischen Repräsentation von Web Services übergehen, soll eine Übersicht über alle auftretenden graphischen Strukturelemente und die ihnen entsprechenden *Relationen* bzw. **ADTs** erfolgen. Es wird versucht, auf die Strukturelemente in der Reihenfolge ihres Auftretens im zu Grunde liegenden Buch einzugehen.

Wie auch in [WS] stehen im folgenden Namen bzw. Bezeichnungen der jeweiligen Komponente entweder am inneren oder äußeren Rand des jeweiligen Strukturelements. Evtl. aufgeführte Seitenzahlen geben das erste Auftreten des entsprechenden Elements an.

### Übersicht der Strukturelemente:

- 1) Zur losen Zusammenfassung mehrerer Komponenten zu einem Oberbegriff bzw. zu einer größeren Komponente (Seite 4) wird die folgende graphische Darstellung verwendet.



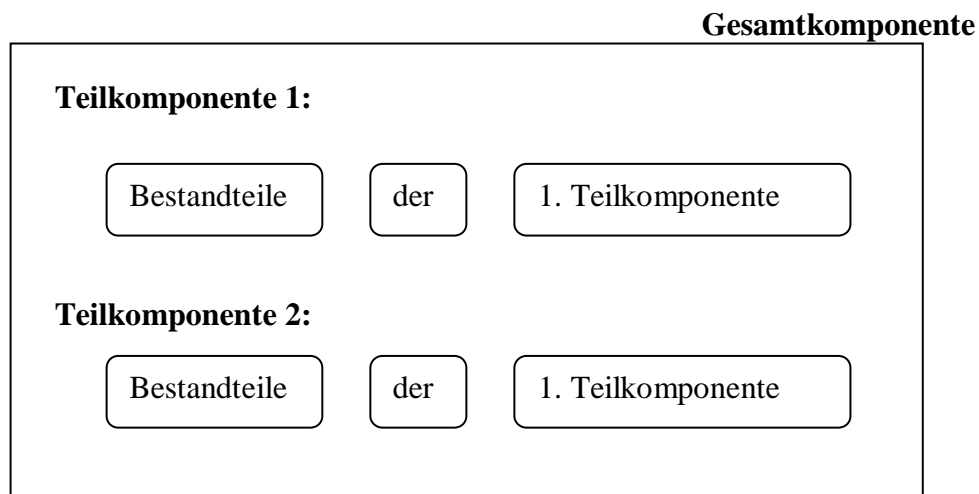
Formal wird hierzu der ADT **Container** eingeführt. Die Zugehörigkeit zu einer Komponente wird durch eine *partOf*-Relation ausgedrückt. Das obige Beispiel lässt sich also auch wie folgt darstellen:

Informationssystem  $\in$  **Container**

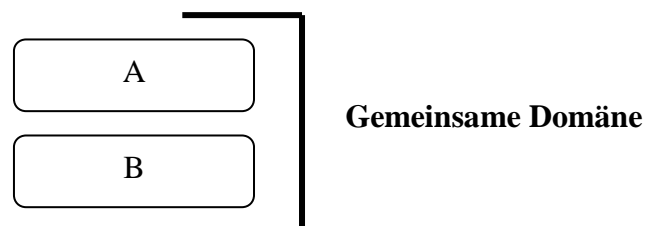
{Präsentation, Anwendung(en), Ressourcen} *partOf* Informationssystem

Auf die hier noch nicht näher definierten Objekte (z.B. Präsentation) wird im Punkt 3) eingegangen.

Es seien an dieser Stelle noch weitere graphische Möglichkeiten (Seite 55) zur Darstellung von Teilkomponenten aufgezeigt:



und



Alle gezeigten Darstellungen können auch zur Veranschaulichung von gemeinsamen z.B. Domänen-Bereichen dienen (Seite 78, Seite 105).

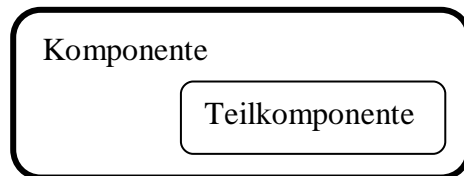
- 2) Die Darstellung von nicht näher beschriebenen Komponenten (auch: extern, unabhängig... –Seite 4) erfolgt im Buch in dieser Form:



Der entsprechende ADT ist **Object** und würde für das gegebene Beispiel so aussehen:

Client ∈ **Object**

Falls es sich um ein strukturiertes Objekt handelt, so wird teilweise zur Unterstreichung dieser Tatsache die folgende Darstellung gewählt:



Mit Hilfe von ADTs und Relationen ausgedrückt erhält man:

Komponente ∈ **structObject**

Teilkomponente ∈ **Object**

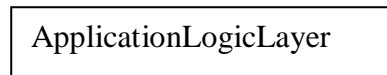
{Teilkomponente} *partOf* Komponente bzw. Teilkomponente *partOf* Komponente

- 3) Komponenten mit im Text klar beschriebenen Aufgaben/Schichten bzw. eindeutig als solche zu erkennende Aktivitäten bzw. Prozesse (–Seite 4) werden durch **Activity** repräsentiert. Analog zu den in Punkt 2) erwähnten strukturierten Objekten gibt es auch die Möglichkeit, strukturierte Aktivitäten (Seite 11) darzustellen –in Form von **structActivity**.

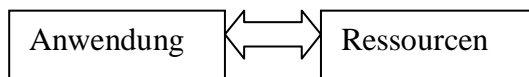
Will man z.B. den ApplicationLogicLayer als Schicht definieren, so erhält man:

ApplicationLogicLayer ∈ **Activity**

Die entsprechende graphische Repräsentation ist:



- 4) Sind zwei Komponenten auf diese Art



miteinander verbunden, so hängt die Funktionalität der einen (stark) von der der anderen ab (Seite 4). Dies kann soweit gehen, dass die Komponenten einzeln nicht verwendbar sind (z.B. Abhängigkeit von einem Gesamtsystem).

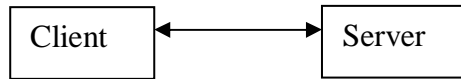
Die Relation, die diesem Sachverhalt Rechnung trägt ist *con*. Das genannte Beispiel kann also auch so notiert werden:

Anwendung ∈ **Activity**

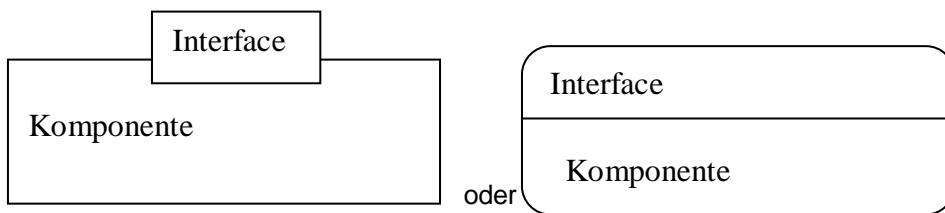
Ressourcen ∈ **Activity**

Anwendung *con* Ressourcen

- 5) Die Relation `comm` stellt den Kommunikations-/Informationsfluss zwischen zwei Komponenten dar. Dies könnte z.B. eine Beziehung zwischen einem n-tier System und einer Komponente bzw. einem anderen n-tier System sein.  
 Der Informationsfluss zwischen einem Client und einem Server wird dementsprechend notiert als:  
 Client `comm` Server, wobei Client  $\in$  **Activity** und Server  $\in$  **Activity**.  
 Sofern kein neuer ADT vorgestellt wird, wird im Folgenden der Datentyp der vorkommenden Objekte nicht mehr explizit genannt –er ist dann der graphischen Darstellung des jeweiligen Beispiels zu entnehmen. Für das Client/Server Beispiel wäre diese:



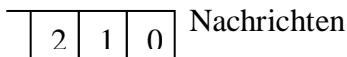
- 6) Besitzt eine Komponente ein Interface, so wird dies im Buch wie folgt dargestellt (Seite 14 bzw. 142):



Will man diesen Zusammenhang nicht-graphisch darstellen, so erfolgt dies über die `interfaceOf`-Relation:

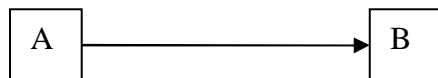
Interface `interfaceOf` Komponente

- 7) Neben Aktivitäten und Objekten gilt es unter anderem auch sog. Queues (Seite 25) zu betrachten:

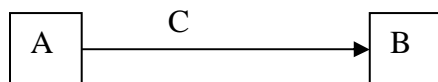


Der dazugehörige ADT wird mit **Queue** bezeichnet. Nachrichten  $\in$  **Queue** gibt also das obige Beispiel wieder. Hierbei kann auf jedes Queue-Element einzeln zugegriffen werden – gekennzeichnet wird dies durch z.B. Pfeile zum bzw. vom jeweiligen Element hin oder weg (Nachrichten[0] entspricht dem ersten Elemente der Queue).

- 8) Besteht zwischen mehreren Aktivitäten eine bestimmte Reihenfolge (Seite 38), so kann dies mit der Relation `pred(ecessor)Of` dargestellt werden.  
 A `predOf` B bedeutet demnach, dass die Aktivität A vor der Aktivität B stattfindet.



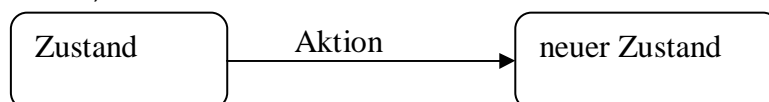
Ist der Pfeil darüber hinaus beschriftet, wie dies hier



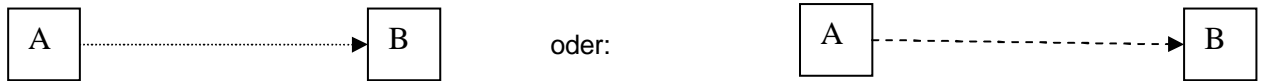
der Fall ist, so spiegelt dies einen Funktionsaufruf (Seite 202) wieder. Nach Einführung der Relation `invokeOp` lässt sich das auch wie folgt darstellen:

`invokeOp` (A, C, B)

Die Beschriftung des Pfeils entspricht dem Namen der aufgerufenen Funktion. Wird im hier gelieferten Beispiel also die Aktivität A ausgeführt und dann die Funktion C (z.B. innerhalb der Aktivität A) aufgerufen, so führt dies zur Ausführung von Aktivität B. Dies kann natürlich auch im Sinne von Zustandsübergängen verstanden werden, die beteiligten Typen sind dann jedoch Objekte (Seite 201):

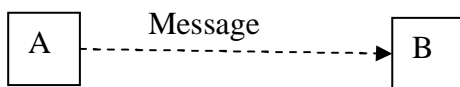


9) Die Folgenden beiden Skizzen drücken eine Vorbedingung, Teilnahme oder Beeinflussung aus:



Ist die Aktivität A eine Vorbedingung der Aktivität B, so kommt die Relation `allows` zur Anwendung: `A allows B`  
 Sollte A jedoch an der Aktivität B teilnehmen, so kann auch geschrieben werden: `A participates B`  
 Sofern die Teilnahme oder Beeinflussung über (mehrere) Zwischenstationen erfolgt, so dient zur Darstellung dieses Umstandes die Relation `allowsVia`. Die Notation dieser Relation entspricht der der Relation `sendVia` unter Punkt 18).

10) Ähnlich dem Aufruf von Funktionen wird auch das Versenden von Nachrichten gehandhabt:



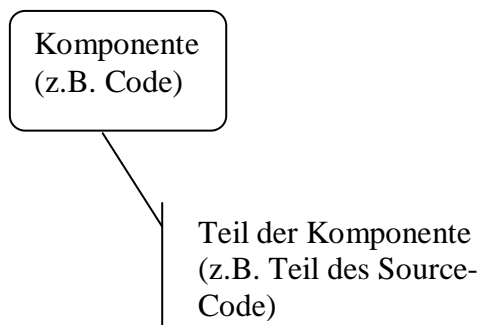
Dementsprechend wird die zu verwendende Relation `sendMsg` analog zu der Relation `invokeOp` definiert und angewandt: `sendMsg(A, Message, B)`.

11) Für die auf Seite 37 erfolgte Darstellung von Ressourcen führen wir den neuen ADT **Resource** ein. Die Folgende



sieht in ADT-Schreibweise so aus:  
`Datenbank ∈ Resource`

12) Will man einen Teil einer Komponente besonders hervorheben (Seite 37) weil dieser z.B. für die Interaktion mit der Außenwelt zuständig ist, so kann dies graphisch wie folgt aussehen:



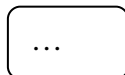
Die, dieses graphische Strukturelement wiedergebende, Relation ist `elementOf: TeilDerKomponente elementOf Komponente`.

- 13) Diese graphische Darstellung von mehreren Objekten bzw. Komponenten des gleichen Typs (Seite 55):



wird durch die Relation `numberOf` repräsentiert und gibt das genannte Beispiel wie folgt wieder:  
`numberOf ObjektA`  
 Selbiges geht natürlich auch mit den restlichen Komponenten- bzw. Interfacedarstellungsformen (also z.B. Aktivitäten).

- 14) Zur Darstellung weiterer, nicht näher benannter, Komponenten dient dieses Symbol (Seite 55):



Der dazugehörige ADT ist **undefObj**. Da die Elemente dieses Typs keine Bezeichnungen besitzen, werden sie in den Strukturbeschreibungen dieses Dokumentes mit Nummern versehen. Dies erfolgt entsprechend diesem Beispiel:

`-(1) ∈ undefObj`

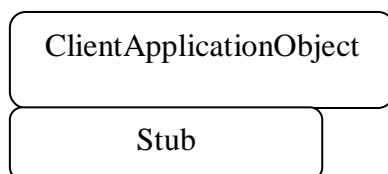
Weitere unbenannte Komponenten würden dann mit “-(2)”, “-(3)” usw. bezeichnet werden.

- 15) Die Darstellung einer Spezifikation bzw. eines Interfaces ohne eine zugehörige Komponente (Seite 56) kann mit dem folgenden Symbol erfolgen:

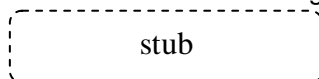


Als ADT dient hierzu der **Specification** –Typ. `IDLofServiceProvider ∈ Specification` gibt also den gleichen Sachverhalt wieder.

- 16) Die Relation `equal` stellt gleichgestellte, zusammengehörige Komponenten, wie dies z.B. auf Seite 56 anzutreffen ist, dar. Die graphische Notation von `Stub equal ClientApplicationObject` sieht so aus:



- 17) Die graphische Darstellung neu hinzugekommener Komponenten (Seite 73) wird zu gleich auch für die Veranschaulichung von optionalen Komponenten verwendet (Seite 239):

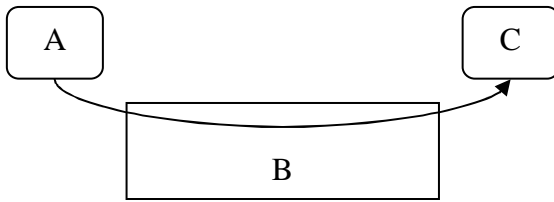


„stub“ kann also eine optionale oder eine neu hinzugekommene Komponente sein. Um dies mit Hilfe von Relationen wiederzugeben ist wie folgt vorzugehen:

`stub ∈ Object`

`new stub`

- 18) Werden Nachrichten unter Verwendung von Mittlern (Seite 61) versandt, so wird dies so dargestellt:

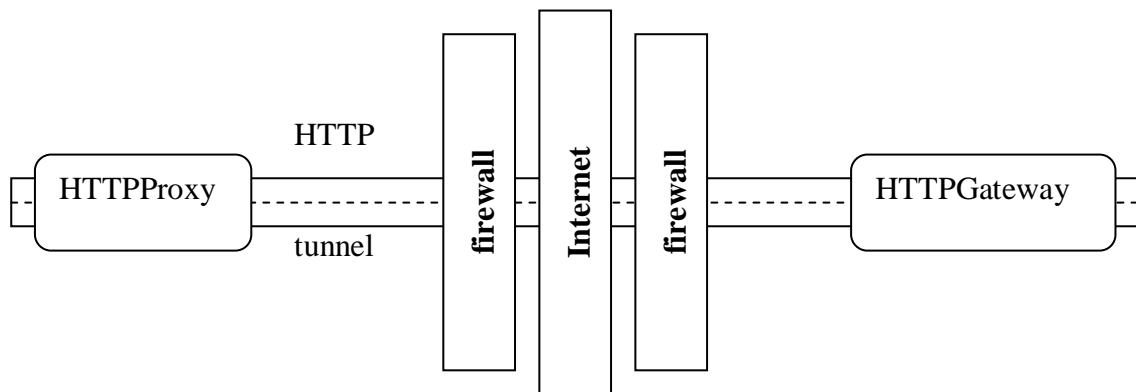


Die dies widerspiegelnde Relation heißt `sendVia` und stellt sich für das Beispiel so dar:

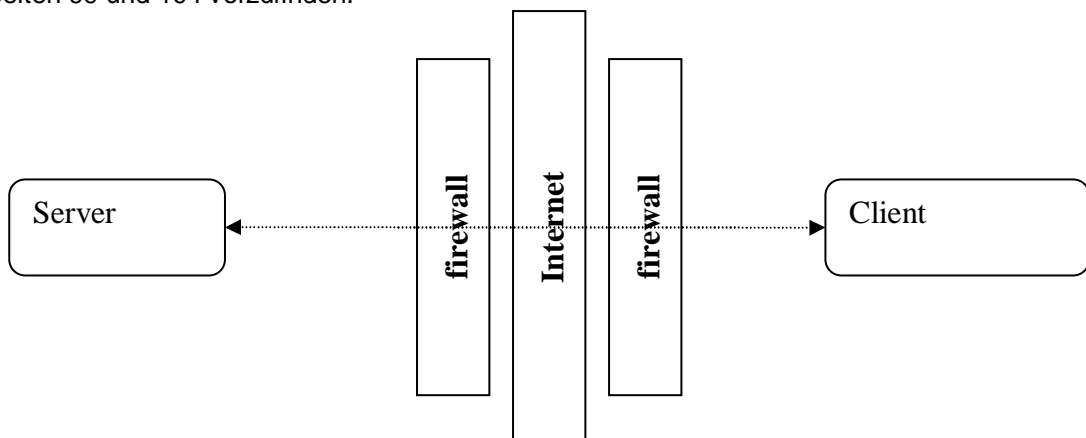
`sendVia (A, {B}, C)`

Analog zu dieser Notation ist auch `allowsVia` (9.) definiert bzw. anzuwenden.

- 19) Im Zuge des Internets (als Beispiel) kommt es auch zur Kommunikation durch andere Komponenten hindurch (z.B. Firewalls –Seite 96). Dies sieht graphisch dargestellt so aus:



Eine alternative Darstellungsmöglichkeit, bei der der Tunnel nicht benannt wird, ist auf den Seiten 99 und 104 vorzufinden.



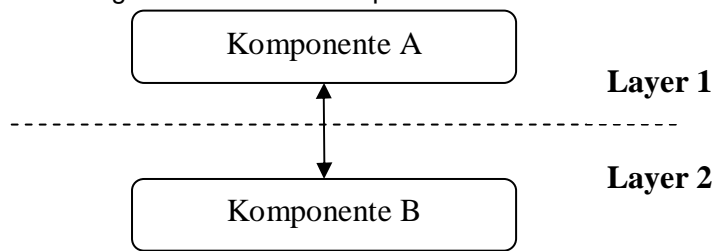
Um diese beiden Darstellungsformen wiederzugeben wird zum einen der ADT **Tunnel** und zum anderen die Relation `tunnelled` eingeführt:

`HTTPTunnel` ∈ **Tunnel**

`tunnelled (HTTPProxy, {Firewall, Internet, Firewall}, HTTPGateway, [HTTPTunnel])`

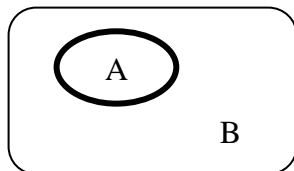
Die eingeführte Relation besagt, dass zwischen HTTPProxy und HTTPGateway eine Verbindung mit dem Namen HTTPTunnel besteht, welche durch die beiden Firewalls und das Internet hindurchgeht. Die Angabe des Namens der Verbindung ist optional und hat nur zu erfolgen, wenn dies in der graphischen Darstellung ebenfalls erfolgte.

- 20) Die im Buch anzutreffende Abgrenzung zwischen Komponenten verschiedener Schichten bzw. die lose Zweiteilung innerhalb einer Komponente



unterscheidet sich zwar von der, die bei Teilkomponenten zum Einsatz kommt, auf Grund der inhaltlich gleichen Aussage wird jedoch die gleiche ADT/Relationen -Beschreibung verwandt wie in 1).

- 21) Wird ein Objekt durch eine andere Komponente implementiert (Seite 100-103), so wird dies durch die Relation `implements` bzw. die anschließende graphische Darstellung ausgedrückt.  
`B implements A` und



sagen beide aus, dass A als B implementiert wird (Client z.B. als Applet)  
 Hierbei tritt auch ein neuer ADT **RawObject** ( $A \in \text{RawObject}$ ) auf.

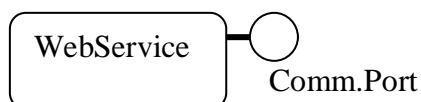
- 22) Betrachtet man `WebServices`, so kommt man nicht umhin mögliche Zugangspunkt(e) bzw. sog. `entry point(s)` zu betrachten (Seite 135).  
 Sind diese unbenannt, so genügt die Verwendung der Relation `entryPoints`.  
`entryPoints (WebService)` würde dann diese Situation wiedergeben:



Beide Darstellungen stellen das gleiche dar. Evtl. Interaktionen mit dem entsprechenden `WebService` finden dann über diese Ports statt.

Sollen die Ports jedoch benannt werden, so sieht die Definition wie folgt aus:

`Comm.Port`  $\in$  **Port**  
`Comm.Port` `portOf` `WebService`



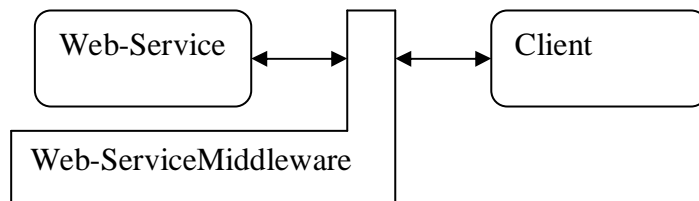
Verbindungen zu den Ports können in diesem Fall direkt über den Port-Namen erfolgen.

- 23) Organisations-, Struktur- oder Architektureinheit (Seite 142) werden symbolisch so dargestellt:



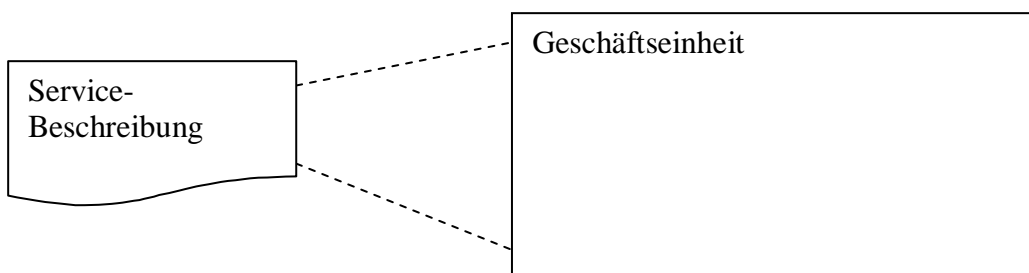
Der dazugehörige ADT ist **Structure**. Hier gilt demnach: `Architektur`  $\in$  **Structure**

- 24) Sofern (interne) Middleware besonders hervorgehoben werden sollte, so erfolgte dies entsprechend diesem Beispiel:



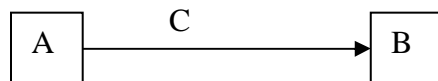
Mit Hilfe des ADT **Middle** erhält man:  
 Web-ServiceMiddleware  $\in$  **Middle**  
 Web-Service  $\text{comm}$  Web-ServiceMiddleware  
 Client  $\text{comm}$  Web-ServiceMiddleware

- 25) Die hier gezeigte Darstellung für die Verfeinerung von Komponenten (Seite 183):



Findet sich in der Relation *refines* wieder:  
 Geschäftseinheit *refines* Service-Beschreibung

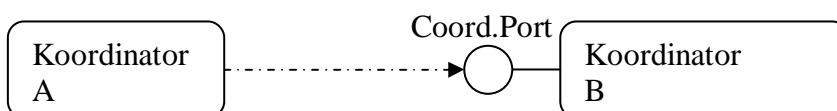
- 26) Stellen Bezeichnungen an (Verbindungs- oder Kommunikations-) Kanten z.B. verwendete Protokolle (Seite 209) oder (Rollen-) Informationen (Seite 215) und nicht Aktionen (siehe *invokeOp*) bzw. versendete Nachrichten (siehe *sendMsg*) dar, kommt die Relation *label* zum Einsatz. Hier ein kleines Beispiel, indem die Aktivität A unter Verwendung des Protokolls C mit der Aktivität B in Verbindung steht.



Dies entspricht:  
 $\text{label}((A \text{ predOf } B), C)$   
 Anstelle der Relation *predOf* sind auch alle anderen „Pfeilrelationen“ wie z.B. *con* und *comm*, aber auch *rel*, möglich.

- 27) Die Koordination zwischen Komponenten (Seite 224) wird durch den hier zu sehende Pfeil und die Relation *coord* ausgedrückt:

KoordinatorA *coord* Coord.Port  
 und

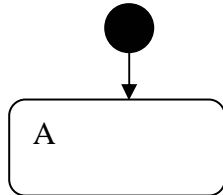


28) Zur Darstellung von Start- und Endpunkte eines Prozesses reichen nach Einführung des ADTs **Point** die bisher zur Verfügung stehenden Relationen:

$-(1) \in \mathbf{Point}$

$-(1) \text{ predOf } A$

Die entsprechende graphische Darstellung des Startpunktes eines Prozesses, der nur das Objekt A beinhaltet, sieht so aus

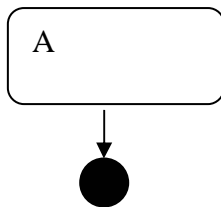


Will man den Endpunkt eines Prozesses definieren, so sehen die beiden Darstellungsarten wie im Folgenden angeben aus:

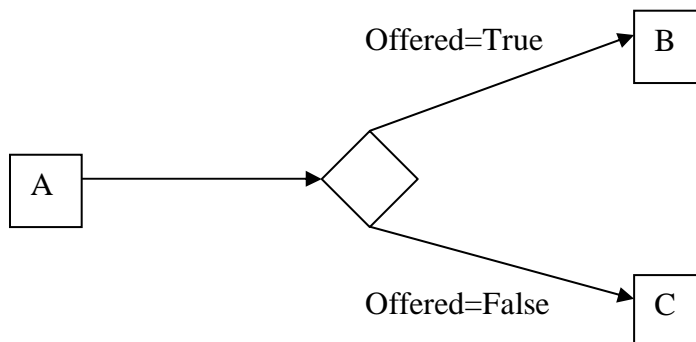
$-(2) \in \mathbf{Point}$

$A \text{ predOf } -(2)$

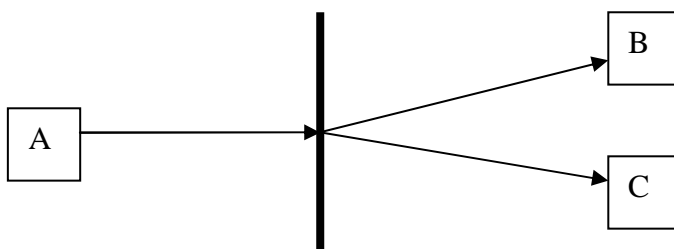
bzw.



29) Die Darstellung von Alternativen und AND-Verknüpfungen erfolgt mit bekannten Symbolen:  
Die Alternative:



und die AND-Verknüpfung:



Die folgenden beiden Relationen geben diese Sachverhalte wieder.

Die Alternative:

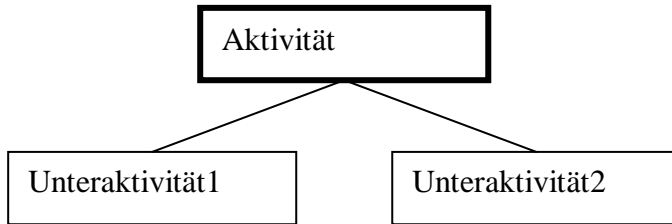
`OrStatement (A, Offered, B, C)`

Offered ist hierbei vom Datentyp Boolean und besitzt keine eigenes Darstellungssymbol (außer „am entsprechenden Pfeil stehen“). Sollte das „oder“-Symbol allein stehend bzw. nicht im Rahmen einer solchen Alternative benötigt werden, so kann der ADT **Or** verwendet werden.

Die AND-Verknüpfung:

`AndStatement ({A},{B,C})`

30) Die explizite Darstellung von strukturierten Aktivitäten (Seite 287) ist hier zu erkennen:



Im Prinzip stellt dies eine Verfeinerung von 23) dar.

Die dazugehörige Relation ist `parentOf`, wobei folgende Schreibweise gilt:

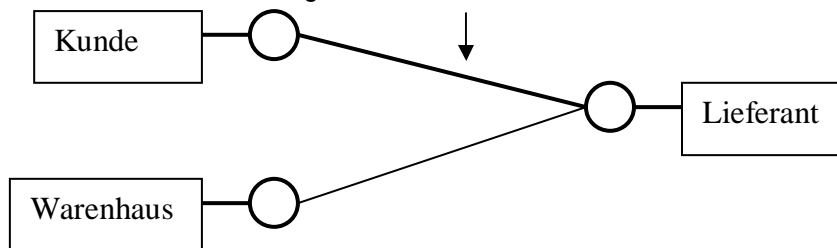
`Aktivität parentOf {Unteraktivität1, Unteraktivität2}`

31) Bestehende Partnerschaften (Seite 290) drückt die Relation `rel` aus.

So gibt

`Kunde rel Lieferant`

z.B. eine in dieser Grafik dargestellte Partnerschaft wieder:

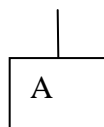


32) Zur besonderen Kennzeichnung eines Fehlers (Seite 291) dient dieses durch den ADT **Error** repräsentierte Symbol:

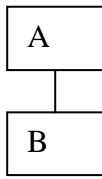


33) Die Darstellung von Zeiträume vor, zwischen und nach z.B. Aktivitäten (Seite 23) wird im Folgenden dargelegt, wobei jeweils zuerst die entsprechende Relation und dann die dazugehörige graphische Darstellung angegeben wird:

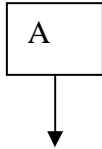
1) `period_pred (A)`



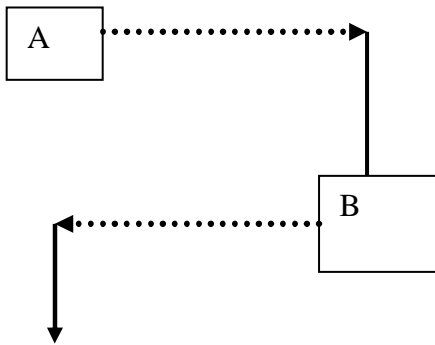
2) `period_betw (A, B)`



3) `period_succ (A)`



34) Wird eine Aktivität durch eine anderen Aktivität (Seite 23) blockiert, so stellt dies die Relation `blockedBy` dar:  
- A `blockedBy` B



#### **Anmerkungen zu einigen Grafiken:**

- 1) Seite 14 –Die Darstellung sollte aus Gründen der Übersichtlichkeit auf zwei Grafiken aufgeteilt werden
- 2) Seite 37 – „IDL“ und „language specific call interface“ werden wie Objekte behandelt
- 3) Befinden sich in einer Grafik nummerierte Zeichenketten, so symbolisieren diese eine Reihenfolge (Seite 40) und können bzw. sollten (aus Gründen der Konsistenz)entsprechend dem 8. Punkt notiert werden. Außerdem sollten Teilkomponenten immer entsprechend der `partOf` Relation dargestellt werden. An einigen stellen im Buch wird dies jedoch so gehandhabt, dass in einer Komponente mehrere Bezeichnungen auftreten und die unterstrichene den Namen der Komponente symbolisiert während die restlichen Bezeichnungen für Teilkomponenten stehen (Seite 38).
- 4) Seiten 38 und 40– Aktivitäten/Funktionen der Objekte sollten generell auch als solche dargestellt werden (siehe 3)).
- 5) Seite 51 – „client application“ und „client“ sollten mit dem Interface verbunden sein und dieses entsprechend der Interface Struktur (siehe 6) in der obigen Auflistung) dargestellt werden.
- 6) Seite 80 –alle verbal beschriebenen Aktivitäten etc. werden im Folgenden direkt in die Grafik mit aufgenommen
- 7) Seite 185 - von „businessEntity“ nach „UDDI“ sollte ein Pfeil sein??

Nachdem nun alle Strukturelemente durch `Relationen` bzw. **ADTs** beschrieben wurden, geht es in den folgenden Kapiteln um die Darstellung der Grafiken (aus dem genannten Buch von G.Alonso) mit Hilfe eben dieser `Relationen` und **ADTs**. Während viele Grafiken nur Strukturen beschreiben und keinerlei Dynamik enthalten und dementsprechend die Darstellung in Form einer ASM nicht sinnvoll ist, werden alle Grafiken, die dynamisches Verhalten beschreiben, neben der statischen Darstellung (`Relationen` und **ADTs**) auch in Form einer ASM beschrieben. In Bezug auf die Notation ist noch anzumerken, dass anstatt des `par` Operators (siehe [GUR99] und [WR]) alle parallel auszuführenden Regeln untereinander geschrieben werden.

# Kapitel 1

## Verschiedene Schichten eines Informationssystems (Grafik Seite 4/9):

PresentationLayer ∈ **Activity**  
ApplicationLayer ∈ **Activity**  
ResourceManagementLayer ∈ **Activity**  
Client ∈ **Object**  
InformationSystem ∈ **Container**  
(1tier ∈ **Activity**) → Seite 11

### **Relationen:**

PresentationLayer partOf InformationSystem  
ApplicationLayer partOf InformationSystem  
ResourceManagementLayer partOf InformationSystem

Client con PresentationLayer  
PresentationLayer con ApplicationLayer  
ApplicationLayer con ResourceManagementLayer

## Zusammenfassung von Schichten in der 1-tier Architektur (Grafik Seite 11):

PresentationLayer ∈ **Activity**  
ApplicationLayer ∈ **Activity**  
ResourceManagementLayer ∈ **Activity**  
Client ∈ **Object**  
InformationSystem ∈ **Container**  
1tier ∈ **structActivity**

### **Relationen:**

PresentationLayer partOf 1tier  
ApplicationLayer partOf 1tier  
ResourceManagementLayer partOf 1tier  
1tier partOf InformationSystem

Client con PresentationLayer  
PresentationLayer con ApplicationLayer  
ApplicationLayer con ResourceManagementLayer

## Zusammenfassung von Schichten in der 2-tier Architektur (Grafik Seite 12):

PresentationLayer ∈ **Activity**  
ApplicationLayer ∈ **Activity**  
ResourceManagementLayer ∈ **Activity**  
Server ∈ **structActivity**  
Client ∈ **structObject**  
InformationSystem ∈ **Container**

**Relationen:**

PresentationLayer partOf Client

ApplicationLayer partOf Server

ResourceManagementLayer partOf Server

Server partOf InformationSystem

PresentationLayer partOf InformationSystem

Client con Server

ApplicationLayer con ResourceManagementLayer

**Interne Organisation der Anwendungslogik-Schicht (Grafik Seite 14):**

Server  $\in$  **structActivity**

ResourceManagementLayer  $\in$  **Activity**

serverAPI  $\in$  **Activity**

ServiceInterface  $\in$  **Activity**

Service  $\in$  **Object**

**Relationen:**

numberOf ServiceInterface

numberOf Service

ResourceManagementLayer partOf Server

serverAPI interfaceOf Server

ServiceInterface interfaceOf serverAPI

Service partOf serverAPI

ServiceInterface comm Service

serverAPI con ResourceManagementLayer

**Clients als Integrationswerkzeug (Grafik Seite 15):**

Die im Folgenden in Klammern stehenden Zahlen hinter einem Bezeichner dienen nur zur formalen Unterscheidung von Objekten mit gleicher Bezeichnung, in der entsprechenden grafischen Darstellung sind sie nicht mit anzugeben.

ApplicationLogic  $\in$  **Activity**

PresentationLayer1  $\in$  **Activity**

PresentationLayer2  $\in$  **Activity**

ApplicationLogicLayer(1)  $\in$  **Activity**

ApplicationLogicLayer(2)  $\in$  **Activity**

ResourceManagementLayer(1)  $\in$  **Activity**

ResourceManagementLayer(2)  $\in$  **Activity**

Server1  $\in$  **structActivity**

Server2  $\in$  **structActivity**

Client  $\in$  **structObject**

**Relationen:**

ApplicationLogic partOf Client

PresentationLayer1 partOf Client

PresentationLayer2 partOf Client

ApplicationLogicLayer(1) partOf Server1  
ResourceManagementLayer(1) partOf Server1

ApplicationLogicLayer(2) partOf Server2  
ResourceManagementLayer(2) partOf Server2

PresentationLayer1 con Server1  
ApplicationLogicLayer(1) con ResourceManagementLayer(1)  
PresentationLayer2 con Server2  
ApplicationLogicLayer(2) con ResourceManagementLayer(2)

### **3-tier Architekturen und Middleware (Grafik Seite 16):**

Client ∈ **structObject**  
PresentationLayer ∈ **Activity**  
Middleware ∈ **structActivity**  
ApplicationLogicLayer ∈ **Activity**  
ResourceManagementLayer ∈ **Activity**  
InformationSystem ∈ **Container**

#### **Relationen:**

PresentationLayer partOf Client  
  
ApplicationLogicLayer partOf Middleware  
  
Middleware partOf InformationSystem  
ResourceManagementLayer partOf InformationSystem  
PresentationLayer partOf InformationSystem

### **Integration von Systemen mit unterschiedlichen Architekturen (Grafik Seite 17):**

Client(1) ∈ **structObject**  
PresentationLayer ∈ **Activity**  
  
Middleware ∈ **structActivity**  
ApplicationLogicLayer ∈ **Activity**  
IntegrationLogic ∈ **Activity**  
Client(2) ∈ **Object**  
Client(3) ∈ **Object**  
ResourceManagementLayer ∈ **structActivity**  
Wrapper(1) ∈ **Activity**  
Wrapper(2) ∈ **Activity**  
Wrapper(3) ∈ **Activity**  
1tier ∈ **structActivity**  
2tier ∈ **structActivity**  
3tier ∈ **structActivity**  
-(1) ∈ **Resource**  
-(2) ∈ **Activity**  
-(3) ∈ **Activity**  
-(4) ∈ **structActivity**  
-(5) ∈ **Activity**

-(6) ∈ **structActivity**  
-(7) ∈ **structActivity**  
-(8) ∈ **structActivity**  
-(9) ∈ **Resource**  
-(10) ∈ **Activity**  
-(11) ∈ **Activity**

**Relationen:**

PresentationLayer partOf Client(1)

ApplicationLogicLayer partOf Middleware  
{IntegrationLogic, Client(2), Client(3)} partOf ApplicationLogicLayer

{Wrapper(1), Wrapper(2), Wrapper(3)} partOf ResourceManagementLayer  
{1tier, 2tier, 3tier} partOf ResourceManagementLayer

-(1) partOf 1tier  
-(2) partOf 2tier  
-(3) partOf 2tier  
-(5) partOf -(4)  
-(4) partOf 3tier  
-(9) partOf -(7)  
-(10) partOf -(8)  
-(11) partOf -(8)  
-(7) partOf -(6)  
-(8) partOf -(6)  
-(6) partOf 3tier

Client(1) con Middleware  
Middleware con ResourceManagementLayer  
ApplicationLogicLayer comm Wrapper(1)  
ApplicationLogicLayer comm Wrapper(2)  
ApplicationLogicLayer comm Wrapper(3)  
Client(2) comm 2tier  
-(2) con -(3)  
Client(3) comm 3tier  
Wrapper(1) comm 1tier  
Wrapper(2) comm 2tier  
Wrapper(3) comm -(4)  
-(4) con -(6)  
-(10) con -(11)

**Erweiterung eines 3-tier zu einem n-tier System (Grafik Seite 20):**

Client ∈ **structObject**  
WebBrowser ∈ **Object**  
InformationSystem ∈ **Container**  
PresentationLayer ∈ **structActivity**  
WebServer ∈ **Object**  
HTMLFilter ∈ **Object**  
Middleware ∈ **structActivity**  
ApplicationLogicLayer ∈ **Activity**  
ResourceManagementLayer ∈ **Activity**

**Relationen:**

WebBrowser partOf Client

WebServer partOf PresentationLayer

HTMLFilter partOf PresentationLayer

ApplicationLogicLayer partOf Middleware

PresentationLayer partOf InformationSystem

Middleware partOf InformationSystem

ResourceManagementLayer partOf InformationSystem

WebBrowser con WebServer

PresentationLayer con Middleware

Middleware con ResourceManagementLayer

WebServer comm HTMLFilter

**Prozessunterbrechung auf Grund eines synchronen Aufrufes (Grafik Seite 23):**

Request  $\in$  **Activity**

Response  $\in$  **Activity**

InvokingExecutionThread  $\in$  **Container**

InvokedExecutionThread  $\in$  **Container**

**Relationen:**

Request partOf InvokingExecutionThread

Response partOf InvokedExecutionThread

period\_pred (Request)

Request blockedBy Response

**Asynchrone Aufrufe und Queues (Grafik Seite 25):**

put(1)  $\in$  **Activity**

fetch(1)  $\in$  **Activity**

put(2)  $\in$  **Activity**

fetch(2)  $\in$  **Activity**

queue(1)  $\in$  **Queue**

queue(2)  $\in$  **Queue**

InvokingExecutionThread  $\in$  **Container**

InvokedExecutionThread  $\in$  **Container**

**Relationen:**

period\_pred (put(1))

period\_betw (put(1), fetch(1))

pred\_succ (fetch(1))

period\_betw (put(2), fetch(2))

put(1) part queue(1)

queue(1) part fetch(2)

put(2) part queue(2)

queue(2) part fetch(1)

## Kapitel 2

### Abstraktion mit RPC (Grafik Seite 31):

RemoteProcedureCall  $\in$  **Activity**

sockets  $\in$  **Activity**

TCP,UDP  $\in$  **Activity**

InternetProtocol(IP)  $\in$  **Activity**

#### **Relationen:**

RemoteProcedureCall  $\text{con}$  sockets

sockets  $\text{con}$  TCP,UDP

TCP,UDP  $\text{con}$  InternetProtocol(IP)

### Entwicklung verteilter Anwendungen mit RPC (Grafik Seite 37):

ClientProcess  $\in$  **Activity**

ClientCode  $\in$  **Object**

ClientStub  $\in$  **Object**

DevelopmentEnvironment  $\in$  **Activity**

IDLSources  $\in$  **Resource**

IDLCompiler  $\in$  **Activity**

InterfaceHeaders  $\in$  **Resource**

ServerProcess  $\in$  **Activity**

ServerCode  $\in$  **Object**

ServerStub  $\in$  **Object**

IDL  $\in$  **Object**

LanguageSpecificCallInterface(1)  $\in$  **Object**

LanguageSpecificCallInterface(1)  $\in$  **Object**

#### **Relationen:**

ClientCode  $\text{partOf}$  ClientProcess

ClientStub  $\text{partOf}$  ClientProcess

LanguageSpecificCallInterface(1)  $\text{partOf}$  ClientProcess

LanguageSpecificCallInterface(1)  $\text{elementOf}$  ClientCode

ClientStub  $\text{equal}$  LanguageSpecificCallInterface(1)

IDL  $\text{partOf}$  DevelopmentEnvironment

IDLSources  $\text{partOf}$  DevelopmentEnvironment

IDLcompiler  $\text{partOf}$  DevelopmentEnvironment

InterfaceHeaders  $\text{partOf}$  DevelopmentEnvironment

ServerCode  $\text{partOf}$  ServerProcess

ServerStub  $\text{partOf}$  ServerProcess

LanguageSpecificCallInterface(2)  $\text{partOf}$  ServerProcess

LanguageSpecificCallInterface(2)  $\text{elementOf}$  ServerCode

ServerStub  $\text{equal}$  LanguageSpecificCallInterface(2)

IDL  $\text{predOf}$  IDLSources

IDLSources  $\text{predOf}$  IDLCompiler

IDLCompiler  $\text{predOf}$  InterfaceHeaders

IDLCompiler  $\text{allows}$  ClientStub

IDLCompiler  $\text{allows}$  ServerStub

InterfaceHeaders allows ClientCode  
InterfaceHeaders allows ServerCode

**RPC- Grundfunktionen (Grafik Seite 38):**

Rotmarkierte Elemente werden in der nächsten Grafik ausgetauscht

ClientProcess ∈ **Activity**  
ServerProcess ∈ **Activity**  
Client ∈ **Object**  
ProcedureCall ∈ **Activity**  
ClientStub ∈ **Object**  
Bind ∈ **Activity**  
Marshal ∈ **Activity**  
Serialize ∈ **Activity**  
Send ∈ **Activity**  
CommunicationModule(1) ∈ **Activity**  
Server ∈ **Object**  
Procedure ∈ **Activity**  
Dispatcher(SelectStub) ∈ **Activity**  
ServerStub ∈ **Object**  
Unmarshal ∈ **Activity**  
Deserialize ∈ **Activity**  
Receive ∈ **Activity**  
CommunicationModule(2) ∈ **Activity**

**Relationen:**

Client partOf ClientProcess  
ProcedureCall partOf Client

ClientStub partOf ClientProcess  
Bind partOf ClientStub  
Marshal partOf ClientStub  
Serialize partOf ClientStub  
Send partOf ClientStub  
CommunicationModule(1) partOf ClientProcess

ProcedureCall predOf ClientStub  
Send predOf CommunicationModule(1)

Server partOf ServerProcess  
Procedure partOf Server

ServerStub partOf ServerProcess  
Unmarshal partOf ServerStub  
Deserialize partOf ServerStub  
Receive partOf ServerStub

Dispatcher(SelectStub) partOf ServerProcess  
CommunicationModule(2) partOf ServerProcess

Unmarshal predOf Procedure  
CommunicationModule(1) predOf CommunicationModule(2)  
CommunicationModule(2) predOf Dispatcher(SelectStub)

Dispatcher(SelectStub) predOf Receive

### **Erweiterung der Grundfunktionen um Dynamic Binding (Grafik Seite 40):**

2.find ∈ **Activity**  
5.Send ∈ **Activity**  
0.register ∈ **Activity**  
7.receive ∈ **Activity**  
3.QueryForServerImplementingTheProcedure ∈ **Activity**  
NameAndDirectoryService(Binder) ∈ **Activity**  
4.SendAddressOfServer ∈ **Activity**  
6.InvokeProcedure ∈ **Activity**  
1.RegisterServerAndProcedure ∈ **Activity**

#### **Relationen:**

2.find comm CommunicationModule(1)  
5.send predOf CommunicationModule(1)  
3.QueryForServerImplementingTheProcedure partOf CommunicationModule(1)  
3.QueryForServerImplementingTheProcedure predOf NameAndDirectoryService(Binder)  
6.InvokeProcedure partOf CommunicationModule(1)  
6.InvokeProcedure partOf CommunicationModule(2)

4.SendAddressOfServer partOf NameAndDirectoryService(Binder)  
4.SendAddressOfServer predOf CommunicationModule(1)

Dispatcher(SelectStub) predOf 7.receive  
0.register predOf CommunicationModule(2)

1.RegisterServerAndProcedure partOf CommunicationModule(2)  
1.RegisterServerAndProcedure predOf NameAndDirectoryService(Binder)

Auch wenn die Nummerierung der Aktivitäten schon deren Reihenfolge darstellt, so kann man diese zusätzlich auch wie folgt formulieren:

0.register allows 1.RegisterServerAndProcedure  
1.RegisterServerAndProcedure allows 2.find  
2.find allows 3.QueryForServerImplementingTheProcedure  
3.QueryForServerImplementingTheProcedure allows 4.SendAddressOfServer  
4.SendAddressOfServer allows 5.send  
5.send allows 6.InvokeProcedure  
6.InvokeProcedure allows 7.receive

### **DCE- Architektur (Grafik Seite 44):**

(Erweiterung von Seite 37)

Die Aktivität “DevelopmentEnvironment” der Grafik auf Seite 37 wird hier in “DCEDevelopmentEnvironment” umbenannt. Alle anderen Definitionen bleiben bestehen und werden um die Folgenden ergänzt.

RPC\_API(1) ∈ **Object**  
RPCRuntimeServiceLibrary(1) ∈ **Object**  
RPC\_API(2) ∈ **Object**  
RPCRuntimeServiceLibrary(2) ∈ **Object**

DCERuntimeEnvironment ∈ **Activity**  
RPCProtocols ∈ **Activity**  
SecurityService ∈ **Activity**  
CellService ∈ **Activity**  
DistributedFileService ∈ **Activity**  
ThreadService ∈ **Activity**

**Relationen:**

RPC\_API(1) partOf ClientProcess  
RPCRuntimeServiceLibrary(1) partOf ClientProcess  
RPC\_API(1) elementOf ClientProcess  
RPCRuntimeServiceLibrary(1) equal RPC\_API(1)  
RPCRuntimeServiceLibrary(1) comm DCERuntimeEnvironment

RPC\_API(2) partOf ClientProcess  
RPCRuntimeServiceLibrary(2) partOf ClientProcess  
RPC\_API(2) elementOf ClientProcess  
RPCRuntimeServiceLibrary(2) equal RPC\_API(2)  
RPCRuntimeServiceLibrary(2) comm DCERuntimeEnvironment

**Resourcenverwaltung mit DBMS (Grafik Seite 48a):**

Client ∈ **Object**  
ResourceManager(DBMS) ∈ **Activity**  
Database ∈ **Resource**

**Relationen:**

Client predOf ResourceManager(DBMS)  
ResourceManager(DBMS) predOf Database

**Verwaltung heterogener Resourcen mit DBMS (Grafik Seite 48b):**

Client ∈ **Object**  
ResourceManager1 ∈ **Activity**  
ResourceManager2 ∈ **Activity**  
Database1 ∈ **Resource**  
Database2 ∈ **Resource**  
Withdraw ∈ **Activity**  
Deposit ∈ **Activity**  
SourceAccount ∈ **Container**  
DestinationAccount ∈ **Container**

**Relationen:**

Withdraw partOf Client  
Deposit partOf Client  
Withdraw predOf ResourceManager1  
ResourceManager1 predOf Database1  
Database1 comm SourceAccount

Deposit predOf ResourceManager2  
ResourceManager2 predOf Database2  
Database2 comm DestinationAccount

### **Kapselung von RPC-Aufrufen (Grafik Seite 49)\*:**

Die Reihenfolge der beschriebenen Aktivitäten wird hier durch zusätzliche Relationen beschrieben – dies wurde in der graphischen Darstellung nicht gemacht.

#### 1) statisch:

ClientProcess ∈ **Activity**  
ServerProcess ∈ **Activity**  
TransactionManager ∈ **Activity**  
Client ∈ **Object**  
ClientStub ∈ **Object**  
Server ∈ **Object**  
ServerStub ∈ **Object**  
-(1) ∈ **Object**  
1.BOT ∈ **Activity**  
2.RegisterTxn&CreateContext ∈ **Activity**  
3.CreateTxnId ∈ **Activity**  
4.ProcedureCall ∈ **Activity**  
5.AddTxnId&ContextToCall ∈ **Activity**  
6.ExtractContextAndTxnId ∈ **Activity**  
7.RegisterServerForTxn ∈ **Activity**  
8.LookupTxnId ∈ **Activity**  
9.ProcedureCall ∈ **Activity**  
10.EOT ∈ **Activity**  
11.RequestCommit ∈ **Activity**  
12.LookupTxnId ∈ **Activity**  
13.ParticipateIn2PC ∈ **Activity**  
14.ConfirmTermination ∈ **Activity**  
RegisterTxn ∈ **Activity**  
RegisterClientForTxn ∈ **Activity**  
ReturnTxnId ∈ **Activity**  
RegisterServerForTxn ∈ **Activity**  
Run2PC ∈ **Activity**  
NotifyClientOfOutcome ∈ **Activity**

#### **Relationen:**

Client partOf ClientProcess

1.BOT partOf Client  
4.ProcedureCall partOf Client  
10.EOT partOf Client

ClientStub partOf ClientProcess

2.RegisterTxn&CreateContext partOf ClientStub  
5.AddTxnId&ContextToCall partOf ClientStub  
11.RequestCommit partOf ClientStub  
14.ConfirmTermination partOf ClientStub

Server partOf ServerProcess

9.Procedure partOf Server

ServerStub partOf ServerProcess

6.ExtractContextAndTxnId partOf ServerStub  
7.RegisterServerForTxn partOf ServerStub  
13.ParticipateIn2PC partOf ServerStub

-(1) partOf TransactionManager

3.CreateTxnId partOf -(1)

RegisterTxn partOf -(1)

RegisterClientForTxn partOf -(1)

ReturnTxnId partOf -(1)

8.LookupTxnId partOf -(1)

RegisterServerForTxn partOf -(1)

12.LookupTxnId partOf -(1)

Run2PC partOf -(1)

NotifyClientOfOutcome partOf -(1)

1.BOT predOf 2.RegisterTxn&CreateContext

2.RegisterTxn&CreateContext predOf 3.CreateTxnId

3.CreateTxnId predOf RegisterTxn

RegisterTxn predOf RegisterClientForTxn

RegisterClientForTxn predOf ReturnTxnId

ReturnTxnId predOf 4.ProcedureCall

4.ProcedureCall predOf 5.AddTxnId&ContextToCall

5.AddTxnId&ContextToCall predOf 6.ExtractContextAndTxnId

6.ExtractContextAndTxnId predOf 7.RegisterServerForTxn

7.RegisterServerForTxn predOf 8.LookupTxnId

7.RegisterServerForTxn predOf 9.Procedure

8.LookupTxnId predOf RegisterServerForTxn

RegisterServerForTxn allows 9.Procedure

9.Procedure predOf 10.EOT

10.EOT predOf 11.RequestCommit

11.RequestCommit predOf 12.LookupTxnId

12.LookupTxnId predOf Run2PC

Run2PC predOf 13.ParticipateIn2PC

Run2PC predOf NotifyClientOfOutcome

13.ParticipateIn2PC allows NotifyClientOfOutcome

NotifyClientOfOutcome predOf 14.ConfirmTermination

14.ConfirmTermination predOf 10.EOT

## 2) dynamisch:

Variablen:

clnt  $\in$  Clients

svr  $\in$  Servers

mng  $\in$  Managers

Funktionen:

BeginRPCCall: Clients  $\rightarrow$  Boolean

BOT: Clients  $\rightarrow$  {BOTCall}

ProcedureCall: Clients  $\rightarrow$  {ProcedureCalled}

RegisterTXN&CreateContext: Clients  $\rightarrow$  {CreateContext}

ReturnTXNId: Clients  $\rightarrow$  {TXNIdCreated}

AddTXNId&ContextToCall: Clients  $\rightarrow$  {TXNId&ContextAdded}

RegisterServerTXN: Servers  $\rightarrow$  {ServerRegistered}

RegisterServerForTXN: Servers  $\rightarrow$  {RegisterServer}

ExtractContext&TXNId: Clients  $\rightarrow$  {Context&TXNIdExtracted}

Procedure: Servers  $\rightarrow$  {Proceeding}

EOT: Clients  $\rightarrow$  {EOTCall}

RequestCommit: Clients  $\rightarrow$  {CommitRequested}

MyServer: Clients  $\rightarrow$  Servers  
 ClientState: Clients  $\rightarrow$  {BOTCall, ProcedureCalled, TerminationConfirmed,  
                                   CreateContext, TXNId&ContextAdded, EOTCall, CommitRequested,  
                                   ClientNotified}  
 MngrState: Clients+Servers  $\rightarrow$  {TXNIdCreated, ServerRegistered, running2PC}  
 ServerState: Servers  $\rightarrow$  {Context&TXNIdExtracted, RegisterServer, Proceeding,  
                                   ParticipatingIn2PC}  
 Run2PC: Servers  $\rightarrow$  {running2PC}  
 ParticipateIn2PC: Servers  $\rightarrow$  {ParticipatingIn2PC}  
 NotifyClientOfOutcome: Clients  $\rightarrow$  {ClientNotified}  
 ConfirmTermination: Clients  $\rightarrow$  {TerminationConfirmed}

ASM:

EncapsulatedRPCCall =<sub>def</sub>  
     ClientProcess(clnt)  
     ServerProcess(srvr)  
     TransactionManager(mngr)

ClientProcess(Self) =<sub>def</sub>  
     Client(Self)  
     ClientStub(Self)

Client(Self) =<sub>def</sub>  
     **If** BeginRPCCall(Self)  
         ClientState(Self) := BOT(Self)  
     **If** MngrState(Self) = TXNIdCreated  
         ClientState(Self) := ProcedureCall(Self)  
     **If** ClientState(Self) = TerminationConfirmed  
     **or** ServerState(MyServer(Self)) = Proceeding  
         ClientState(Self) := EOT(Self)

ClientStub(Self) =<sub>def</sub>  
     **If** ClientState(Self) = BOTCall  
         ClientState(Self) := RegisterTXN&CreateContext(Self)  
     **If** ClientState(Self) = ProcedureCalled  
         ClientState(Self) := AddTXNId&ContextToCall(Self)  
     **If** ClientState(Self) = EOTCall  
         ClientState(Self) := RequestCommit(Self)  
     **If** ClientState(Self) = ClientNotified  
         ClientState(Self) := ConfirmTermination(Self)

ServerProcess(Self) =<sub>def</sub>  
     Server(Self)  
     ServerStub(Self)

Server(Self) =<sub>def</sub>  
     **If** MngrState(Self) = ServerRegistered  
         ServerState(Self) := Procedure(Self)

ServerStub(Self) =<sub>def</sub>  
     **For all** client  $\in$  Clients  
         **If** ClientState(client) = TXNId&ContextAdded  
             ServerState(Self) := ExtractContext&TXNId(client)  
             MyServer(client) := Self  
     **If** ServerState(Self) = Context&TXNIdExtracted  
         ServerState(Self) := RegisterServerForTXN(Self)

```

If MngrState(Self) = running2PC
    ServerState(Self) := ParticipateIn2PC(Self)

TransactionManager(Self) =def
    CreateTXN(Self)
    LookupTXN(Self)

CreateTXN(Self) =def
    For all client ∈ Clients
        If ClientState(client) = CreateContext
            CreateTXNid(client)
            RegisterTXN(client)
            RegisterClientForTXN(client)
            MngrState(client) := ReturnTXNid(client)

LookupTXN(Self) =def
    For all server ∈ Servers
        If ServerState(server) = RegisterServer
            LookupTXNid(server)
            MngrState(server) := RegisterServerTXN(server)
    For all client ∈ Clients
        If ClientState(client) = CommitRequested
            and ServerState(MyServer(client)) != ParticipatingIn2PC
                LookupTXNid(MyServer(client))
                MngrState(MyServer(client)) := Run2PC(MyServer(client))
        If ServerState(MyServer(client)) = ParticipatingIn2PC
            MngrState(client) := NotifyClientOfOutcome(client)

```

**Grundbestandteile eines TP-Monitors (Grafik Seite 51):**

ClientApplication ∈ **Object**  
 User ∈ **Object**  
 TPMonitor ∈ **Activity**  
 TransactionManagar ∈ **Activity**  
 Interface(API,Presentation,Authentication) ∈ **Activity**  
 ProgramFlow ∈ **Activity**  
 Router ∈ **Activity**  
 CommunicationManager ∈ **Activity**  
 RegisteredPrograms ∈ **Resource**  
 Resources ∈ **Resource**  
 TPServices ∈ **Activity**  
 Wrapper(1) ∈ **Object**  
 Resource(1) ∈ **Activity**  
 Wrapper(2) ∈ **Object**  
 Resource(2) ∈ **Activity**  
 Wrapper(3) ∈ **Object**  
 Resource(3) ∈ **Activity**

**Relationen:**

TransactionManagar partOf TPMonitor  
 Interface(API,Presentation,Authentication) partOf TPMonitor  
 ProgramFlow partOf TPMonitor  
 Router partOf TPMonitor  
 CommunicationManager partOf TPMonitor

RegisteredPrograms partOf TPMonitor  
Resources partOf TPMonitor  
TPServices partOf TPMonitor

ClientApplication comm TPMonitor  
User comm TPMonitor

Interface(API,Presentation,Authentication) comm ProgramFlow  
ProgramFlow comm RegisteredPrograms  
ProgramFlow comm Router  
Router comm Resources  
Router comm CommunicationManager  
CommunicationManager comm Wrapper(1)  
CommunicationManager comm Wrapper(2)  
CommunicationManager comm Wrapper(3)  
Wrapper(1) comm Resource(1)  
Wrapper(2) comm Resource(2)  
Wrapper(3) comm Resource(3)

**High-Level Sichtweise auf die CORBA-Architektur (Grafik Seite 55):**

User-DefinedObjects ∈ **Object**  
CORBAFacilities ∈ **Activity**  
VerticalFacilities ∈ **Container** → anders als in der Grafik  
Financials ∈ **Object**  
SupplyChain ∈ **Object**  
-(1) ∈ **undefObj**  
HorizontalFacilities ∈ **Container** → anders als in der Grafik  
DistributedDocuments ∈ **Object**  
InformationManagement ∈ **Object**  
SystemsManagement ∈ **Object**  
TaskManagement ∈ **Object**  
ObjectRequestBroker ∈ **Activity**  
CORBAServices ∈ **Activity**  
Naming ∈ **Object**  
Trader ∈ **Object**  
Transactions ∈ **Object**  
Concurrency ∈ **Object**  
Events ∈ **Object**  
Query ∈ **Object**  
Lifecycle ∈ **Object**  
Security ∈ **Object**  
Properties ∈ **Object**  
Collection ∈ **Object**  
Relationships ∈ **Object**  
Externalization ∈ **Object**  
Time ∈ **Object**  
Startup ∈ **Object**  
Licensing ∈ **Object**  
Persistence ∈ **Object**

**Relationen:**

numberOf User-DefinedObjects  
 User-DefinedObjects comm ObjectRequestBroker

{Financials, SupplyChain, -(1)} partOf VerticalFacilities  
 VerticalFacilities partOf CORBAFacilities  
 {DistributedDocuments, InformationManagement} partOf HorizontalFacilities  
 {SystemsManagement, TaskManagement} partOf HorizontalFacilities  
 HorizontalFacilities partOf CORBAFacilities  
 CORBAFacilities comm ObjectRequestBroker

ObjectRequestBroker comm CORBAServices

{Naming, Trader, Transactions, Concurrency, Events, Query} partOf CORBAServices  
 {Lifecycle, Security, Properties, Collection} partOf CORBAServices  
 {Relationships, Externalization, Time, Startup} partOf CORBAServices  
 {Licensing, Persistence} partOf CORBAServices

**Rolle von IDL-Spezifikationen bei Clients und Servern (Grafik Seite 56):**

IDLofServiceProvider ∈ **Specification**  
 IDLCompiler(ClientSide) ∈ **Object**  
 IDLCompiler(ServerSide) ∈ **Object**  
 ApplicationObject(Client)(1) ∈ **Object**  
 ApplicationObject(Client)(2) ∈ **Object**  
 Stub ∈ **Object**  
 ApplicationObject(ServiceProvider) ∈ **Object**  
 Skeleton ∈ **Object**  
 DynamicInvocationInterface ∈ **Activity**  
 ObjectRequestBroker ∈ **Activity**  
 InterfaceRepository ∈ **Resource**

**Relationen:**

IDLofServiceProvider allows IDLCompiler(Client)  
 IDLofServiceProvider allows IDLCompiler(ServerSide)

IDLCompiler(Client) allows Stub  
 ApplicationObject(Client)(2) equal Stub  
 Stub comm ObjectRequestBroker

IDLCompiler(ServerSide) allows Skeleton  
 ApplicationObject(ServiceProvider) equal Skeleton  
 Skeleton comm ObjectRequestBroker

ApplicationObject(Client)(1) comm DynamicInvocationInterface  
 DynamicInvocationInterface interfaceOf ObjectRequestBroker

ObjectRequestBroker comm InterfaceRepository

**Nachrichtenbasierte Interaktionen (Grafik Seite 61a):**

ClientApplication  $\in$  **Object**

QuotationTool  $\in$  **Object**

Message-OrientedMiddleware(MOM)  $\in$  **Activity**

**Relationen:**

sendVia (ClientApplication, Message-OrientedMiddleware(MOM), QuotationTool)

**Nachrichtenbasierte Interaktionen (Grafik Seite 61b):**

ClientApplication  $\in$  **Object**

QuotationTool  $\in$  **Object**

Message-OrientedMiddleware(MOM)  $\in$  **Activity**

**Relationen:**

sendVia (QuotationTool, Message-OrientedMiddleware(MOM), ClientApplication)

**Message queuing model (Grafik Seite 62):**

ClientApplication  $\in$  **Object**

QuotationTool  $\in$  **Object**

Message-OrientedMiddleware(MOM)  $\in$  **Activity**

MOMCore  $\in$  **Container**

InboundQueue  $\in$  **Queue**

QueuedMessages  $\in$  **Queue**

**Relationen:**

InboundQueue partOf Message-OrientedMiddleware(MOM)

QueuedMessages partOf Message-OrientedMiddleware(MOM)

MOMCore partOf Message-OrientedMiddleware(MOM)

ClientApplication predOf MOMCore

QuotationTool predOf MOMCore

MOMCore predOf InboundQueue

InboundQueue predOf ClientApplication

MOMCore predOf QueuedMessages

QueuedMessages predOf QuotationTool

**Message queuing model with shared queues (Grafik Seite 63):**

QuotationTool1  $\in$  **Object**

QuotationTool2  $\in$  **Object**

QuotationTooln  $\in$  **Object**

MOMCore  $\in$  **Container**

InboundSharedQueue  $\in$  **Queue**

**Relationen:**

MOMCore predOf InboundSharedQueue

InboundSharedQueue[0] predOf QuotationTool1

InboundSharedQueue[1] predOf QuotationTool2

InboundSharedQueue[2] predOf QuotationTooln

## Kapitel 3

### **Nachrichtenbasierte Interaktion mit neuen Systemen (Grafik Seite 73):**

InventoryManagement ∈ **Object**  
ERP ∈ **Object**  
Dispatcher ∈ **Object**  
Shipping ∈ **Object**  
Month-EndClosing ∈ **Object**  
Message-OrientedMiddleware ∈ **Activity**

#### **Relationen:**

new Month-EndClosing  
sendVia (Dispatcher, Message-OrientedMiddleware, InventoryManagement)  
sendVia (Dispatcher, Message-OrientedMiddleware, ERP)  
sendVia (Dispatcher, Message-OrientedMiddleware, Shipping)  
sendVia (Dispatcher, Message-OrientedMiddleware, Month-EndClosing)

### **Message Broker und benutzerdefinierte Routing-Logik (Grafik Seite 74):**

Die hier dargestellte Struktur entspricht der des “message queueing models”, aus diesem Grund wird hier nur auf die andere Benennung der einzelnen Komponenten eingegangen und die vorgestellte Struktur (Grafik Seite 62) nicht noch einmal wiederholt, sondern in Form von “MessageQueueingModel” (beinhaltet ADT- und Relationen-Definitionen) übernommen. Die Umbenennung wird in der Form “alte Bezeichnung” → “neue Bezeichnung” vorgenommen. Diese Umbenennung erfolgt sowohl im ADT-Teil als auch im Relationen-teil des “MessageQueueingModel”.

ClientApplication → Sender  
QuotationTool → Receiver  
Message-OrientedMiddleware(MOM) → MessageBroker  
MOMCore → MessageBrokerCore  
InboundQueue → Queue(1)  
QueuedMessages → Queue(2)  
MessageQueueingModel

### **Message Broker und das “Publish/Subscribe”-Modell (Grafik Seite 76):**

InventoryManagement(Subscriber) ∈ **Object**  
ERP(Subscriber) ∈ **Object**  
Dispatcher(Publisher) ∈ **Object**  
Shipping(Subscriber) ∈ **Object**  
Month-EndClosing(Subscriber) ∈ **Object**  
MessageBroker ∈ **Activity**

#### **Relationen:**

new Month-EndClosing  
MessageBroker predOf InventoryManagement(Subscriber)  
MessageBroker predOf ERP(Subscriber)  
Dispatcher(Publisher) predOf MessageBroker

MessageBroker predOf Shipping(Subscriber)  
MessageBroker predOf Month-EndClosing(Subscriber)

**Verteilte Message-Broker (Grafik Seite 78 (3.5)):**

AdministrativeDomainA ∈ **Container**  
Admin(1) ∈ **Object**  
Client(1) ∈ **Object**  
Client(2) ∈ **Object**  
-(1) ∈ **undefObj**  
MessageBrokerMB-A ∈ **Activity**  
AdministrativeDomainB ∈ **Container**  
Admin(2) ∈ **Object**  
Client(3) ∈ **Object**  
Client(4) ∈ **Object**  
-(2) ∈ **undefObj**  
MessageBrokerMB-B ∈ **Activity**  
AdministrativeDomainC ∈ **Container**  
Admin(3) ∈ **Object**  
Client(5) ∈ **Object**  
Client(6) ∈ **Object**  
-(3) ∈ **undefObj**  
MessageBrokerMB-C ∈ **Activity**

**Relationen:**

Admin(1) partOf AdministrativeDomainA  
Client(1) partOf AdministrativeDomainA  
Client(2) partOf AdministrativeDomainA  
-(1) partOf AdministrativeDomainA  
MessageBrokerMB-A partOf AdministrativeDomainA

Admin(2) partOf AdministrativeDomainB  
Client(3) partOf AdministrativeDomainB  
Client(4) partOf AdministrativeDomainB  
-(2) partOf AdministrativeDomainB  
MessageBrokerMB-B partOf AdministrativeDomainB

Admin(3) partOf AdministrativeDomainC  
Client(5) partOf AdministrativeDomainC  
Client(6) partOf AdministrativeDomainC  
-(3) partOf AdministrativeDomainC  
MessageBrokerMB-C partOf AdministrativeDomainC

Admin(1) comm MessageBrokerMB-A  
Client(1) comm MessageBrokerMB-A  
Client(2) comm MessageBrokerMB-A  
-(1) comm MessageBrokerMB-A  
MessageBrokerMB-A comm MessageBrokerMB-B

Admin(2) comm MessageBrokerMB-B  
Client(3) comm MessageBrokerMB-B  
Client(4) comm MessageBrokerMB-B  
-(2) comm MessageBrokerMB-B

MessageBrokerMB-C comm MessageBrokerMB-C

Admin(3) comm MessageBrokerMB-C

Client(5) comm MessageBrokerMB-C

Client(6) comm MessageBrokerMB-C

-(3) comm MessageBrokerMB-C

**High-Level Architektur von EAI-Systemen (Grafik Seite 78 (3.6)):**

IntegrationApplication(ContainsTheCompositionLogic) ∈ **Object**

MessageBroker ∈ **Activity**

SmartQuotationAdapter ∈ **Object**

DatabaseAdapter ∈ **Object**

SmartForecastingAdapter ∈ **Object**

E-MailAdapter ∈ **Object**

XYZAdapter ∈ **Object**

SmartQuotation ∈ **Activity**

DBMSApplications ∈ **Resource**

SmartForecasting ∈ **Activity**

Client ∈ **Object**

XYZ ∈ **Activity**

MessageBrokerSystem ∈ **Container**

**Relationen:**

numberOf Client

IntegrationApplication(ContainsTheCompositionLogic) comm MessageBroker

MessageBroker comm SmartQuotationAdapter

MessageBroker comm DatabaseAdapter

MessageBroker comm SmartForecastingAdapter

MessageBroker comm E-MailAdapter

MessageBroker comm XYZAdapter

SmartQuotationAdapter comm SmartQuotation

DatabaseAdapter comm DBMSApplications

SmartForecastingAdapter comm SmartForecasting

E-MailAdapter comm Client

XYZAdapter comm XYZ

Erweiterung der Grafik für spätere Verwendung (Grafik Seite 90):

MessageBroker partOf MessageBrokerSystem

SmartQuotationAdapter partOf MessageBrokerSystem

DatabaseAdapter partOf MessageBrokerSystem

SmartForecastingAdapter partOf MessageBrokerSystem

E-MailAdapter partOf MessageBrokerSystem

XYZAdapter partOf MessageBrokerSystem

SmartQuotation partOf MessageBrokerSystem

DBMSApplications partOf MessageBrokerSystem

SmartForecasting partOf MessageBrokerSystem

Clients partOf MessageBrokerSystem

XYZ partOf MessageBrokerSystem

## **Nachrichtenaustausch und Prozessaufrufe (Grafik Seite 80)\*:**

1) statisch:

RFQProcessing ∈ **Object**

MessageBroker ∈ **Activity**

SmartQuotationAdapter ∈ **Object**

SmartQuotation ∈ **Activity**

SmartForecastingAdapter ∈ **Object**

SmartForecasting ∈ **Activity**

Folgende Aktivitäten werden im Buch nur verbal beschrieben und durch Nummerierung der Pfeile in die Grafik eingebunden –hier werden sie wie andere Aktivitäten dargestellt

SubscribeToMessageQuote ∈ **Activity**

SubscribeToMessageQuoteRequest ∈ **Activity**

SubscribeToMessageNewQuote ∈ **Activity**

PublishAQuoteRequestMessage ∈ **Activity**

DeliverAQuoteRequestMessage ∈ **Activity**

InvokeGetQuoteFunction ∈ **Activity**

PublishAQuoteMessage ∈ **Activity**

DeliverAQuoteMessage ∈ **Activity**

PublishANewQuoteMessage ∈ **Activity**

DeliverANewQuoteMessage ∈ **Activity**

InvokeCreateForecastEntryProcedure ∈ **Activity**

### **Relationen:**

SubscribeToMessageQuote partOf RFQProcessing

PublishAQuoteRequestMessage partOf RFQProcessing

PublishANewQuoteMessage partOf RFQProcessing

DeliverAQuoteRequestMessage partOf MessageBroker

DeliverAQuoteMessage partOf MessageBroker

DeliverANewQuoteMessage partOf MessageBroker

SubscribeToMessageQuoteRequest partOf SmartQuotationAdapter

InvokeGetQuoteFunction partOf SmartQuotationAdapter

PublishAQuoteMessage partOf SmartQuotationAdapter

SubscribeToMessageNewQuote partOf SmartForecastingAdapter

InvokeCreateForecastEntryProcedure partOf SmartForecastingAdapter

SubscribeToMessageQuote allows MessageBroker

SubscribeToMessageQuoteRequest allows MessageBroker

SubscribeToMessageNewQuote allows MessageBroker

SubscribeToMessageQuote allows PublishAQuoteRequestMessage

SubscribeToMessageQuoteRequest allows PublishAQuoteRequestMessage

SubscribeToMessageNewQuote allows PublishAQuoteRequestMessage

PublishAQuoteRequestMessage predOf DeliverAQuoteRequestMessage

DeliverAQuoteRequestMessage predOf InvokeGetQuoteFunction

InvokeGetQuoteFunction predOf SmartQuotation

SmartQuotation allows PublishAQuoteMessage

PublishAQuoteMessage predOf DeliverAQuoteMessage

DeliverAQuoteMessage predOf PublishANewQuoteMessage

PublishANewQuoteMessage predOf DeliverANewQuoteMessage

DeliverANewQuoteMessage predOf InvokeCreateForecastEntryProcedure

InvokeCreateForecastEntryProcedure predOf SmartForecasting

## 2) dynamisch:

benötigte Universen:

Messages ( beinhaltet alle Nachrichten, die während Interaktionen zwischen den Agenten auftreten können)

MessageTypes =<sub>def</sub> { Quote, newQuote, QuoteRequest }

SWAgents =<sub>def</sub> { SWA\_RFQProcessing, SWA\_SmartQuotationAdapter, SWA\_SmartForecastingAdapter }

Konstantensymbole zur Repräsentation der auftretenden Zustände (Erläuterung des Zustandes):

RequestPublicated (RFQProcessing published the RequestMsg)

RequestDelivered (the message broker delivered RequestMsg to SmartQuotationAdapter)

QuoteAvailable (result of GetQuote function available)

QuotePublicated (SmartQuotationAdapter published QuoteMsg)

QuoteDelivered (the message broker delivered QuoteMsg to RFQProcessing)

newQuoteAvailable (RFQProcessing created newQuoteMsg)

newQuotePublicated (RFQProcessing published newQuoteMsg)

newQuoteDelivered (the message broker delivered newQuoteMsg to SmartForecastingAdapter)

EntryAvailable (new forecast entry has been created)

Variablen:

Initialized ∈ Boolean

state ∈ {RequestPublicated, RequestDelivered, QuotePublicated, QuoteDelivered, QuoteAvailable, newQuoteAvailable, newQuotePublicated, newQuoteDelivered, EntryAvailable}

QuoteRequestMsg ∈ Messages (message type: QuoteRequest)

QuoteMsg ∈ Messages (message type: Quote)

newQuoteMsg ∈ Messages (message type: newQuote)

Msg ∈ Messages

swagent ∈ SWAgents

MsgType ∈ MessageTypes

Funktionen:

GetQuote (returns a quotation message –taking into account a given quote request)

CreateForecastEntry (creates an entry in the SmartForecasting system)

ReturnMsgType (returns the message type of a message)

PublishedMsg: SWAgents → Messages

ReceivedMsg: SWAgents → Messages

SubscribedTo: MessageTypes → SWAgents

ASM:

```
ProcessQuoteRequest =def
```

```
    RFQProcessing
    SmartQuotationAdapter
    SmartForecastingAdapter
if Initialized
    MessageBroker
```

```
RFQProcessing =def
```

```
if !Initialized
    SubscribeToMessage( SWA_RFQProcessing, Quote )
    Initialized := true
if Initialized
    PublishMessage( SWA_RFQProcessing, QuoteRequestMsg )
if state = QuoteDelivered
    newQuoteMsg := ReceivedMsg( SWA_RFQProcessing )
    state := newQuoteAvailable
if state = newQuoteAvailable
    PublishMessage(SWA_RFQProcessing, newQuoteMsg)
```

```
SmartQuotationAdapter =def
```

```
if !Initialized
    SubscribeToMessage( SWA_SmartQuotationAdapter, QuoteRequest )
    Initialized := true
if state = RequestDelivered
    QuoteMsg := GetQuote( ReceivedMsg( SWA_SmartQuotationAdapter ) )
    state := QuoteAvailable
if state = QuoteAvailable
    PublishMessage( SWA_SmartQuotationAdapter, QuoteMsg )
```

```
SmartForecastingAdapter =def
```

```
if !Initialized
    SubscribeToMessage( SWA_SmartForecastingAdapter, newQuote )
    Initialized := true
if state = newQuoteDelivered
    CreateForecastEntry( ReceivedMsg(SWA_SmartForecastingAdapter) )
    state := EntryAvailable
```

SubscribeToMessage(swagent, MsgType) =<sub>def</sub>  
    SubscribedTo(MsgType) := swagent

PublishMessage(swagent, Msg) =<sub>def</sub>  
    PublishedMsg(swagent) := Msg  
    **if** ReturnMsgType(Msg) = QuoteRequest  
        state := RequestPublicated  
    **if** ReturnMsgType(Msg) = newQuote  
        state := newQuotePublicated  
    **if** ReturnMsgType(Msg) = Quote  
        state := QuotePublicated

MessageBroker =<sub>def</sub>  
    **if** state = RequestPublicated  
        **choose** swagent ∈ SWAgents  
        **where** swagent = SubscribedTo( QuoteRequest )  
        DeliverMessage ( swagent, PublishedMsg( SubscribedTo(Quote) ) )  
    **if** state = QuotePublicated  
        **choose** swagent ∈ SWAgents  
        **where** swagent = SubscribedTo(Quote)  
        DeliverMessage ( swagent, PublishedMsg(SubscribedTo(QuoteRequest)))  
    **if** state = newQuotePublicated  
        **choose** swagent ∈ SWAgents  
        **where** swagent = SubscribedTo( newQuote )  
        DeliverMessage ( swagent, PublishedMsg( SubscribedTo(Quote) ) )

DeliverMessage(swagent, Msg) =<sub>def</sub>  
    ReceivedMsg(swagent) := Msg  
    **if** ReturnMsgType(Msg) = QuoteRequest  
        state := RequestDelivered  
    **if** ReturnMsgType(Msg) = newQuote  
        state := newQuoteDelivered  
    **if** ReturnMsgType(Msg) = Quote  
        state := QuoteDelivered

**Workflow-Spezifikation eines Rechnungsprozesses (Grafik Seite 85)\*:**

1) statisch:

-(1) ∈ **Point**

-(2) ∈ **Point**

-(3) ∈ **Point**  
 -(4) ∈ **Point**  
 CheckIfOfferedProduct ∈ **Object**  
 CheckIfWorthProceeding ∈ **Object**  
 GetQuoteFromQuotationSystem ∈ **Object**  
 GetQuoteFromSupplier ∈ **Object**  
 UpdateQuotationSystem ∈ **Object**  
 SendQuoteToCustomer ∈ **Object**  
 EnterQuoteInForecastingSystem ∈ **Object**  
 ApplyOrder ∈ **Object**  
 Offered ∈ **Boolean**  
 GoAhead ∈ **Boolean**  
 ContractExists ∈ **Boolean**

**Relationen:**

-(1) predOf CheckIfOfferedProduct  
 OrStatement  
     (CheckIfOfferedProduct, Offered, GetQuoteFromQuotationSystem, CheckIfWorthProceeding)  
 OrStatement  
     (CheckIfWorthProceeding, GoAhead, GetQuoteFromQuotationSystem, -(2))  
 OrStatement  
     (GetQuoteFromQuotationSystem, ContractExists, Proceed, GetQuoteFromSupplier)  
 GetQuoteFromSupplier predOf UpdateQuotationSystem  
 UpdateQuotationSystem predOf Proceed  
 AndStatement ({Proceed}, {SendQuoteToCustomer, EnterQuoteInForecastingSystem})

2) dynamisch:

Variablen:

OrderedProduct ∈ Product  
 Offered ∈ Boolean  
 GoAhead ∈ Boolean  
 ContractExists ∈ Boolean  
 Quote ∈ Message  
 Updated ∈ Boolean  
 MsgSend ∈ Boolean  
 QuoteEntered ∈ Boolean  
 Finished ∈ Boolean

Funktionen:

CheckIfOfferedProduct: Product → Boolean  
 CheckIfWorthProceeding: Product → Boolean  
 GetQuoteFromQuotationSystem: Product → {Boolean, Message}  
 GetQuoteFromSupplier: Product → Message  
 UpdateQuotationSystem: Message → Boolean  
 SendQuoteToCustomer: Message → Boolean  
 EnterQuoteInForecastingSystem: Message → Boolean  
 FinishProcess: → Boolean

ASM:

QuotationProcess =<sub>def</sub>  
     **If** OrderedProduct != **undef** **and** Offered = **undef**  
         Offered := CheckIfOfferedProduct(OrderedProduct)  
     **If** (Offered **or** GoAhead) **and** ContractExists = **undef**  
         {ContractExists, Quote} := GetQuoteFromQuotationSystem(OrderedProduct)  
     **If** !Offered **and** GoAhead = **undef**  
         GoAhead := CheckIfWorthProceeding(OrderedProduct)

```

If !GoAhead
    Finished := FinishProcess()
If !ContractExists and Quote = undef
    Quote := GetQuoteFromSupplier(OrderedProduct)
If !ContractExists and Quote != undef
    Updated := UpdateQuotationSystem(Quote)
If ContractExists or Updated
    MsgSend := SendQuoteToCustomer(Quote)
    QuoteEntered := EnterQuoteInForecastingSystem(Quote)
    Finished := FinishProcess()

```

### **Grundaufgaben eines WfMS (Grafik Seite 86):**

```

ResourceBroker ∈ Activity
WorkflowEngine ∈ Activity
WorkflowDefinitions ∈ Specification
WorkflowDesigner ∈ Object
CompletedWorkItems ∈ Queue
Resource1 ∈ Queue
Resource2 ∈ Queue
ResourceN ∈ Queue
InboundQueue ∈ Container
OutboundQueues ∈ Container

```

#### **Relationen:**

```
CompletedWorkItems partOf InboundQueue
```

```
Resource1 partOf OutboundQueues
Resource2 partOf OutboundQueues
ResourceN partOf OutboundQueues
```

```
numberOf WorkflowDefinitions
```

```
WorkflowDesigner comm WorkflowDefinitions
```

```
label ((CompletedWorkItems predOf WorkflowEngine), 1)
label ((WorkflowDefinitions predOf WorkflowEngine), 2)
label ((WorkflowEngine predOf ResourceBroker), 3)
label ((ResourceBroker predOf WorkflowEngine), 4)

```

```
label ((WorkflowEngine predOf Resource1), 5)
label ((WorkflowEngine predOf Resource2), 5)
label ((WorkflowEngine predOf ResourceN), 5)

```

### **Kombination von Workflow- und EAI-Techniken (Grafik Seite 90):**

```

WfMS ∈ Object
WfMSAdapter ∈ Object
MessageBrokerSystem ∈ Container (von Seite 78)

```

#### **Relationen:**

```

WfMS comm WfMSAdapter
WfMSAdapter comm MessageBrokerSystem

```

## Kapitel 4

### Client-/Server-Interaktion über Zwischenhändler (Grafik Seite 96):

HTTPClient ∈ **Object**  
HTTPProxy ∈ **Object**  
Firewall(1) ∈ **Activity**  
WideAreaNetwork(Internet) ∈ **Activity**  
Firewall(2) ∈ **Activity**  
HTTPGateway ∈ **Object**  
HTTPServer ∈ **Object**  
HTTPTunnel ∈ **Tunnel**

#### **Relationen:**

HTTPClient comm HTTPProxy  
tunnelled(HTTPProxy, {Firewall(1), WideAreaNetwork(Internet), Firewall(2)},  
HTTPGateway, HTTPTunnel)  
HTTPGateway comm HTTPServer

### Verschlüsselung zwischen HTTPS Servern und Clients (Grafik Seite 97):

HTTPSClient ∈ **Object**  
HTTPSServer ∈ **Object**  
ApplicationLayer ∈ **Container**  
SecureSocketsLayer(SSL) ∈ **Activity**  
TCP/IP ∈ **Activity**  
NetworkLayer ∈ **Container**

#### **Relationen:**

HTTPSClient partOf ApplicationLayer  
HTTPSServer partOf ApplicationLayer

SecureSocketsLayer(SSL) partOf NetworkLayer  
TCP/IP partOf NetworkLayer

HTTPSClient comm SecureSocketsLayer(SSL)  
HTTPSServer comm SecureSocketsLayer(SSL)  
SecureSocketsLayer(SSL) comm TCP/IP

### Erweiterung der 3-tier Architektur (Grafik Seite 99):

Client(1) ∈ **Object**  
Middleware ∈ **Activity**  
Server(ResourceManager) ∈ **Activity**  
HTTPServer ∈ **Object**  
WideAreaNetwork(Internet) ∈ **Activity**  
Client(2) ∈ **Object**  
HTTPClient ∈ **Object**

**Relationen:**

Client(1) comm Middleware  
 Middleware comm HTTPServer  
 Middleware comm Server(ResourceManager)

HTTPClient partOf Client(2)

WideAreaNetwork(Internet) part HTTPServer  
 WideAreaNetwork(Internet) part HTTPClient

**Applets als Möglichkeit der Client-Implementation (Grafik Seite 100):**

WebServer ∈ **Activity**  
 Middleware ∈ **Activity**  
 Server(ResourceManager) ∈ **Activity**  
 Firewall ∈ **Activity**  
 WideAreaNetwork(Internet) ∈ **Activity**  
 Client ∈ **RawObject**  
 Applet ∈ **Object**  
 JavaVirtualMachine ∈ **Object**  
 Browser ∈ **Object**  
 MiddlewareAndServer ∈ **Container** (für Grafik auf Seite 101)  
 FirewallAndWAN ∈ **Container** (für Grafik auf Seite 101)

**Relationen:**

Applet implements Client  
 Applet partOf JavaVirtualMachine  
 JavaVirtualMachine partOf Browser

Server(ResourceManager) comm Middleware  
 Middleware comm WebServer  
 tunnelled(WebServer, {Firewall, WideAreaNetwork(Internet)}, Client)

{Middleware, Server(ResourceManager)} partOf MiddlewareAndServer  
 {Firewall, WideAreaNetwork(Internet)} partOf FirewallAndWAN

**CGI-Programme als Interfaces (Grafik Seite 101):**

MiddlewareAndServer ∈ **Container** (siehe Seite 100)  
 FirewallAndWAN ∈ **Container** (siehe Seite 100)  
 Client ∈ **Object**  
 CGIProgram ∈ **Object**  
 WebServer ∈ **Activity**  
 Browser ∈ **Object**  
 HTTPGETRequest ∈ **Tunnel**

**Relationen:**

CGIProgram implements Client  
 MiddlewareAndServer comm CGIProgram  
 CGIProgram comm WebServer  
 tunnelled (WebServer, {FirewallAndWAN}, Browser, HTTPGETRequest)

### **Servlets zur Interaktion mit Serveranwendungen (Grafik Seite 103):**

MiddlewareAndServer ∈ **Container** (siehe Seite 100)

FirewallAndWAN ∈ **Container** (siehe Seite 100)

Browser ∈ **Object**

HTTPGETRequest ∈ **Tunnel**

Client ∈ **Object**

JavaThread ∈ **Object**

JavaServerProcess ∈ **Object**

WebServer ∈ **Object**

#### **Relationen:**

MiddlewareAndServer comm Client

JavaThread implements Client

JavaThread partOf JavaServerProcess

JavaServerProcess partOf WebServer

tunnelled (WebServer, {FirewallAndWAN}, Browser, HTTPGETRequest)

### **Schichten der Anwendungsserver (Grafik Seite 104):**

FirewallAndWAN ∈ **Container** (siehe Seite 100)

ResourceManagementLayer ∈ **Activity**

ApplicationServer ∈ **Activity**

ApplicationLogicLayer ∈ **Activity**

ConnectionToResourceMgmtLayer ∈ **Container**

PresentationLayer ∈ **Activity**

WebServer ∈ **Object**

OtherServers(email,SOAP,..) ∈ **Object**

Browser ∈ **Object**

Client ∈ **Object**

HTTP ∈ **Tunnel**

OtherProtocols ∈ **Tunnel**

#### **Relationen:**

ConnectionToResourceMgmtLayer partOf ApplicationLogicLayer

ApplicationLogicLayer partOf ApplicationServer

PresentationLayer partOf ApplicationServer

ConnectionToResourceMgmtLayer comm ResourceManagementLayer

ApplicationLogicLayer comm PresentationLayer

ApplicationServer comm OtherServers(email,SOAP,..)

ApplicationServer comm WebServer

tunnelled (OtherServers(email,SOAP,..), {FirewallAndWAN}, Client, OtherProtocols)

tunnelled (WebServer, {FirewallAndWAN}, Browser, HTTP)

### **Auszug aus der J2EE-Spezifikation (Grafik Seite 105):**

Servlets ∈ **Object**

JavaServerPages(JSP) ∈ **Object**

JavaAPIForXMLProcessing(JASP) ∈ **Object**  
JavaMail ∈ **Object**  
JavaAuthenticationAndAuthorizationService(JAAS) ∈ **Object**  
EnterpriseJavaBeans(EJB) ∈ **Object**  
JavaMessageService(JMS) ∈ **Object**  
JavaTransactionAPI(JTA) ∈ **Object**  
JavaNamingAndDirectoryInterface(JNDI) ∈ **Object**  
JavaDatabaseConnectivity(JDBC) ∈ **Object**  
Java2ConnectorArchitecture(J2CA) ∈ **Object**  
SupportForCommunicationAndPresentation ∈ **Container**  
SupportForTheApplicationIntegration ∈ **Container**  
SupportForAccessToResourceManagers ∈ **Container**

### **Relationen:**

Servlets partOf SupportForCommunicationAndPresentation  
JavaServerPages(JSP) partOf SupportForCommunicationAndPresentation  
JavaAPIForXMLProcessing(JASP) partOf SupportForCommunicationAndPresentation  
JavaMail partOf SupportForCommunicationAndPresentation  
JavaAuthenticationAndAuthorizationService(JAAS) partOf  
SupportForCommunicationAndPresentation

EnterpriseJavaBeans(EJB) partOf SupportForTheApplicationIntegration  
JavaMessageService(JMS) partOf SupportForTheApplicationIntegration  
JavaTransactionAPI(JTA) partOf SupportForTheApplicationIntegration  
JavaNamingAndDirectoryInterface(JNDI) partOf SupportForTheApplicationIntegration

JavaDatabaseConnectivity(JDBC) partOf SupportForAccessToResourceManagers  
Java2ConnectorArchitecture(J2CA) partOf SupportForAccessToResourceManagers

### **Anwendungsserver zur Unterstützung der Anwendungslogik (Grafik Seite 106):**

ApplicationServer ∈ **Activity**  
DBMSApplications ∈ **Resource**  
EnterpriseSystem1 ∈ **Activity**  
EnterpriseSystem2 ∈ **Activity**  
EnterpriseSystemN ∈ **Activity**  
PresentationLayer ∈ **Activity**  
ApplicationLogicLayer ∈ **Activity**  
Services(LoadBalancing, Pooling, Caching, Transaction, Persistence,...) ∈ **Object**  
EJBContainer ∈ **Object**  
JNDI ∈ **Object**  
JMS ∈ **Object**  
EJB(1) ∈ **Object**  
EJB(2) ∈ **Object**  
EJB(3) ∈ **Object**  
JDBC ∈ **Object**  
J2CAResourceAdapter(1) ∈ **Object**  
J2CAResourceAdapter(2) ∈ **Object**  
OtherAdapters ∈ **Object**  
Administration(ManagementAndSecurity) ∈ **Object**

**Relationen:**

PresentationLayer partOf ApplicationServer  
 ApplicationLogicLayer partOf ApplicationServer

Services(LoadBalancing,Pooling,Caching,Transaction,Persistence,..) partOf ApplicationLogicLayer  
 EJBContainer partOf ApplicationLogicLayer  
 EJB(1) partOf EJBContainer  
 EJB(2) partOf EJBContainer  
 EJB(3) partOf EJBContainer  
 JNDI partOf ApplicationLogicLayer  
 JMS partOf ApplicationLogicLayer  
 JDBC partOf ApplicationLogicLayer  
 J2CAResourceAdapter(1) partOf ApplicationLogicLayer  
 J2CAResourceAdapter(2) partOf ApplicationLogicLayer  
 OtherAdapters partOf ApplicationLogicLayer  
 Administration(ManagementAndSecurity) partOf ApplicationLogicLayer  
 DBMSApplication comm JDBC  
 EnterpriseSystem1 comm J2CAResourceAdapter(1)  
 EnterpriseSystem2 comm J2CAResourceAdapter(2)  
 EnterpriseSystemN comm OtherAdapters

**Anwendungsserver zur Unterstützung des "Presentation Layers" (Grafik Seite 109):**

ResourceManagementLayer ∈ **Activity**  
 ApplicationServer ∈ **Activity**  
 ApplicationLogicLayer ∈ **Activity** → könnte dem auf Seite 106 entsprechen  
 ConnectionToResourceMgmtLayer ∈ **Container**  
 PresentationLayer ∈ **Activity**  
 Services ∈ **Object**  
 MultideviceContentDelivery ∈ **Object**  
 Servlets ∈ **Object**  
 JSPs ∈ **Object**  
 XMLSupport ∈ **Object**  
 WebServicesSupport ∈ **Object**  
 PersonalizationLogic ∈ **Object**  
 Administration ∈ **Object**  
 ServersForOtherConnections(WAP...) ∈ **Object**  
 Client ∈ **Object**  
 E-MailServer ∈ **Object**  
 WebServer ∈ **Object**

**Relationen:**

PresentationLayer partOf ApplicationServer  
 ApplicationLogicLayer partOf ApplicationServer

ConnectionToResourceMgmtLayer partOf ApplicationLogicLayer  
 ConnectionToResourceMgmtLayer comm ResourceManagementLayer

ApplicationLogicLayer comm PresentationLayer

Services partOf PresentationLayer  
 MultideviceContentDelivery partOf PresentationLayer

Servlets partOf PresentationLayer  
JSPs partOf PresentationLayer  
XMLSupport partOf PresentationLayer  
WebServicesSupport partOf PresentationLayer  
PersonalizationLogic partOf PresentationLayer  
Administration partOf PresentationLayer

ServletsForOtherConnections(WAP...) comm ApplicationServer  
Client equal ServletsForOtherConnections(WAP...)  
E-MailServer comm ApplicationServer  
WebServer comm ApplicationServer

### **Mögliche Verbindung zwischen zwei 3-tier Systemen (Grafik Seite 112):**

Client(1) ∈ **Object**  
Middleware(1) ∈ **Activity**  
Server(ResourceManager)(1) ∈ **Object**  
Client(2) ∈ **Object**  
Middleware(2) ∈ **Activity**  
Server(ResourceManager)(2) ∈ **Object**  
WideAreaNetwork(Internet) ∈ **Activity**

#### **Relationen:**

Client(1) comm Middleware(1)  
Middleware(1) comm Server(ResourceManager)(1)  
Client(2) comm Middleware(2)  
Middleware(2) comm Server(ResourceManager)(2)  
tunnelled (Client(1), {WideAreaNetwork(Internet)}, Client(2))  
tunnelled (Client(1), {WideAreaNetwork(Internet)}, Middleware(2))  
tunnelled (Middleware(1), {WideAreaNetwork(Internet)}, Client(2))  
tunnelled (Middleware(1), {WideAreaNetwork(Internet)}, Middleware(2))  
tunnelled (Middleware(1), {WideAreaNetwork(Internet)}, Server(ResourceManager)(2))  
tunnelled (Server(ResourceManager)(1), {WideAreaNetwork(Internet)}, Middleware(2))

### **Integration von Middleware-Plattformen (Grafik Seite 114):**

WideAreaNetwork(Internet) ∈ **Activity**  
3tierSystem(1) ∈ **Container** → Erweiterung für Grafik auf Seite 116  
3tierSystem(2) ∈ **Container** → dito  
Client(1) ∈ **Object**  
Middleware(1) ∈ **Object**  
Remote-MiddlewareProtocol(1) ∈ **Activity**  
WANCommunicationProtocol(1) ∈ **Activity**  
Server(ResourceManager)(1) ∈ **Activity**  
Client(2) ∈ **Object**  
Middleware(2) ∈ **Object**  
Remote-MiddlewareProtocol(2) ∈ **Activity**  
WANCommunicationProtocol(2) ∈ **Activity**  
Server(ResourceManager)(2) ∈ **Activity**

**Relationen:**

Remote-MiddlewareProtocol(1) partOf Middleware(1)  
 WANCommunicationProtocol(1) partOf Middleware(1)  
 Remote-MiddlewareProtocol(1) comm WANCommunicationProtocol(1)  
 Middleware(1) comm Client(1)  
 Middleware(1) comm Server(ResourceManager)(1)  
 Client(1) partOf 3tierSystem(1)  
 Middleware(1) partOf 3tierSystem(1)  
 Server(ResourceManager)(1) partOf 3tierSystem(1)

Remote-MiddlewareProtocol(2) partOf Middleware(2)  
 WANCommunicationProtocol(2) partOf Middleware(2)  
 Remote-MiddlewareProtocol(2) comm WANCommunicationProtocol(2)  
 Middleware(2) comm Client(2)  
 Middleware(2) comm Server(ResourceManager)(2)  
 Client(2) partOf 3tierSystem(2)  
 Middleware(2) partOf 3tierSystem(2)  
 Server(ResourceManager)(2) partOf 3tierSystem(2)

*tunnelled* (WANCommunicationProtocol(1), {WideAreaNetwork(Intenet)},  
 WANCommunicationProtocol(2))

**B2B Interaktion auf Middleware-Level (Grafik Seite 116):**

3tierSystem(1) ∈ **Container** → siehe Seite 114  
 3tierSystem(2) ∈ **Container** → dito  
 WebServer(1) ∈ **Object**  
 WebServer(2) ∈ **Object**  
 HTTPSTunnel ∈ **Tunnel**  
 Firewall(1) ∈ **Activity**  
 Firewall(2) ∈ **Activity**  
 WideAreaNetwork(Internet) ∈ **Activity**

**Relationen:**

*tunnelled*  
 (WANCommunicationProtocol(1), {WebServer(1), Firewall(1), WideAreaNetwork(Internet),  
 Firewall(2), WebServer(2)}, WANCommunicationProtocol(2), HTTPSTunnel)

## Kapitel 5

**Manuelle Integration firmenübergreifender Prozesse (Grafik Seite 126):**

Customer ∈ **Activity**  
 InternalInfrastructure(1) ∈ **Object**  
 ManualIntegration(1) ∈ **Object**  
 Supplier ∈ **Activity**  
 WebServer(1) ∈ **Object**  
 InternalInfrastrucutre(2) ∈ **Object**

ManualIntegration(2) ∈ **Object**  
Warehouse ∈ **Activity**  
WebServer(2) ∈ **Object**  
InternalInfrastrucutre(3) ∈ **Object**  
ManualIntegration(3) ∈ **Object**

**Relationen:**

ManualIntegration(1) partOf Customer  
InternalInfrastructure(1) partOf Customer

WebServer(1) partOf Supplier  
InternalInfrastructure(2) partOf Supplier  
ManualIntegration(2) partOf Supplier

WebServer(2) partOf Supplier  
InternalInfrastructure(3) partOf Supplier  
ManualIntegration(3) partOf Supplier

*label*((InternalInfrastructure(1) comm ManualIntegration(1)), InternalProcurementrequests)  
ManualIntegration(1) comm WebServer(1)

WebServer(1) comm InternalInfrastructure(2)  
InternalInfrastructure(2) comm ManualIntegration(2)

ManualIntegration(2) comm WebServer(2)  
WebServer(2) comm InternalInfrastructure(3)  
InternalInfrastructure(3) comm ManualIntegration(3)  
*label*((ManualIntegration(3) predOf WebServer(1)), B2BInteractions(WebPages,WebForms,E-Mail))

**B2B Integration im Sinne von EAI (Grafik Seite 127):**

Customer ∈ **Activity**  
Customer'sAdapters ∈ **Object**  
InternalInfrastructure(1) ∈ **Object**  
ThirdParty ∈ **Activity**  
WfMS(Global) ∈ **Object**  
WfMSAdapter ∈ **Object**  
MessageBroker ∈ **Activity**  
Warehouse ∈ **Activity**  
Warehouse'sAdapters ∈ **Object**  
InternalInfrastructure(2) ∈ **Object**  
Supplier ∈ **Activity**  
Supplier'sAdapters ∈ **Object**  
InternalInfrastructure(3) ∈ **Object**

**Relationen:**

numberOf Customer'sAdapters  
numberOf Warehouse'sAdapters  
numberOf Supplier'sAdapters

Customer'sAdapters partOf Customer  
InternalInfrastructure(1) partOf Customer

Warehouse'sAdapters partOf Warehouse  
InternalInfrastructure(2) partOf Warehouse

Supplier'sAdapters partOf Supplier  
InternalInfrastructure(3) partOf Supplier

WfMS(Global) partOf ThirdParty  
WfMSAdapter partOf ThirdParty  
MessageBroker partOf ThirdParty

label ((InternalInfrastructure(1) comm Customer'sAdapters), InternalProcurementRequests)  
Customer'sAdapters comm MessageBroker

InternalInfrastructure(2) comm Warehouse'sAdapters  
Warehouse'sAdapters comm MessageBroker

InternalInfrastructure(3) comm Supplier'sAdapters  
Supplier'sAdapters comm MessageBroker

MessageBroker comm WfMSAdapter  
WfMSAdapter comm WfMS(Global)

### **Firmenübergreifende Integration (Grafik Seite 128):**

Customer ∈ **Activity**  
MessageBrokerXYZ(1) ∈ **Activity**  
Customer'sAdapters ∈ **Object**  
InternalInfrastructure(1) ∈ **Object**  
Supplier ∈ **Activity**  
MessageBrokerXYZ(2) ∈ **Activity**  
Supplier'sAdapters ∈ **Object**  
InternalInfrastructure(2) ∈ **Object**

#### **Relationen:**

numberOf Customer'sAdapters  
numberOf Supplier'sAdapters

InternalInfrastructure(1) partOf Customer  
Customer'sAdapters partOf Customer  
MessageBrokerXYZ(1) partOf Customer

InternalInfrastructure(2) partOf Supplier  
Supplier'sAdapters partOf Supplier  
MessageBrokerXYZ(2) partOf Supplier

InternalInfrastructure(1) comm Customer'sAdapters  
Customer'sAdapters comm MessageBrokerXYZ(1)  
MessageBrokerXYZ(1) comm MessageBrokerXYZ(2)  
MessageBrokerXYZ(2) comm Supplier'sAdapters  
Supplier'sAdapters comm InternalInfrastructure(2)

**Point-to-point Interaktion (Grafik Seite 129):**

Customer ∈ **Activity**  
Warehouse ∈ **Activity**  
AnotherParty(XYZ) ∈ **Activity**  
YetAnotherParty(ABC) ∈ **Activity**  
Supplier ∈ **Activity**  
MiddlewareForSupplier-CustomerInteraction ∈ **Object**  
MiddlewareForSupplier-WarehouseInteraction ∈ **Object**  
MiddlewareForSupplier-XYZInteraction ∈ **Object**  
MiddlewareForSupplier-ABCInteraction ∈ **Object**  
MiddlewareForIntegratingTheMiddleware ∈ **Object**  
Supplier'sAdapters(1) ∈ **Object**  
Supplier'sAdapters(2) ∈ **Object**  
Supplier'sAdapters(3) ∈ **Object**  
InternalInfrastructure ∈ **Object**

**Relationen:**

numberOf Supplier'sAdapters(1)  
numberOf Supplier'sAdapters(2)  
numberOf Supplier'sAdapters(3)

MiddlewareForSupplier-CustomerInteraction partOf Supplier  
MiddlewareForSupplier-WarehouseInteraction partOf Supplier  
MiddlewareForSupplier-XYZInteraction partOf Supplier  
MiddlewareForSupplier-ABCInteraction partOf Supplier  
MiddlewareForIntegratingTheMiddleware partOf Supplier  
Supplier'sAdapters(1) partOf Supplier  
Supplier'sAdapters(2) partOf Supplier  
Supplier'sAdapters(3) partOf Supplier  
InternalInfrastructure partOf Supplier

Customer comm Warehouse  
Customer comm YetAnotherParty(ABC)  
Customer comm MiddlewareForSupplier-CustomerInteraction

Warehouse comm AnotherParty(XYZ)  
Warehouse comm MiddlewareForSupplier-WarehouseInteraction

AnotherParty(XYZ) comm MiddlewareForSupplier-XYZInteraction  
YetAnotherParty(ABC) comm MiddlewareForSupplier-ABCInteraction

**Lösungsansatz von WebServices (Grafik Seite 134):**

Customer ∈ **Activity**  
WebService(1) ∈ **Object**  
InternalStructure(1) ∈ **Object**

Supplier ∈ **Activity**  
WebService(2) ∈ **Object**  
InternalStructure(2) ∈ **Object**

Warehouse ∈ **Activity**  
WebService(3) ∈ **Object**

InternalStructure(3) ∈ **Object**

**Relationen:**

WebService(1) interfaceOf Customer  
InternalInfrastructure(1) partOf Customer

WebService(2) interfaceOf Supplier  
InternalInfrastructure(2) partOf Supplier

WebService(3) interfaceOf Warehouse  
InternalInfrastructure(3) partOf Warehouse

label((WebService(1) comm InternalInfrastructure(1)), InternalProcurementRequests)  
WebService(1) comm WebService(2)

label((WebService(2) comm InternalInfrastructure(2)),  
StandardizedLanguagesAndProtocolsNeedOnlyWebServiceMiddleware)

label((WebService(3) predOf WebService(2)), ProtocolBasedInteractions)  
WebService(3) comm InternalInfrastructure(3)

**Einstiegspunkte zu lokalen Services (Grafik Seite 135):**

CompanyA(Provider) ∈ **Activity**  
InternalService(1) ∈ **Object**  
InternalService(2) ∈ **Object**  
Middleware(1) ∈ **Activity**  
WebService(1) ∈ **Activity**  
WebService(2) ∈ **Activity**  
WebService(3) ∈ **Activity**  
WideAreaNetwork(Internet) ∈ **Activity**  
CompanyB(Client) ∈ **Activity**  
InternalService(3) ∈ **Object**  
InternalService(4) ∈ **Object**  
Middleware(2) ∈ **Activity**  
Client ∈ **RawObject**

**Relationen:**

InternalService(1) partOf CompanyA(Provider)  
InternalService(2) partOf CompanyA(Provider)  
Middleware(1) partOf CompanyA(Provider)  
WebService(1) interfaceOf CompanyA(Provider)  
WebService(2) interfaceOf CompanyA(Provider)  
WebService(3) interfaceOf CompanyA(Provider)  
entryPoints (WebService(1))  
entryPoints (WebService(2))  
entryPoints (WebService(3))

InternalService(3) partOf CompanyB(Client)  
InternalService(4) partOf CompanyB(Client)  
Middleware(2) partOf CompanyB(Client)  
Client partOf CompanyB(Client)

InternalService(1) comm Middleware(1)  
InternalService(2) comm Middleware(1)  
Middleware(1) comm WebService(1)  
Middleware(1) comm WebService(2)  
Middleware(1) comm WebService(3)

InternalService(3) comm Middleware(2)  
InternalService(4) comm Middleware(2)  
Middleware(2) comm Client

tunnelled (WebService(2), {WideAreaNetwork(Internet)}, Client)

### **Anwendungsintegration mit Hilfe von WebServices (Grafik Seite 136):**

CompanyA ∈ **Activity**  
IntegratingApplication(WithCompositionLogic) ∈ **Object**  
WebService-EnabledBroker ∈ **Object**  
SmartQuotation ∈ **Activity**  
DBMSApplications ∈ **Resource**  
SmartForeCasting ∈ **Activity**  
SendmailApplication ∈ **Object**  
Users ∈ **Object**  
XYZ ∈ **Activity**  
ViaWebServicesAccessibleBack-EndSystems ∈ **Container**

#### **Relationen:**

numberOf Users  
IntegratingApplication(WithCompositionLogic) partOf CompanyA  
WebService-EnabledBroker partOf CompanyA  
  
SmartQuotation partOf ViaWebServicesAccessibleBack-EndSystems  
DBMSApplications partOf ViaWebServicesAccessibleBack-EndSystems  
SmartForeCasting partOf ViaWebServicesAccessibleBack-EndSystems  
SendmailApplication partOf ViaWebServicesAccessibleBack-EndSystems  
Users partOf ViaWebServicesAccessibleBack-EndSystems  
XYZ partOf ViaWebServicesAccessibleBack-EndSystems  
ViaWebServicesAccessibleBack-EndSystems partOf CompanyA

IntegratingApplication(WithCompositionLogic) comm WebService-EnabledBroker  
WebService-EnabledBroker comm SmartQuotation  
WebService-EnabledBroker comm DBMSApplications  
WebService-EnabledBroker comm SmartForecasting  
WebService-EnabledBroker comm SendmailApplication  
WebService-EnabledBroker comm XYZ  
SendmailApplication comm Users

### **Grundlage der Service-Beschreibungen (Grafik Seite 137):**

ServiceDescriptionAndDiscoveryStack ∈ **Container**  
VerticalStandards ∈ **Activity**  
PropertiesAndSemantics ∈ **Activity**  
BusinessProtocols ∈ **Activity**

Interfaces ∈ **Activity**  
CommonBaseLanguage ∈ **Activity**  
Directories ∈ **Activity**

**Relationen:**

VerticalStandards partOf ServiceDescriptionAndDiscoveryStack  
PropertiesAndSemantics partOf ServiceDescriptionAndDiscoveryStack  
BusinessProtocols partOf ServiceDescriptionAndDiscoveryStack  
Interfaces partOf ServiceDescriptionAndDiscoveryStack  
CommonBaseLanguage partOf ServiceDescriptionAndDiscoveryStack  
Directories partOf ServiceDescriptionAndDiscoveryStack

VerticalStandards equals PropertiesAndSemantics  
PropertiesAndSemantics equals BusinessProtocols  
BusinessProtocols equals Interfaces  
Interfaces equals CommonBaseLanguage

**Grundgerüst der Service-Interaktionen (Grafik Seite 139):**

ServiceInteractionStack ∈ **Container**  
MiddlewareProperties(HorizontalProtocols) ∈ **Activity**  
ProtocolInfrastructure(Meta-Protocols) ∈ **Activity**  
BasicAndSecureMessaging ∈ **Activity**  
Transport ∈ **Activity**

**Relationen:**

MiddlewareProperties(HorizontalProtocols) partOf ServiceInteractionStack  
ProtocolInfrastructure(Meta-Protocols) partOf ServiceInteractionStack  
BasicAndSecureMessaging partOf ServiceInteractionStack  
Transport partOf ServiceInteractionStack

MiddlewareProperties(HorizontalProtocols) equals ProtocolInfrastructure(Meta-Protocols)  
ProtocolInfrastructure(Meta-Protocols) equals BasicAndSecureMessaging  
BasicAndSecureMessaging equals Transport

**Interne und Externe Architekturen bei WebServices (Grafik Seite 142):**

CompanyA(Provider) ∈ **Activity**  
WebService(1) ∈ **Object**  
WebServiceInterface ∈ **Object**  
AccessToInternalSystems ∈ **Object**  
InternalArchitecture ∈ **Structure**  
Middleware ∈ **Activity**  
InternalService(1) ∈ **Object**  
InternalService(2) ∈ **Object**  
CompanyD(Client) ∈ **Activity**  
Client ∈ **RawObject**  
ExternalArchitecture ∈ **Structure**  
CompanyB(Provider) ∈ **Activity**  
WebService(2) ∈ **Object**  
WebService(3) ∈ **Object**  
CompanyC(Provider) ∈ **Activity**

WebService(4) ∈ **Object**  
WebService(5) ∈ **Object**  
WebService(6) ∈ **Object**

**Relationen:**

WebServiceInterface interfaceOf WebService(1)  
AccessToInternalsystems partOf WebService(1)  
WebService(1) partOf CompanyA(Provider)

InternalArchitecture partOf CompanyA(Provider)  
Middleware partOf CompanyA(Provider)  
InternalService(1) partOf CompanyA(Provider)  
InternalService(2) partOf CompanyA(Provider)

Client partOf CompanyD(Client)

WebService(2) partOf CompanyB(Provider)  
WebService(3) partOf CompanyB(Provider)

WebService(4) partOf CompanyC(Provider)  
WebService(5) partOf CompanyC(Provider)  
WebService(6) partOf CompanyC(Provider)

InternalService(1) comm Middleware  
InternalService(2) comm Middleware  
Middleware comm InternalArchitecture  
InternalArchitecture comm AccessToInternalsystems  
WebServiceInterface comm ExternalArchitecture

ExternalArchitecture comm Client  
ExternalArchitecture comm WebService(2)  
ExternalArchitecture comm WebService(5)

**Konventionelle Middleware als Integrationsplattform (Grafik Seite 144):**

OtherTiers ∈ **Container**  
ServiceInterface(1) ∈ **Object**  
IntegrationLogic(1) ∈ **Object**  
Middleware(1) ∈ **Activity**  
ServiceInterface(2) ∈ **Object**  
IntegrationLogic(2) ∈ **Object**  
Middleware(2) ∈ **Activity**  
ServiceInterface(3) ∈ **Object**  
IntegrationLogic(3) ∈ **Object**  
Middleware(3) ∈ **Activity**  
ResourceManager(1) ∈ **Object**  
ResourceManager(2) ∈ **Object**  
ResourceManager(3) ∈ **Object**  
ResourceManager(4) ∈ **Object**

**Relationen:**

ServiceInterface(1) equal IntegrationLogic(1)  
ServiceInterface(2) equal IntegrationLogic(2)

ServiceInterface(3) equal IntegrationLogic(3)

OtherTiers comm ServiceInterface(1)  
IntegrationLogic(1) comm Middleware(1)

Middleware(1) comm ServiceInterface(2)  
Middleware(1) comm ServiceInterface(3)

IntegrationLogic(2) comm Middleware(2)  
IntegrationLogic(3) comm Middleware(3)

Middleware(2) comm ResourceManager(1)  
Middleware(2) comm ResourceManager(2)

Middleware(3) comm ResourceManager(3)  
Middleware(3) comm ResourceManager(4)

### **Implementation von WebServices in Verbindung mit tier-Systemen (Grafik Seite 145):**

CompanyA(ServiceProvider) ∈ **Activity**  
WebServiceInterface ∈ **Object**  
AccessToInternalSystems ∈ **Object**  
WebServicesMiddleware(Internal) ∈ **Middle**  
ServiceInterface ∈ **Object**  
IntegrationLogic ∈ **Object**  
ConventionalMiddleware(IncludesMiddlewareServices) ∈ **Activity**  
OtherTiers ∈ **Container**  
ClientsFromOtherCompanies ∈ **Container**

#### **Relationen:**

WebServiceInterface equal AccessToInternalSystems

WebServiceInterface partOf CompanyA(ServiceProvider)  
AccessToInternalSystems partOf CompanyA(ServiceProvider)  
WebServiceMiddleware(Internal) partOf CompanyA(ServiceProvider)  
ServiceInterface partOf CompanyA(ServiceProvider)  
IntegrationLogic partOf CompanyA(ServiceProvider)  
ConventionalMiddleware(IncludesMiddlewareServices) partOf CompanyA(ServiceProvider)  
OtherTiers partOf CompanyA(ServiceProvider)  
ServiceInterface equal IntegrationLogic

ClientsFromOtherCompanies comm WebServicesMiddleware(Internal)  
WebServicesMiddleware(Internal) comm WebServiceInterface  
AccessToInternalSystems comm WebServicesMiddleware(Internal)  
WebServicesMiddleware(Internal) comm ConventionalMiddleware(IncludesMiddlewareServices)

ConventionalMiddleware(IncludesMiddlewareServices) comm IntegrationLogic  
ConventionalMiddleware(IncludesMiddlewareServices) comm OtherTiers

### **Externe Architektur von WebServices (Grafik Seite 146):**

CompanyA(ServiceRequester) ∈ **Activity**

WebServiceClient ∈ **Object**  
 WebServiceMiddleware(Internal)(1) ∈ **Middle**  
 OtherTiers(1) ∈ **Container**  
 CompanyB(ServiceProvider) ∈ **Activity**  
 WebService ∈ **Object**  
 WebServiceMiddleware(Internal)(2) ∈ **Middle**  
 OtherTiers(2) ∈ **Object**  
 CompanyC(DirectoryServiceProvider) ∈ **Activity**  
 ServiceDescriptions ∈ **Specification**  
 3.Interact ∈ **Activity**  
 2.Find ∈ **Activity**  
 1.PublishTheServiceDescription ∈ **Activity**

**Relationen:**

numberOf ServiceDescriptions  
 ServiceDescriptions partOf CompanyC(DirectoryServiceProvider)

WebServiceClient partOf CompanyA(ServiceRequester)  
 WebServiceMiddleware(Internal)(1) partOf CompanyA(ServiceRequester)  
 OtherTiers(1) partOf CompanyA(ServiceRequester)

WebService partOf CompanyB(ServiceProvider)  
 WebServiceMiddleware(Internal)(2) partOf CompanyB(ServiceProvider)  
 OtherTiers(2) partOf CompanyB(ServiceProvider)

WebServiceClient comm WebServiceMiddleware(Internal)(1)  
 WebServiceMiddleware(Internal)(1) comm OtherTiers(1)  
 WebServiceMiddleware(Internal)(1) comm 3.Interact  
 WebServiceMiddleware(Internal)(1) comm 2.Find

3.Interact comm WebServiceMiddleware(Internal)(2)

WebServiceMiddleware(Internal)(2) comm 1.PublishTheServiceDescription  
 WebService comm WebServiceMiddleware(Internal)(2)  
 WebServiceMiddleware(Internal)(2) comm OtherTiers(2)

1.PublishTheServiceDescription comm ServiceDescriptions  
 ServiceDescriptions comm 2.Find

1.PublishTheServiceDescription allows 2.Find  
 2.Find allows 3.Interact

**Externe Architektur von WebServices und P2P-Protokolle (Grafik Seite 148):**

CompanyA(ServiceRequester) ∈ **Activity**  
 WebServiceClient ∈ **Object**  
 InternalMiddleware(1) ∈ **Middle**  
 OtherTiers(1) ∈ **Container**  
 ExternalCapabilities(1) ∈ **Activity**  
 TransactionMgmt(1) ∈ **Object**  
 OtherProtocolInfrastructure(1) ∈ **Object**  
 CompositionEngine(1) ∈ **Object**

ExternalMiddleware ∈ **Container**  
 CompanyB(ServiceProvider) ∈ **Activity**  
 WebService ∈ **Object**  
 InternalMiddleware(2) ∈ **Middle**  
 OtherTiers(2) ∈ **Container**  
 ExternalCapabilities(2) ∈ **Activity**  
 TransactionMgmt(2) ∈ **Object**  
 OtherProtocolInfrastructure(2) ∈ **Object**  
 CompositionEngine(2) ∈ **Object**  
 CompanyC(DirectoryServiceProvider) ∈ **Activity**  
 ServiceDescriptions ∈ **Specification**

**Relationen:**

numberOf ServiceDescriptions  
 ServiceDescriptions partOf CompanyC(DirectoryServiceProvider)

WebServiceClient partOf CompanyA(ServiceRequester)  
 InternalMiddleware(1) partOf CompanyA(ServiceRequester)  
 OtherTiers(1) partOf CompanyA(ServiceRequester)  
 ExternalCapabilities(1) partOf CompanyA(ServiceRequester)

TransactionMgmt(1) partOf ExternalCapabilities(1)  
 OtherProtocolInfrastructure(1) partOf ExternalCapabilities(1)  
 CompositionEngine(1) partOf ExternalCapabilities(1)

WebService partOf CompanyB(ServiceProvider)  
 InternalMiddleware(2) partOf CompanyB(ServiceProvider)  
 OtherTiers(2) partOf CompanyB(ServiceProvider)  
 ExternalCapabilities(2) partOf CompanyB(ServiceProvider)

TransactionMgmt(2) partOf ExternalCapabilities(2)  
 OtherProtocolInfrastructure(2) partOf ExternalCapabilities(2)  
 CompositionEngine(2) partOf ExternalCapabilities(2)

WebServiceClient comm InternalMiddleware(1)  
 InternalMiddleware(1) comm OtherTiers(1)  
 InternalMiddleware(1) comm ExternalCapabilities(1)  
 ExternalCapabilities(1) comm ServiceDescriptions  
 ExternalCapabilities(1) comm ExternalCapabilities(2)

ExternalCapabilities(2) comm ServiceDescriptions  
 ExternalCapabilities(2) comm InternalMiddleware(2)  
 InternalMiddleware(2) comm OtherTiers(2)  
 InternalMiddleware(2) comm WebService

ExternalCapabilities(1) partOf ExternalMiddleware  
 ExternalCapabilities(2) partOf ExternalMiddleware  
 CompanyC(DirectoryServiceProvider) partOf ExternalMiddleware

## Kapitel 6

### Aufruf von WebServices mit SOAP-Nachrichten (Grafik Seite 153):

ServiceRequestor  $\in$  **Activity**  
ApplicationObject(Client)  $\in$  **Object**  
SOAP-BasedMiddleware(1)  $\in$  **Object**  
ServiceProvider  $\in$  **Activity**  
ApplicationObject(ServiceProvider)  $\in$  **Object**  
SOAP-BasedMiddleware(2)  $\in$  **Object**

#### **Relationen:**

ApplicationObject(Client) partOf ServiceRequestor  
SOAP-BasedMiddleware(1) partOf ServiceRequestor  
ApplicationObject(ServiceProvider) partOf ServiceProvider  
SOAP-BasedMiddleware(2) partOf ServiceProvider

ApplicationObject(Client) comm SOAP-BasedMiddleware(1)  
label((SOAP-BasedMiddleware(1) comm SOAP-BasedMiddleware(2)),  
SOAPMessagesExchangedOnTopOfHTTP,SMTP,OtherTransportProtocols)

ApplicationObject(ServiceProvider) comm SOAP-BasedMiddleware(2)

### WSDL-Spezifikationen analog zu IDL-Spezifikationen (Grafik Seite 154):

ServiceRequestor  $\in$  **Activity**  
ApplicationObject(Client)  $\in$  **Object**  
Stub  $\in$  **Object**  
SOAP-BasedMiddleware(1)  $\in$  **Object**  
ServiceProvider  $\in$  **Activity**  
ApplicationObject(ServiceProvider)  $\in$  **Object**  
Skeleton  $\in$  **Object**  
SOAP-BasedMiddleware(2)  $\in$  **Object**  
WSDLCompiler(ClientSide)  $\in$  **Object**  
WSDLCompiler(ServerSide)  $\in$  **Object**  
WSDLOfServiceProvider  $\in$  **Specification**

#### **Relationen:**

ApplicationObject(Client) partOf ServiceRequestor  
Stub partOf ServiceRequestor  
SOAP-BasedMiddleware(1) partOf ServiceRequestor  
ApplicationObject(ServiceProvider) partOf ServiceProvider  
Skeleton partOf ServiceProvider  
SOAP-BasedMiddleware(2) partOf ServiceProvider

ApplicationObject(Client) equal Stub  
Stub comm SOAP-BasedMiddleware(1)

label((SOAP-BasedMiddleware(1) comm SOAP-BasedMiddleware(2)), SOAPMessages)

ApplicationObject(ServiceProvider) equal Skeleton  
Skeleton comm SOAP-BasedMiddleware(2)

WSDLofServiceProvider allows WSDLCompiler(ClientSide)  
WSDLofServiceProvider allows WSDLCompiler(ServerSide)

WSDLCompiler(ClientSide) allows Stub  
WSDLCompiler(ServerSide) allows Skeleton

**UDDI-Grundidee (Grafik Seite 155):**

ServiceRequestor ∈ **Activity** → siehe Seite 154  
ServiceProvider ∈ **Activity** → dito  
UDDIRegistry ∈ **Activity**  
ServiceDescriptions ∈ **Resource**  
SOAP-BasedMiddleware(3) ∈ **Object**  
LookForServices ∈ **Activity**  
PublishServiceDescription ∈ **Activity**  
-(1) ∈ **Port**  
-(2) ∈ **Port**

**Relationen:**

numberOf ServiceDescriptions  
-(1) portOf SOAP-BasedMiddleware(3)  
-(2) portOf SOAP-BasedMiddleware(3)  
entryPoints(SOAP-BasedMiddleware(3))

ServiceRequestor siehe Seite 154  
ServiceProvider siehe Seite 154

label((SOAP-BasedMiddleware(1) comm SOAP-BasedMiddleware(2)), SOAPMessages)  
SOAP-BasedMiddleware(1) comm LookForServices  
SOAP-BasedMiddleware(2) comm PublishServiceDescription

SOAP-BasedMiddleware(3) partOf UDDIRegistry  
ServiceDescriptions partOf UDDIRegistry  
ServiceDescriptions comm SOAP-BasedMiddleWare(3)

LookForServices comm -(1)  
PublishServiceDescription comm -(2)

**Schematische Darstellung einer SOAP-Nachricht (Grafik Seite 157):**

SOAPEnvelope ∈ **Activity**  
SOAPHeader ∈ **Activity**  
HeaderBlock ∈ **Activity**  
SOAPBody ∈ **Activity**  
BodyBlock ∈ **Activity**

**Relationen:**

numberOf HeaderBlock  
numberOf BodyBlock

HeaderBlock partOf SOAPHeader  
SOAPHeader partOf SOAPEnvelope

BodyBlock partOf SOAPBody  
SOAPBody partOf SOAPEnvelope

**Dokumentenbasierte Interaktion (Grafik Seite 159):**

SOAPEnvelope(1) ∈ **Activity**  
SOAPBody(1) ∈ **Activity**  
PurchaseOrderDocument(ProductItem,Quantity) ∈ **Activity**  
SOAPEnvelope(2) ∈ **Activity**  
SOAPBody(2) ∈ **Activity**  
AcknowledgementDocument(OrderID) ∈ **Activity**

**Relationen:**

PurchaseOrderDocument(ProductItem,Quantity) partOf SOAPBody(1)  
SOAPBody(1) partOf SOAPEnvelope(1)  
AcknowledgementDocument(OrderID) partOf SOAPBody(2)  
SOAPBody(2) partOf SOAPEnvelope(2)

**RPC-basierte Interaktion (Grafik Seite 159):**

(b-RPC Style Interaction)

SOAPEnvelope(1) ∈ **Activity**  
SOAPBody(1) ∈ **Activity**  
SOAPEnvelope(2) ∈ **Activity**  
SOAPBody(2) ∈ **Activity**  
MethodNameOrderGoods ∈ **Activity**  
InputParameter1ProductItem ∈ **Activity**  
InputParameter2ProductItem ∈ **Activity**  
MethodReturn ∈ **Activity**  
ReturnValueOrderID ∈ **Activity**

**Relationen:**

InputParameter1ProductItem partOf MethodNameOrderGoods  
InputParameter2ProductItem partOf MethodNameOrderGoods  
MethodNameOrderGoods partOf SOAPBody(1)  
SOAPBody partOf SOAPEnvelope(1)  
  
ReturnValueOrderID partOf MethodReturn  
MethodReturn partOf SOAPBody(2)  
SOAPBody(2) partOf SOAPEnvelope(2)

**Beispiel für einer SOAP-Nachricht (Grafik Seite 160):**

Envelope ∈ **Activity**  
Header ∈ **Activity**  
Blocks(1) ∈ **Activity**  
Body ∈ **Activity**  
Blocks(2) ∈ **Activity**

**Relationen:**

numberOf Blocks(1)

numberOf Blocks(2)

Blocks(1) partOf Header

Header partOf Envelope

Blocks(2) partOf Body

Body partOf Envelope

**RPC-Aufrufe via HTTP mit SOAP (Grafik Seite 162):**

ServiceRequestor ∈ **Activity**

SOAPEngine(1) ∈ **Object**

HTTPEngine(1) ∈ **Object**

ClientImplementation(OtherTiers) ∈ **Object**

ServiceProvider ∈ **Activity**

SOAPEngine(2) ∈ **Object**

HTTPEngine(2) ∈ **Object**

ServiceImplementation(OtherTiers) ∈ **Object**

HTTPPost(1) ∈ **Activity**

SOAPEnvelope(1) ∈ **Activity**

SOAPHeader(1) ∈ **Activity**

TransactionalContext(1) ∈ **Activity**

SOAPBody(1) ∈ **Activity**

NameOfTheProcedure ∈ **Activity**

InputParameter1 ∈ **Activity**

InputParameter2 ∈ **Activity**

HTTPPost(2) ∈ **Activity**

SOAPEnvelope(2) ∈ **Activity**

SOAPHeader(2) ∈ **Activity**

TransactionalContext(2) ∈ **Activity**

SOAPBody(2) ∈ **Activity**

ReturnParameter ∈ **Activity**

**Relationen:**

SOAPEngine(1) partOf ServiceRequestor

HTTPEngine(1) partOf ServiceRequestor

ClientImplementation(OtherTiers) partOf ServiceRequestor

SOAPEngine(2) partOf ServiceProvider

HTTPEngine(2) partOf ServiceProvider

ServiceImplementation(OtherTiers) partOf Provider

TransactionalContext(1) partOf SOAPHeader(1)

SOAPHeader(1) partOf SOAPEnvelope(1)

SOAPEnvelope(1) partOf HTTPPost(1)

NameOfTheProcedure partOf SOAPBody(1)

InputParameter1 partOf SOAPBody(1)

InputParameter2 partOf SOAPBody(1)

SOAPBody(1) partOf SOAPEnvelope

TransactionalContext(2) partOf SOAPHeader(2)

SOAPHeader(2) partOf SOAPEnvelope(2)

SOAPEnvelope(2) partOf HTTPPost(2)

ReturnParameter partOf SOAPBody(2)

SOAPBody(2) partOf SOAPEnvelope(2)

SOAPEngine(1) comm ClientImplementation(OtherTiers)

SOAPEngine(1) comm HTTPEngine(1)

HTTPEngine(1) comm HTTPPost(1)

HTTPEngine(1) comm HTTPPost(2)

SOAPEngine(2) comm ServiceImplementation(OtherTiers)

SOAPEngine(2) comm HTTPEngine(2)

HTTPEngine(2) comm HTTPPost(1)

HTTPEngine(2) comm HTTPPost(2)

### **Eine einfache SOAP-Implementierung (Grafik Seite 163):**

ServiceRequestor ∈ **Activity**

ClientImplementation ∈ **Object**

ClientStub ∈ **Object**

SOAPEngine ∈ **Object**

HTTPEngine ∈ **Object**

ServiceProvider ∈ **Activity**

ServiceImplementation ∈ **Object**

ServerStub ∈ **Object**

SOAPRouter ∈ **Object**

HTTPServer ∈ **Object**

InvokeServiceAsALocalCall ∈ **Container**

InvokeSOAPEngineToPrepareSOAPMessage ∈ **Container**

PackageSOAPIntoHTTPAnd PassItToAnHTTPClient ∈ **Container**

SendMessageToTheProvider ∈ **Container**

PassContentOfHTTPMessageToRouter ∈ **Container**

ParseMessage,IdentifyAppropriateStubAndDeliverParsedMessage ∈ **Container**

InvokeLocalProcedureOfServiceImplementation ∈ **Container**

### **Relationen:**

InvokeServiceAsALocalCall partOf ClientImplementation

ClientImplementation partOf ServiceRequestor

InvokeSOAPEngineToPrepareSOAPMessage partOf ClientStub

ClientStub partOf ServiceRequestor

PackageSOAPIntoHTTPAnd PassItToAnHTTPClient partOf SOAPEngine

SOAPEngine partOf ServiceRequestor

SendMessageToTheProvider partOf HTTPEngine

HTTPEngine partOf ServiceRequestor

ServiceImplementation partOf ServiceProvider

InvokeLocalProcedureOfServiceImplementation partOf ServerStub

ServerStub partOf ServiceProvider

ParseMessage,IdentifyAppropriateStubAndDeliverParsedMessage partOf SOAPRouter

SOAPRouter partOf ServiceProvider  
PassContentOfHTTPMessageToRouter partOf HTTPServer  
HTTPServer partOf ServiceProvider

InvokeServiceAsALocalCall predOf ClientStub  
InvokeSOAPEngineToPrepareSOAPMessage predOf SOAPEngine  
PackageSOAPIntoHTTPAnd PassItToAnHTTPClient predOf HTTPEngine  
SendMessageToTheProvider predOf HTTPServer  
PassContentOfHTTPMessageToRouter predOf SOAPRouter  
ParseMessage,IdentifyAppropriateStubAndDeliverParsedMessage predOf ServerStub  
InvokeLocalProcedureOfServiceImplementation predOf ServiceImplementation

**Beispiel für eine WSDL Service-Spezifikation (Grafik Seite 167):**

WSDLSpecification ∈ **Activity**  
AbstractPart ∈ **Activity**  
Types ∈ **Activity**  
Messages ∈ **Activity**  
Operations ∈ **Activity**  
PortTypes ∈ **Activity**  
ConcretePart ∈ **Activity**  
Bindings ∈ **Activity**  
ServicesAndPorts ∈ **Activity**

**Relationen:**

Types partOf AbstractPart  
Messages partOf AbstractPart  
Operations partOf AbstractPart  
PortTypes partOf AbstractPart  
AbstractPart partOf WSDLSpecification

Bindings partOf ConcretePart  
ServicesAndPorts partOf ConcretePart  
ConcretePart partOf WSDLSpecification

**WSDL-Spezifikation eines Services (Grafik Seite 170):**

AbstractPart ∈ **Activity**  
Messages ∈ **Activity**  
OperationAndPortType ∈ **Activity**  
ConcretePart ∈ **Activity**  
Binding ∈ **Activity**  
PortAndService ∈ **Activity**

**Relationen:**

Messages partOf AbstractPart  
OperationAndPortType partOf AbstractPart

Binding partOf ConcretePart  
PortAndService partOf ConcretePart

### Erzeugung von WSDL-Dokumenten mit Hilfe von APIs (Grafik Seite 173):

ServiceRequestor ∈ **Activity**  
ApplicationObject(Client) ∈ **Object**  
Stub ∈ **Object**  
SOAPBasedMiddleware(1) ∈ **Object**  
ServiceProvider ∈ **Activity**  
ApplicationObject(ServiceProvider) ∈ **Object**  
Skeleton ∈ **Object**  
SOAPBasedMiddleware(2) ∈ **Client**  
WSDLGenerator ∈ **Object**  
WSDLOfServiceProvider ∈ **Specification**  
WSDLCompiler(ClientSide) ∈ **Object**  
WSDLCompiler(ServerSide) ∈ **Object**

#### **Relationen:**

ApplicationObject(Client) partOf ServiceRequestor  
Stub partOf ServiceRequestor  
Stub equal ApplicationObject(Client)  
SOAPBasedMiddleware(1) partOf ServiceRequestor

ApplicationObject(ServiceProvider) partOf ServiceProvider  
Skeleton partOf ServiceProvider  
Skeleton equal ApplicationObject(ServiceProvider)  
SOAPBasedMiddleware(2) partOf ServiceProvider

ApplicationObject(ServiceProvider) allows WSDLGenerator  
WSDLGenerator allows WSDLOfServiceProvider  
WSDLOfServiceProvider allows WSDLCompiler(ClientSide)  
WSDLOfServiceProvider allows WSDLCompiler(ServerSide)  
WSDLCompiler(ClientSide) allows Stub  
WSDLCompiler(ServerSide) allows Skeleton

Stub comm SOAPBasedMiddleware(1)  
Skeleton comm SOAPBasedMiddleware(2)  
label(SOAPBasedMiddleware(1) comm SOAPBasedMiddleware(2)), SOAPMessages)

### schematische Darstellung eines UDDI-Eintrages (Grafik Seite 177):

UDDIRegistry ∈ **Container**  
businessEntity ∈ **Activity**  
Name(1) ∈ **Object**  
Contacts ∈ **Object**  
Description(1) ∈ **Object**  
Identifiers(1) ∈ **Object**  
Categories(1) ∈ **Object**  
businessService ∈ **Activity**  
ServiceKey ∈ **Object**  
Name(2) ∈ **Object**  
Description(2) ∈ **Object**  
Categories(2) ∈ **Object**  
bindingTemplate ∈ **Activity**  
BindingKey ∈ **Object**  
Description(3) ∈ **Object**

Address ∈ **Object**  
 DetailedInfo ∈ **Object**  
 ReferencesTotModels ∈ **Object**  
 tModel(1) ∈ **Activity**  
 Key(1) ∈ **Object**  
 Name(3) ∈ **Object**  
 Description(4) ∈ **Object**  
 OverviewDoc(1) ∈ **Object**  
 Identifiers(2) ∈ **Object**  
 Categories(3) ∈ **Object**  
 tModel(2) ∈ **Activity**  
 Key(2) ∈ **Object**  
 Name(4) ∈ **Object**  
 Description(5) ∈ **Object**  
 OverviewDoc(2) ∈ **Object**  
 Identifiers(3) ∈ **Object**  
 Categories(4) ∈ **Object**  
 SpecsStoredAtTheProvidersSide ∈ **Specification**

**Relationen:**

numberOf businessEntity  
 numberOf businessService  
 numberOf bindingTemplate  
 numberOf tModel(1)  
 numberOf tModel(2)

BindingKey partOf bindingTemplate  
 Description(3) partOf bindingTemplate  
 Address partOf bindingTemplate  
 DetailedInfo partOf bindingTemplate  
 ReferencesTotModels partOf bindingTemplate

bindingTemplate partOf businessService  
 ServiceKey partOf businessService  
 Name(2) partOf businessService  
 Description(2) partOf businessService  
 Categories(2) partOf businessService

businessService partOf businessEntity  
 Name(1) partOf businessEntity  
 Contacts partOf businessEntity  
 Description(1) partOf businessEntity  
 Identifiers(1) partOf businessEntity  
 Categories(1) partOf businessEntity

businessEntity partOf UDDIRegistry

Key(1) partOf tModel(1)  
 Name(3) partOf tModel(1)  
 Description(4) partOf tModel(1)  
 OverviewDoc(1) partOf tModel(1)  
 Identifiers(2) partOf tModel(1)  
 Categories(3) partOf tModel(1)  
 tModel(1) partOf UDDIRegistry

Key(2) partOf tModel(2)  
 Name(4) partOf tModel(2)  
 Description(4) partOf tModel(2)  
 OverviewDoc(2) partOf tModel(2)  
 Identifiers(3) partOf tModel(2)  
 Categories(4) partOf tModel(2)  
 tModel(2) partOf UDDIRegistry  
  
 Identifiers(1) predOf tModel(1)  
 Categories(1) predOf tModel(1)  
  
 ReferencesTotModels predOf tModel(2)  
 Categories(2) predOf tModel(1)  
  
 tModel(2) predOf SpecsStoredAtTheProvidersSite

**Interaktion zwischen UDDI-Registries (Grafik Seite 181):**

ServiceRequestor ∈ **Object**  
 ServiceProvider ∈ **Object**  
 UDDIRegistryA ∈ **Activity**  
 UDDIRegistryB ∈ **Activity**  
 WebServiceInterface(1) ∈ **Object**  
 ServiceDescriptions(1) ∈ **Specification**  
 WebServiceInterface(2) ∈ **Object**  
 ServiceDescriptions(2) ∈ **Specification**  
 InquiryAPI(1) ∈ **Port**  
 PublishersAPI(1) ∈ **Port**  
 InquiryAPI(2) ∈ **Port**  
 PublishersAPI(2) ∈ **Port**

**Relationen:**

numberOf ServiceDescriptions(1)  
 numberOf ServiceDescriptions(2)  
 entryPoints (WebServiceInterface(1))  
 entryPoints (WebServiceInterface(2))  
  
 WebServiceInterface(1) partOf UDDIRegistryA  
 ServiceDescriptions(1) partOf UDDIRegistryA  
  
 WebServiceInterface(2) partOf UDDIRegistryB  
 ServiceDescriptions(2) partOf UDDIRegistryB  
  
 InquiryAPI(1) portOf WebServiceInterface(1)  
 PublishersAPI(1) portOf WebServiceInterface(1)  
  
 InquiryAPI(2) portOf WebServiceInterface(2)  
 PublishersAPI(2) portOf WebServiceInterface(2)  
  
 label ((ServiceRequestor predOf InquiryAPI(1)), SOAP/HTTP)  
 label ((ServiceProvider predOf PublishersAPI(1)), SOAP/HTTPS)  
 WebServiceInterface(1) comm ServiceDescriptions(1)

label((WebServiceInterface(1) comm WebServiceInterface(2)),  
SubscriptionReplicationAndCustodyTransferAPIs)  
WebServiceInterface(2) comm ServiceDescriptions(2)

**schematischer Zusammenhang zwischen UDDI und WSDL (Grafik Seite 183):**

ServiceRequestor ∈ **Activity**  
ServiceProvider ∈ **Activity**  
WSDLServiceDescriptions ∈ **Specification**  
InquiryAPI ∈ **Port**  
PublishersAPI ∈ **Port**  
WebServiceInterface ∈ **Object**  
ServiceDescriptions ∈ **Specification**  
tModel ∈ **Activity**  
businessEntity ∈ **Activity**  
businessService ∈ **Activity**  
bindingTemplate ∈ **Activity**  
SDRefinement ∈ **Container**

→ für Grafik Seite 185

**Relationen:**

numberOf WSDLServiceDescriptions  
numberOf ServiceDescriptions  
numberOf bindingTemplate

WSDLServiceDescriptions partOf ServiceProvider  
InquiryAPI portOf WebServiceInterface  
PublishersAPI portOf WebServiceInterface  
WebServiceInterface partOf UDDIRegistry  
ServiceDescriptions partOf UDDIRegistry  
bindingTemplate partOf businessService  
businessService partOf businessEntity  
businessEntity partOf SDRefinement  
tModel partOf SDRefinement

→ für Grafik Seite 185

→ für Grafik Seite 185

SDRefinement refines ServiceDescriptions

label( (ServiceRequestor predOf InquiryAPI), SOAP/HTTP)  
label( (ServiceProvider predOf PublishersAPI), SOAP/HTTPS)  
WebServiceInterface comm ServiceDescriptions  
bindingTemplate predOf tModel  
tModel predOf WSDLServiceDescriptions

**Anbieten eines internen Services als Webservice (Grafik Seite 185):**

ServiceProvider ∈ **Activity**  
ServiceImplementation ∈ **Object**  
WSDLGenerator ∈ **Object**  
ServerStub ∈ **Object**  
SOAPRouter ∈ **Object**  
HTTPEngine ∈ **Object**  
WSDLServiceDescriptions ∈ **Specification**  
WSDLCompiler ∈ **Object**

UDDIPublisher ∈ **Object**  
SDRefinement ∈ **Container**  
UDDIRegistry ∈ **Activity**  
InquiryAPI ∈ **Port**  
PublishersAPI ∈ **Port**

→ aus Grafik Seite 183

**Relationen:**

InquiryAPI portOf UDDIRegistry  
PublishersAPI portOf UDDIRegistry

ServiceImplementation partOf ServiceProvider  
WSDLGenerator partOf ServiceProvider  
ServerStub partOf ServiceProvider  
SOAPRouter partOf ServiceProvider  
HTTPEngine partOf ServiceProvider  
WSDLServiceDescriptions partOf ServiceProvider  
WSDLCompiler partOf ServiceProvider  
UDDIPublisher partOf ServiceProvider

HTTPEngine predOf SOAPRouter  
SOAPRouter predOf ServerStub  
ServerStub predOf ServiceImplementation

ServiceImplementation allows WSDLGenerator  
WSDLGenerator allows WSDLServiceDescriptions  
WSDLServiceDescriptions allows WSDLCompiler  
WSDLServiceDescriptions allows UDDIPublisher  
tModel predOf WSDLServiceDescriptions  
WSDLCompiler allows ServerStub  
WSDLCompiler allows SOAPRouter

sendVia (UDDIPublisher, {SDRefinement}, PublishersAPI)

## Kapitel 7

### Kommunikation zwischen einem Client und einem Webservice (Grafik Seite 198)\*:

1) statisch:

Customer(Client) ∈ **Object**  
Supplier(WebService) ∈ **Object**

**Relationen:**

entryPoints (Supplier(WebService))

invokeOp (Customer(Client), 1:requestQuote, Supplier(WebService))  
invokeOp (Customer(Client), 2:orderGoods, Supplier(WebService))  
invokeOp (Customer(Client), 3:makePayment, Supplier(WebService))

2) dynamisch:

Variablen:

Quote ∈ Message  
Goods ∈ Message

SupplierMsg ∈ Message  
GoodsOrdered ∈ Boolean  
GoodsPaid ∈ Boolean

Funktionen:

RequestQuote: Message x Message → Quote  
OrderGoods: Message x Message → Boolean  
MakePayment: Message x Message → Boolean

ASM:

SampleConversation =<sub>def</sub>  
ClientSC

ClientSC =<sub>def</sub>  
**If** Goods != **undef** **and** SupplierMsg != **undef** **and** Quote = **undef**  
Quote := RequestQuote(Goods, SupplierMsg)  
**If** Quote != **undef** **and** !GoodsOrdered  
GoodsOrdered := OrderGoods(Goods, SupplierMsg)  
**If** GoodsOrdered **and** !GoodsPaid  
GoodsPaid := MakePayment(Goods, SupplierMsg)

### Zustandsgraph einer Konversations-Spezifikation (Grafik Seite 201):

-(1) ∈ **Point**  
RequestQuote ∈ **Activity**  
QuoteRequested ∈ **Object**  
OrderGoods ∈ **Activity**  
GoodsOrdered ∈ **Object**  
cancelOrder ∈ **Activity**  
makePayments ∈ **Activity**  
OrderCanceled ∈ **Object**  
OrderCompleted ∈ **Object**  
-(2) ∈ **Point**  
-(3) ∈ **Point**

**Relationen:**

-(1) predOf RequestQuote  
RequestQuote predOf QuoteRequested  
QuoteRequested predOf OrderGoods  
OrderGoods predOf GoodsOrdered  
GoodsOrdered predOf cancelOrder  
GoodsOrdered predOf makePayments  
cancelOrder predOf OrderCanceled  
OrderCanceled predOf -(2)  
makePayments predOf OrderCompleted  
OrderCompleted predOf -(3)

### Bestellvorgang zwischen zwei WebServices (Grafik Seite 202 (7.3))\*:

1) statisch:

Customer ∈ **Object**  
Supplier ∈ **Object**

**Relationen:**

entryPoints (Customer)  
 entryPoints (Supplier)  
 invokeOp (Customer, 1:requestQuote, Supplier)  
 invokeOp (Customer, 2:orderGoods, Supplier)  
 invokeOp (Supplier, 3:confirmOrder, Customer)  
 invokeOp (Customer, 4:makePayment, Supplier)

2) dynamisch:

Variablen:

Quote  $\in$  Message  
 Goods  $\in$  Message  
 CustomerMsg  $\in$  Message  
 SupplierMsg  $\in$  Message  
 GoodsOrdered  $\in$  Boolean  
 OrderConfirmed  $\in$  Boolean  
 GoodsPaid  $\in$  Boolean

Funktionen:

RequestQuote: Message x Message  $\rightarrow$  Message  
 OrderGoods: Message x Message  $\rightarrow$  Boolean  
 MakePayment: Message x Message  $\rightarrow$  Boolean  
 ConfirmOrder: Message x Message  $\rightarrow$  Boolean

ASM:

ProcurementConversation  $=_{def}$   
 SampleConversation  
 SupplierPC

SupplierPC  $=_{def}$   
**If** (Goods  $\neq$  **undef**) **and** (CustomerMsg  $\neq$  **undef**) **and** (GoodsOrdered)  
 OrderConfirmed := ConfirmOrder(Goods, CustomerMsg)

**Allgemeines Protokoll für Bestellvorgänge (Grafik Seite 202 (7.4))\*:**1) statisch:

Customer  $\in$  **Object**  
 Supplier  $\in$  **Object**  
 Warehouse  $\in$  **Object**

**Relationen:**

entryPoints (Customer)  
 entryPoints (Supplier)  
 invokeOp (Customer, 1:requestQuote, Supplier)  
 invokeOp (Customer, 2:orderGoods, Supplier)  
 invokeOp (Supplier, 3:checkShipAvailable, Warehouse)  
 invokeOp (Supplier, 4:confirmOrder, Customer)  
 invokeOp (Customer, 5:makePayment, Supplier)  
 invokeOp (Supplier, 6:orderShipment, Warehouse)  
 invokeOp (Warehouse, 7:getShipmentDetail, Customer)  
 invokeOp (Customer, 8:confirmShipment, Warehouse)

invokeOp (Warehouse, 9:checkShipAvailable, Supplier)

## 2) dynamisch:

Variablen:

ProcurementConversation  
CustomerMsg ∈ Messages  
WarehouseMsg ∈ Messages  
OrderConfirmed ∈ Boolean  
ShippingAvail ∈ Boolean  
ShipmentArranged ∈ Boolean  
ShipmentConfirmed ∈ Boolean  
ShipmentOrdered ∈ Boolean  
WarehouseConfirmedShipment ∈ Boolean

Funktionen:

ProcurementConversation  
CheckShipAvailable: Message x Message x Message → Boolean  
OrderShipment: Message x Message x Message → Boolean  
ConfirmShipment: Message x Message x Message → Boolean  
GetShipmentDetail: Message x Message x Message → Boolean

ASM:

ProcurementProtocol =<sub>def</sub>

**if** WarehouseConfirmedShipment != **true**

CustomerPP

SupplierPP

WarehousePP

CustomerPP =<sub>def</sub>

SampleConversation

**If** ShipmentArranged = **true**

ShipmentConfirmed := ConfirmShipment(CustomerMsg, Goods,  
WarehouseMsg)

SupplierPP =<sub>def</sub>

**If** !GoodsOrdered

GoodsPaid := **true**

**If** GoodsOrdered

ShippingAvail := CheckShipAvailable(SupplierMsg, Goods, WarehouseMsg)

**If** ShippingAvail **and** OrderConfirmed != **true**

OrderConfirmed := ConfirmOrder(Quote, CustomerMsg)

GoodsPaid := !ConfirmOrder(Quote, CustomerMsg)

**If** GoodsPaid **and** ShipmentOrdered != **true**

ShipmentOrdered := OrderShipment(SupplierMsg, Goods, WarehouseMsg)

WarehousePP =<sub>def</sub>

**If** ShipmentOrdered

ShipmentArranged := GetShipmentDetail (WarehouseMsg, Goods,  
CustomerMsg)

**If** ShipmentConfirmed

WarehouseConfirmedShipment := ConfirmShipment(WarehouseMsg,  
Goods, SupplierMsg)

### **Das Bestell-Protokoll als Sequenzdiagramm (Grafik Seite 204):**

Entspricht der Darstellung der Grafik auf Seite 202(7.4), da strukturell keine Unterschiede bestehen – nur eine andere Art der graphischen Darstellung

### **Das Bestell-Protokoll als Aktivitätsdiagramm (Grafik Seite 205)\*:**

1) statisch:

-(1) ∈ **Point**

-(2) ∈ **Point**

-(3) ∈ **Point**

Customer ∈ **Container**

RequestQuote(ToSupplier) ∈ **Object**

OrderGoods(ToSupplier) ∈ **Object**

MakePayment(ToSupplier) ∈ **Object**

ConfirmShipment(ToWarehouse) ∈ **Object**

Supplier ∈ **Container**

CheckShipAvailable(ToWarehouse) ∈ **Object**

ConfirmOrder(ToCustomer) ∈ **Object**

CancelOrder(ToCustomer) ∈ **Object**

OrderShipment(ToWarehouse) ∈ **Object**

Warehouse ∈ **Container**

GetShipmentDetails(ToCustomer) ∈ **Object**

ConfirmShipment(ToSupplier) ∈ **Object**

WarehouseConfirms ∈ **Object**

#### **Relationen:**

RequestQuote(ToSupplier) partOf Customer

OrderGoods(ToSupplier) partOf Customer

MakePayment(ToSupplier) partOf Customer

ConfirmShipment(ToWarehouse) partOf Customer

CheckShipAvailable(ToWarehouse) partOf Supplier

ConfirmOrder(ToCustomer) partOf Supplier

CancelOrder(ToCustomer) partOf Supplier

OrderShipment(ToWarehouse) partOf Supplier

GetShipmentDetails(ToCustomer) partOf Warehouse

ConfirmShipment(ToSupplier) partOf Warehouse

-(1) predOf RequestQuote(ToSupplier)

RequestQuote(ToSupplier) predOf OrderGoods(ToSupplier)

OrderGoods(ToSupplier) predOf CheckShipAvailable(ToWarehouse)

OrStatement

(CheckShipAvailable(ToWarehouse), WarehouseConfirms, ConfirmOrder(ToCustomer),  
CancelOrder(ToCustomer))

CancelOrder(ToCustomer) predOf -(2)

ConfirmOrder(ToCustomer) predOf MakePayment(ToSupplier)

MakePayment(ToSupplier) predOf OrderShipment(ToWarehouse)

OrderShipment(ToWarehouse) predOf GetShipmentDetails(ToCustomer)

GetShipmentDetails(ToCustomer) predOf ConfirmShipment(ToWarehouse)

ConfirmShipment(ToWarehouse) predOf ConfirmShipment(ToSupplier)

ConfirmShipment(ToSupplier) predOf -(3)

2) dynamisch:

Variablen:

ProcurementProtocol  
OrderCancelled ∈ Boolean

Funktionen:

ProcurementProtocol  
CancelOrder: Message x Message x Message → Boolean

ASM:

```
NewProcurementProtocol =def
  If OrderCancelled != true
  and WarehouseConfirmedShipment != true
  CustomerPP
  NewSupplier
  WarehousePP

NewSupplier =def
  SupplierPP
  If ShippingAvail = false
  OrderCancelled := CancelOrder(Quote, CustomerMsg)
```

**Das Bestell-Protokoll aus Kundensicht (Grafik Seite 207)\*:**

1) statisch:

-(1) ∈ **Point**  
-(2) ∈ **Point**  
-(3) ∈ **Point**  
Customer ∈ **Container**  
RequestQuote(ToSupplier) ∈ **Object**  
OrderGoods(ToSupplier) ∈ **Object**  
MakePayment(ToSupplier) ∈ **Object**  
ConfirmShipment(ToWarehouse) ∈ **Object**  
Supplier ∈ **Container**  
ConfirmOrder(ToCustomer) ∈ **Object**  
CancelOrder(ToCustomer) ∈ **Object**  
Warehouse ∈ **Container**  
GetShipmentDetails(ToCustomer) ∈ **Object**  
WarehouseConfirms ∈ **Boolean**

**Relationen:**

RequestQuote(ToSupplier) partOf Customer  
OrderGoods(ToSupplier) partOf Customer  
MakePayment(ToSupplier) partOf Customer  
ConfirmShipment(ToWarehouse) partOf Customer

ConfirmOrder(ToCustomer) partOf Supplier  
CancelOrder(ToCustomer) partOf Supplier

GetShipmentDetails(ToCustomer) partOf Warehouse

-(1) predOf RequestQuote(ToSupplier)

RequestQuote(ToSupplier) predOf OrderGoods(ToSupplier)  
 OrStatement  
     (OrderGoods(ToSupplier), WarehouseConfirms, ConfirmOrder(ToCustomer),  
     CancelOrder(ToCustomer))  
 CancelOrder(ToCustomer) predOf -(2)  
 ConfirmOrder(ToCustomer) predOf MakePayment(ToSupplier)  
 MakePayment(ToSupplier) predOf GetShipmentDetails(ToCustomer)  
 GetShipmentDetails(ToCustomer) predOf ConfirmShipment(ToWarehouse)  
 ConfirmShipment(ToWarehouse) predOf -(3)

## 2) dynamisch:

Da der “customer view” auf das “Procurement Protocol” einen Teil des gesamten auf Seite 202 (7.4) vorgestellten Protokolles darstellt, können auch hier wieder Variablen, Funktionen und ASMs dieses Protokolles verwendet werden. Die in der ASM zur Grafik auf Seite 205 hinzugekommenen Änderungen behalten auch hier ihre Gültigkeit.

Vom “customer view”-Teilprotokoll benötigte Variablen:

ProcurementConversation  
 ShipmentDetail ∈ Message  
 ShipmentConfirmed ∈ Boolean  
 OrderCancelled ∈ Boolean

Funktionen:

ProcurementConversation  
 ConfirmShipment: Message x Message x Message → Boolean  
 GetShipmentDetail: Message x Message x Message → Message  
 CancelOrder: Message x Message x Message → Boolean

ASM:

ProcurementProtocolCV =<sub>def</sub>  
     **If** !OrderCancelled **and** !ShipmentConfirmed  
         CustomerPP  
         SupplierCV  
         WarehouseCV

SupplierCV =<sub>def</sub>  
     **If** ShippingAvail  
         OrderConfirmed := ConfirmOrder(Goods, CustomerMsg)  
     **If** !ShippingAvail  
         OrderCancelled := CancelOrder(Goods, CustomerMsg)

WarehouseCV =<sub>def</sub>  
     **If** ShipmentOrdered  
         ShipmentDetail := GetShipmentDetail (SupplierMsg, Goods, CustomerMsg)

## **Horizontale und vertikale Webservice-Protokolle (Grafik Seite 208):**

WebServices ∈ **Container**  
 ServiceProvider ∈ **Activity**  
 Manufacturing ∈ **Object**  
 HealthCare ∈ **Object**  
 Telecom ∈ **Object**  
 Finance ∈ **Object**  
 -(1) ∈ **undefObj**

Middleware ∈ **Activity**  
SupportForProtocols ∈ **Object**  
OtherWebServicesMiddleware ∈ **Object**  
Environment ∈ **Container**

**Relationen:**

Manufacturing partOf WebServices  
HealthCare partOf WebServices  
Telecom partOf WebServices  
Finance partOf WebServices  
-(1) partOf WebServices  
WebServices partOf ServiceProvider

SupportForProtocols partOf Middleware  
OtherWebServicesMiddleware partOf Middleware

label ((OtherWebServicesMiddleware comm Environment), SOAPMessages)  
OtherWebServicesMiddleware comm SupportForProtocols

Manufacturing comm Middleware  
HealthCare comm Middleware  
Telecom comm Middleware  
Finance comm Middleware  
-(1) comm Middleware

**Client – WebService Interaktion (Grafik Seite 209):**

ServiceProvider ∈ **Object**  
ServiceRequestor(1) ∈ **Object**  
ServiceRequestor(2) ∈ **Object**  
ServiceRequestor(3) ∈ **Object**

**Relationen:**

label ((ServiceProvider comm ServiceRequestor(1)), Protocol1)  
label ((ServiceProvider comm ServiceRequestor(2)), Protocols2&3)  
label ((ServiceProvider comm ServiceRequestor(3)), Protocols4&5)

**Aufgabe des “Conversation Controllers” (Grafik Seite 210):**

ServiceProvider ∈ **Activity**  
ConversationController ∈ **Object**  
ObjectForProtocol 1 ∈ **Object**  
ObjectForProtocol 2 ∈ **Object**  
ObjectForProtocol 3 ∈ **Object**  
ObjectForProtocol 4 ∈ **Object**  
ObjectForProtocol 5 ∈ **Object**  
ServiceRequestor(1) ∈ **Object**  
ServiceRequestor(2) ∈ **Object**  
ServiceRequestor(3) ∈ **Object**

**Relationen:**

ConversationController partOf ServiceProvider

ObjectForProtocol 1 partOf ServiceProvider  
ObjectForProtocol 2 partOf ServiceProvider  
ObjectForProtocol 3 partOf ServiceProvider  
ObjectForProtocol 4 partOf ServiceProvider  
ObjectForProtocol 5 partOf ServiceProvider

label ((ConversationController comm ServiceRequestor(1)), Protocol1)  
label ((ConversationController comm ServiceRequestor(2)), Protocols2&3)  
label ((ConversationController comm ServiceRequestor(3)), Protocols4&5)

label ((ConversationController comm ObjectForProtocol1), Protocol1)  
label ((ConversationController comm ObjectForProtocol2), Protocol2)  
label ((ConversationController comm ObjectForProtocol3), Protocol3)  
label ((ConversationController comm ObjectForProtocol4), Protocol4)  
label ((ConversationController comm ObjectForProtocol5), Protocol5)

### **“Conversation Controller” als Teil eines SOAP-Routers (Grafik Seite 211):**

ServiceProvider ∈ **Activity**  
EJB(1) ∈ **Object**  
EJB(2) ∈ **Object**  
EJB(3) ∈ **Object**  
EJBContainer ∈ **Activity**  
SOAPRouter(WithConversationController) ∈ **Object**  
ConversationID/ObjectMapping ∈ **Resource**  
HTTPServer ∈ **Object**  
SOAPMessagesOnHTTPTransport ∈ **Container**

#### **Relationen:**

EJB(1) partOf EJBContainer  
EJB(2) partOf EJBContainer  
EJB(3) partOf EJBContainer  
EJBContainer partOf ServiceProvider  
SOAPRouter(WithConversationController) partOf ServiceProvider  
ConversationID/ObjectMapping partOf ServiceProvider  
HTTPServer partOf ServiceProvider

HTTPServer predOf SOAPRouter(WithConversationController)  
ConversationID/ObjectMapping predOf SOAPRouter(WithConversationController)  
SOAPRouter(WithConversationController) predOf EJB(2)  
SOAPMessagesOnHTTPTransport predOf HTTPServer

### **Infrastruktur zur Implementierung horizontaler Protokolle (Grafik Seite 214):**

ServiceProvider ∈ **Activity**  
Object(WebServiceImplementation) ∈ **Object**  
HorizontalProtocolImplementation ∈ **Object**  
ConversationRouting&ComplianceVerification ∈ **Object**  
ServiceRequestor ∈ **Object**

#### **Relationen:**

numberOf Object(WebServiceImplementation)

numberOf HorizontalProtocolImplementation

Object(WebServiceImplementation) partOf ServiceProvider

HorizontalProtocolImplementation partOf ServiceProvider

ConversationRouting&ComplianceVerification partOf ServiceProvider

label ((Object(WebServiceImplementation) comm HorizontalProtocolImplementation),  
HorizontalProtocolConversation)

label ((Object(WebServiceImplementation) comm ConversationRouting&ComplianceVerification),  
BusinessProtocolConversation)

label ((HorizontalProtocolImplementation comm ConversationRouting&ComplianceVerification),  
HorizontalProtocolConversation)

label ((ServiceRequestor comm ConversationRouting&ComplianceVerification),  
HorizontalProtocolConversation)

label ((ServiceRequestor comm ConversationRouting&ComplianceVerification),  
BusinessProtocolConversation)

### **Kommunikation von Port-Referenzen und Rolleninformationen (Grafik Seite 215):**

Object(W1) ∈ **Object**

Object(W2) ∈ **Object**

HorizontalProtocolHandler(A) ∈ **Object**

HorizontalProtocolHandler(B) ∈ **Object**

ConversationController(1) ∈ **Object**

ConversationController(2) ∈ **Object**

#### **Relationen:**

label((ConversationController(1) comm ConversationController(2)), ProtocolMessages)

label((ConversationController(1) comm HorizontalProtocolHandler(A)), ProtocolMessages)

label((ConversationController(2) comm HorizontalProtocolHandler(B)), ProtocolMessages)

label((Object(W1) predOf Object(W2)), A'sPortReference)

label((Object(W1) predOf HorizontalProtocolHandler(A), B'sPortReference)

label((Object(W1) predOf HorizontalProtocolHandler(A), A'sRole)

label((Object(W2) predOf Object(W1), B'sPortReference)

label((Object(W2) predOf HorizontalProtocolHandler(B), A'sPortReference)

label((Object(W2) predOf HorizontalProtocolHandler(B), A'sRole)

### **Zentralisierte WS-Koordination (Grafik Seite 216):**

WebService(1) ∈ **Object**

WebService(2) ∈ **Object**

WebService(3) ∈ **Object**

Coordinator ∈ **Object**

#### **Relationen:**

WebService(1) comm WebService(2)

WebService(2) comm WebService(3)

WebService(1) comm Coordinator

WebService(2) comm Coordinator

WebService(3) comm Coordinator

**Verteilte WS-Koordination (Grafik Seite 216):**

WebService(1) ∈ **Object**  
WebService(2) ∈ **Object**  
WebService(3) ∈ **Object**  
Coordinator(1) ∈ **Object**  
Coordinator(2) ∈ **Object**  
Coordinator(3) ∈ **Object**

**Relationen:**

WebService(1) comm WebService(2)  
WebService(2) comm WebService(3)

Coordinator(1) comm Coordinator(2)  
Coordinator(2) comm Coordinator(3)

WebService(1) comm Coordinator(1)  
WebService(2) comm Coordinator(2)  
WebService(3) comm Coordinator(3)

**Erstellung / Aktivierung eines Koordinationskontextes (Grafik Seite 218):**

WebService ∈ **Object**  
Coordinator ∈ **Object**  
CreateCoordinationContext ∈ **Activity**  
CreateCoordinationContextResponse ∈ **Activity**  
-(1) ∈ **undefObj**  
CoordinationType(1) ∈ **Object**  
CurrentContext ∈ **Object**  
-(2) ∈ **undefObj**  
CoordinationContext ∈ **Object**  
Identifier ∈ **Object**  
CoordinationType(2) ∈ **Object**  
RegistrationService ∈ **Object**  
-(3) ∈ **undefObj**  
ActivationRequestorPortType ∈ **Port**  
ActivationCoordinatorPortType ∈ **Port**

**Relationen:**

ActivationRequestorPortType portOf WebService  
ActivationCoordinatorPortType portOf Coordinator

CoordinationType(1) partOf CreateCoordinationContext  
CurrentContext partOf CreateCoordinationContext  
-(1) partOf CreateCoordinationContext

Identifier partOf CoordinationContext  
CoordinationType(2) partOf CoordinationContext  
RegistrationService partOf CoordinationContext  
-(3) partOf CoordinationContext

CoordinationContext partOf CreateCoordinationContextresponse

-(2) partOf CreateCoordinationContextresponse

WebService predOf CreateCoordinationContext

CreateCoordinationContext predOf ActivationCoordinatorPortType

Coordinator predOf CreateCoordinationContextResponse

CreateCoordinationContextResponse predOf ActivationRequestorPortType

### **WS-Registrierung bei einem Koordinator (Grafik Seite 219):**

WebService ∈ **Object**

Coordinator ∈ **Object**

Register ∈ **Activity**

-(1) ∈ **undefObj**

ProtocolIdentifier ∈ **Object**

ParticipantProtocolService ∈ **Object**

RegisterResponse ∈ **Activity**

-(2) ∈ **undefObj**

CoordinatorProtocolService ∈ **Object**

RegistrationRequestorPortType ∈ **Port**

RegistrationCoordinatorPortType ∈ **Port**

#### **Relationen:**

RegistrationRequestorPortType portOf WebService

RegistrationCoordinatorPortType portOf Coordinator

ProtocolIdentifier partOf Register

ParticipantProtocolService partOf Register

-(1) partOf Register

CoordinatorProtocolService partOf RegisterResponse

-(2) partOf RegisterResponse

WebService predOf Register

Register predOf RegistrationCoordinatorPortType

Coordinator predOf RegisterResponse

RegisterResponse predOf RegistrationRequestorPortType

### **Konvention zur Definition von protokoll-spezifischen Schnittstellen (Grafik Seite 220):**

WebService ∈ **Object**

Coordinator ∈ **Object**

XParticipantPortType ∈ **Port**

XCoordinatorPortType ∈ **Port**

#### **Relationen:**

label((WebService predOf XCoordinatorPortType), Protocol-SpecificMessages)

label((Coordinator predOf XCoordinatorPortType), Protocol-SpecificMessages)

## **Nachrichtenaustausch über einen zentralen Koordinator (Grafik Seite 221)\*:**

### 1) statisch:

WebServiceA ∈ **Object**  
ActivationParticipant(1) ∈ **Object**  
RegistrationParticipant(1) ∈ **Object**  
ProtocolParticipant(1) ∈ **Object**  
WebServiceImplementation(1) ∈ **Object**  
CoordinatorC ∈ **Object**  
ActivationCoordinator ∈ **Object**  
RegistrationCoordinator ∈ **Object**  
ProtocolCoordinator ∈ **Object**  
WebServiceB ∈ **Object**  
WebServiceImplementation(2) ∈ **Object**  
ActivationParticipant(2) ∈ **Object**  
RegistrationParticipant(2) ∈ **Object**  
ProtocolParticipant(2) ∈ **Object**

### **Relationen:**

entryPoints(ActivationParticipant(1))  
entryPoints(RegistrationParticipant(1))  
entryPoints(ProtocolParticipant(1))  
entryPoints(WebServiceImplementation(1))  
entryPoints(ActivationCoordinator)  
entryPoints(RegistrationCoordinator)  
entryPoints(ProtocolCoordinator)  
entryPoints(ActivationParticipant(2))  
entryPoints(RegistrationParticipant(2))  
entryPoints(ProtocolParticipant(2))  
entryPoints(WebServiceImplementation(2))

ActivationParticipant(1) partOf WebServiceA  
RegistrationParticipant(1) partOf WebServiceA  
ProtocolParticipant(1) partOf WebServiceA  
WebServiceImplementation(1) partOf WebServiceA

ActivationCoordinator partOf CoordinatorC  
RegistrationCoordinator partOf CoordinatorC  
ProtocolCoordinator partOf CoordinatorC

ActivationParticipant(2) partOf WebServiceB  
RegistrationParticipant(2) partOf WebServiceB  
ProtocolParticipant(2) partOf WebServiceB  
WebServiceImplementation(2) partOf WebServiceB

invokeOp (ActivationParticipant(1), 1:CreateCC, ActivationCoordinator)  
invokeOp (ActivationCoordinator, 2:X1, ActivationParticipant(1))  
invokeOp (RegistrationParticipant(1), 3:Register, RegistrationCoordinator)  
invokeOp (RegistrationCoordinator, 4:ProtocolCoordinator, RegistrationParticipant(1))  
invokeOp (WebServiceImplementation(1), 5:OperationalMessage, WebServiceImplementation(2))  
invokeOp (RegistrationParticipant(2), 6:Register, RegistrationCoordinator)  
invokeOp (RegistrationCoordinator, 7:ProtocolCoordinator, RegistrationParticipant(2))  
sendMsg (ProtocolParticipant(1), 8:Protocol-SpecificMessage, ProtocolCoordinator)  
sendMsg (ProtocolCoordinator, 9:Protocol-SpecificMessage, ProtocolParticipant(2))

## 2) dynamisch:

### Variablen:

Coordinator  $\in$  Coordinators  
WebService  $\in$  Services  
XMsg  $\in$  Message  
CreateMsg  $\in$  Message  
OPMsg  $\in$  OPMessages  
ProtocolSpecificMsg  $\in$  PSMessages  
ProtocolCoordinator  $\in$  Coordinators  
OPMsgReceived  $\in$  Boolean  
Self  $\in$  Services  $\rightarrow$  repräsentiert in der jeweiligen ASM den Webservice/Coordinator, den die ASM darstellt

### Funktionen:

StartConversation: Services  $\rightarrow$  Boolean  
CreateCC:  $\rightarrow$  Message  
SendCC: Message  $\rightarrow$  Message  
Register: Message  $\rightarrow$  Boolean  
SendPC: Services  $\rightarrow$  Coordinators  
OperationalMsg: Messages x Services  $\rightarrow$  Boolean  
ProtocolMsgC: PSMessages x Coordinators  $\rightarrow$  Boolean  
ProtocolMsgS: PSMessages x Services  $\rightarrow$  Boolean  
Registered: Services  $\rightarrow$  Boolean

### ASM:

ConversationExecution =<sub>def</sub>  
    WebService(A)  
    CoordinatorC  
    WebService(B)

WebService(Self) =<sub>def</sub>  
    ActivationParticipant(Self)  
    RegistrationParticipant(Self)  
    ProtocolParticipant(Self)  
    WebServiceImplementation(Self)

CoordinatorC =<sub>def</sub>  
    ActivationCoordinator  
    RegistrationCoordinator  
    ProtocolCoordinator

ActivationParticipant(Self) =<sub>def</sub>  
    **If** StartConversation(Self)  
        CreateMsg := CreateCC(Coordinator)

RegistrationParticipant(Self) =<sub>def</sub>  
    **If** XMsg != **undef** or OPMsgReceived  
        Registered(Self) := Register(Coordinator)

WebServiceImplementation(Self) =<sub>def</sub>  
    **If** ProtocolCoordinator(Self) != **undef** and StartConversation(Self)

```

        choose Webservice ∈ Services and OPMsg ∈ OPMessages
            OperationalMsg(OPMsg, Webservice) := true
    for all OPMsg in OPMessages
        If OperationalMsg(OPMsg, Self) = true
            OPMsgReceived := true

ProtocolParticipant(Self) =def
    If StartConversation(Self)
        choose ProtocolSpecificMsg ∈ PSMessages
            ProtocolMsgC(ProtocolSpecificMsg, Coordinator) := true

ActivationCoordinator =def
    If CreateMsg != undef
        XMsg := SendCC(WebService)

RegistrationCoordinator =def
    for all Webservice in Services
        If Registered(WebService) = true
            ProtocolCoordinator(WebService) := SendPC(WebService)

ProtocolCoordinator =def
    for all PSMsg in PSMessages
        If ProtocolMsg(PSMsg, Self)
            choose Webservice ∈ Services
                ProtocolMsgS(PSMsg, Webservice) := true

```

### **Nachrichtenaustausch bei verteilter Koordination (Grafik Seite 222)\*:**

1) statisch:

```

WebServiceA ∈ Object
CoordinatorCa ∈ Object
WebServiceB ∈ Object
CoordinatorCb ∈ Object

```

#### **Relationen:**

```

invokeOp (WebServiceA, 1:CreateCC, CoordinatorCa)
invokeOp (CoordinatorCa, 2:X1, WebServiceA)
invokeOp (WebServiceA, 3:Register, CoordinatorCa)
invokeOp (CoordinatorCa, 4:ProtocolCoordinator, WebServiceA)

invokeOp (WebServiceA, 5:OperationalMessage, WebServiceB)

invokeOp (WebServiceB, 6:CreateCC, CoordinatorCb)
invokeOp (CoordinatorCb, 7:X2, WebServiceB)
invokeOp (WebServiceB, 8:Register, CoordinatorCb)
invokeOp (CoordinatorCb, 9:Register, CoordinatorCa)
invokeOp (CoordinatorCa, 10:ProtocolCoordinator, CoordinatorCb)
invokeOp (CoordinatorCb, 11: ProtocolCoordinator, WebServiceB)

sendMsg (WebServiceA, 12:ProtocolMessage, CoordinatorCa)
sendMsg (CoordinatorCa, 13:ProtocolMessage, CoordinatorCb)
sendMsg (CoordinatorCa, 14:ProtocolMessage, WebServiceA)
sendMsg (CoordinatorCb, 15:ProtocolMessage, WebServiceB)

```

## 2) dynamisch:

### Variablen:

C-A  $\in$  Coordinators  
C-B  $\in$  Coordinators  
WS-A  $\in$  Services  
WS-B  $\in$  Services  
CreateMsg  $\in$  Message  
Registered  $\in$  Boolean  
OPMsg  $\in$  OPMessages  
ProtocolSpecificMsg  $\in$  PSMessages  
Self  $\in$  Services  $\rightarrow$  repräsentiert in der jeweiligen ASM den Webservice/Coordinator,  
den die ASM darstellt

### Funktionen:

StartConversation: Services  $\rightarrow$  Boolean  
CreateCC: Coordinators x Services  $\rightarrow$  Boolean  
SendCC: Message  $\rightarrow$  Message  
Register: Message  $\rightarrow$  Boolean  
RegisterC: Coordinators x Coordinators  $\rightarrow$  Boolean  
RegisteredC: Coordinators  $\rightarrow$  Boolean  
SendPC: Services  $\rightarrow$  Coordinators  
OperationalMsg: Messages x Services  $\rightarrow$  Boolean  
ProtocolMsgC: PSMessages x Coordinators  $\rightarrow$  Boolean  
ProtocolMsgS: PSMessages x Services  $\rightarrow$  Boolean  
ProtocolCoordinatorC: Coordinators  $\rightarrow$  Boolean  
OPMsgReceived: Services  $\rightarrow$  Boolean  
XMsg: Services  $\rightarrow$  Message  
ProtocolCoordinator: Services  $\rightarrow$  Message  
WSCoordinator: Services  $\rightarrow$  Coordinators

### ASM:

ConversationExecution  $\stackrel{=def}{=}$   
    WebService(WS-A)  
    Coordinator(C-A)  
    WebService(WS-B)  
    Coordinator(C-B)

WebService(Self)  $\stackrel{=def}{=}$   
    **If** StartConversation(Self) **or** OPMsgReceived(Self)  
        CreateCC(WSCoordinator(Self), Self) := **true**  
    **If** XMsg(Self) != **undef**  
        Registered(Self) := Register(WSCoordinator(Self))  
    **If** ProtocolCoordinator(Self) != **undef and** StartConversation(Self)  
        **choose** Webservice  $\in$  Services **and** OPMsg  $\in$  OPMessages  
            OperationalMsg(OPMsg, Webservice) := **true**  
    **If** StartConversation(Self) **and** !OPMsgReceived(Self)  
    **and** ProtocolCoordinator(Self) != **undef**  
        **choose** ProtocolSpecificMsg  $\in$  PSMessages  
            ProtocolMsgC(ProtocolSpecificMsg, WSCoordinator(Self)) := **true**  
    **for all** OPMsg **in** OPMessages  
        **If** OperationalMsg(OPMsg, Self)  
            OPMsgReceived(Self) := **true**

Coordinator(Self)  $\stackrel{=def}{=}$   
    **for all** Webservice **in** Services

```

If CreateCC(Self, WebService)
    XMsg(WebService) := SendCC(WebService)
If Registered(WebService) and !OPMsgReceived(Self)
    ProtocolCoordinator(WebService) := SendPC(WebService)
If Registered(WebService) and OPMsgReceived(WebService)
    choose coord ∈ Coordinators
        RegisterC(coord, Self) := true
    If RegisteredC(Self)
        ProtocolCoordinator(WebService) := SendPC(WebService)
If ProtocolCoordinator(WebService) != undef
and OPMsgReceived(WebService)
    choose ProtocolMsg ∈ PSMessages and coord ∈ Coordinators
        ProtocolMsgC(ProtocolMsg, coord) := true
for all PSMsg in PSMessages
    If ProtocolMsgC(PSMsg, Self)
        choose WebService ∈ Services
            ProtocolMsgS(PSMsg, WebService) := true
for all coord in Coordinators
    If RegisterC(Self, coord)
        RegisteredC(coord) := ProtocolCoordinatorC(coord)

```

**Verknüpfung von Koordinatoren bei verteilter Koordination (Grafik Seite 224):**

CoordinatorN ∈ **Object**  
 CoordinatorN+1 ∈ **Object**  
 WebServiceN+1 ∈ **Object**  
 XParticipantPortType(1) ∈ **Port**  
 XParticipantPortType(2) ∈ **Port**  
 XParticipantPortType(3) ∈ **Port**  
 XCoordinatorPortType(1) ∈ **Port**  
 XCoordinatorPortType(2) ∈ **Port**  
 FurtherCoordinators ∈ **Container**

**Relationen:**

XParticipantPortType(1) portOf CoordinatorN  
 XParticipantPortType(2) portOf CoordinatorN+1  
 XParticipantPortType(3) portOf WebServiceN+1  
 XCoordinatorPortType(1) portOf CoordinatorN+1  
 XCoordinatorPortType(2) portOf CoordinatorN

FurtherCoordinators allows XParticipantPortType(1)  
 CoordinatorN allows XParticipantPortType(2)  
 CoordinatorN+1 allows XParticipantPortType(3)

WebServiceN+1 coord XCoordinatorPortType(1)  
 CoordinatorN+1 coord XCoordinatorPortType(2)  
 CoordinatorN coord FurtherCoordinators

**Port-Typen bei atomaren Transaktionen (Grafik Seite 229):**

WS-CoordinationInterfaces ∈ **Container**  
 ActivationCoordinatorPortType ∈ **Port**  
 RegistrationCoordinatorPortType ∈ **Port**

WS-TransactionInterfacesNeededForChaining ∈ **Container**  
 CompletionParticipantPortType ∈ **Port**  
 CompletionWithAckParticipantPortType ∈ **Port**  
 PhaseZeroParticipantPortType ∈ **Port**  
 2PCParticipantPortType ∈ **Port**  
 OutcomeNotificationsParticipantPortType ∈ **Port**  
 WS-CoordinationInterfacesNeededForChaining ∈ **Container**  
 RegistrationParticipantPortType ∈ **Port**  
 WS-TransactionInterfaces ∈ **Container**  
 CompletionCoordinatorPortType ∈ **Port**  
 CompletionWithAckCoordinatorPortType ∈ **Port**  
 PhaseZeroCoordinatorPortType ∈ **Port**  
 2PCCoordinatorPortType ∈ **Port**  
 OutcomeNotificationCoordinatorPortType ∈ **Port**  
 AtomicTransactionCoordinator ∈ **Object**

**Relationen:**

ActivationCoordinatorPortType portOf AtomicTransactionCoordinator  
 RegistrationCoordinatorPortType portOf AtomicTransactionCoordinator  
 CompletionParticipantPortType portOf AtomicTransactionCoordinator  
 CompletionWithAckParticipantPortType portOf AtomicTransactionCoordinator  
 PhaseZeroParticipantPortType portOf AtomicTransactionCoordinator  
 2PCParticipantPortType portOf AtomicTransactionCoordinator  
 OutcomeNotificationsParticipantPortType portOf AtomicTransactionCoordinator  
 RegistrationParticipantPortType portOf AtomicTransactionCoordinator  
 CompletionCoordinatorPortType portOf AtomicTransactionCoordinator  
 CompletionWithAckCoordinatorPortType portOf AtomicTransactionCoordinator  
 PhaseZeroCoordinatorPortType portOf AtomicTransactionCoordinator  
 2PCCoordinatorPortType portOf AtomicTransactionCoordinator  
 OutcomeNotificationCoordinatorPortType portOf AtomicTransactionCoordinator

ActivationCoordinatorPortType partOf WS-CoordinationInterfaces  
 RegistrationCoordinatorPortType partOf WS-CoordinationInterfaces

CompletionParticipantPortType partOf WS-TransactionInterfacesNeededForChaining  
 CompletionWithAckParticipantPortType partOf WS-TransactionInterfacesNeededForChaining  
 PhaseZeroParticipantPortType partOf WS-TransactionInterfacesNeededForChaining  
 2PCParticipantPortType partOf WS-TransactionInterfacesNeededForChaining  
 OutcomeNotificationsParticipantPortType partOf WS-TransactionInterfacesNeededForChaining

RegistrationParticipantPortType partOf WS-CoordinationInterfacesNeededForChaining

CompletionCoordinatorPortType partOf WS-TransactionInterfaces  
 CompletionWithAckCoordinatorPortType partOf WS-TransactionInterfaces  
 PhaseZeroCoordinatorPortType partOf WS-TransactionInterfaces  
 2PCCoordinatorPortType partOf WS-TransactionInterfaces  
 OutcomeNotificationCoordinatorPortType partOf WS-TransactionInterfaces

**Beispiel für atomare Transaktionen (Grafik Seite 231)\*:**

1) statisch:

WebServiceA ∈ **Object**  
 CoordinatorCa ∈ **Object**

WebServiceB ∈ **Object**  
CoordinatorCb ∈ **Object**

**Relationen:**

invokeOp (WebServiceA, CreateCC, CoordinatorCa)  
invokeOp (CoordinatorCa, T1, WebServiceA)  
invokeOp (WebServiceA, RegisterForCompletion, CoordinatorCa)  
invokeOp (CoordinatorCa, CompletionCoordinator, WebServiceA)  
sendMsg (WebServiceA, OperationalMessage, WebServiceB)

invokeOp (WebServiceB, CreateCC, CoordinatorCb)  
invokeOp (CoordinatorCb, T2, WebServiceB)  
invokeOp (WebServiceB, RegisterForPhaseZero, CoordinatorCb)  
invokeOp (CoordinatorCb, RegisterForPhaseZero, CoordinatorCa)  
invokeOp (CoordinatorCa, PhaseZeroCoordinator, CoordinatorCb)  
invokeOp (CoordinatorCb, PhaseZeroCoordinator, WebServiceB)  
invokeOp (WebServiceB, RegisterFor2PC, CoordinatorCb)  
invokeOp (CoordinatorCb, RegisterFor2PC, CoordinatorCa)  
invokeOp (CoordinatorCa, 2PCCoordinator, CoordinatorCb)  
invokeOp (CoordinatorCb, 2PCCoordinator, WebServiceB)

sendMsg (WebServiceA, Complete, CoordinatorCa)

sendMsg (CoordinatorCa, PhaseZero, CoordinatorCb)  
sendMsg (CoordinatorCb, PhaseZero, WebServiceB)  
sendMsg (WebServiceB, PhaseZeroComplete, CoordinatorCb)  
sendMsg (CoordinatorCb, PhaseZeroComplete, CoordinatorCa)

sendMsg (CoordinatorCa, Prepare, CoordinatorCb)  
sendMsg (CoordinatorCb, Prepare, WebServiceB)  
sendMsg (WebServiceB, Prepared, CoordinatorCb)  
sendMsg (CoordinatorCb, Prepared, CoordinatorCa)

sendMsg (CoordinatorCa, Commit, CoordinatorCb)  
sendMsg (CoordinatorCb, Commit, WebServiceB)  
sendMsg (WebServiceB, Committed, CoordinatorCb)  
sendMsg (CoordinatorCb, Committed, CoordinatorCa)

sendMsg (CoordinatorCa, Completed, WebServiceA)

2) dynamisch:

Variablen:

WS-A ∈ Services  
WS-B ∈ Services  
C-A ∈ Coordinators  
C-B ∈ Coordinators  
PhaseZero ∈ Messages  
PhaseZeroComplete ∈ Messages

Funktionen:

ServiceState: Services → {WaitingForCoordinator, WaitingForCompletion, OPMsgSend, WaitingForPhaseZero, WaitingFor2PC, CompleteMsgSend, PhaseZeroReceived, PrepareReceived, CommitReceived, CompleteReceived}

CoordinatorState: Coordinators  $\rightarrow$  {CreateCoordinator, ContextCreated, CCAvailable, WaitingForPhaseZero, P0CoordAvail, 2PCCoordAvail, PhaseZeroSend, PhaseZeroReceived, PrepareSend, PrepareReceived, CommitSend, CommitReceived, CompleteReceived}

MyCoord: Services  $\rightarrow$  Coordinators

CreateCC: Coordinators  $\rightarrow$  {WaitingForCoordinator}

CreateTransactionContext: Services  $\rightarrow$  {ContextCreated}

RegisterForCompletion: Coordinators  $\rightarrow$  {WaitingForCompletion}

RegisterForPhaseZero: Coordinators  $\rightarrow$  {WaitingForPhaseZero}

PhaseZeroCoordinator: Coordinators  $\rightarrow$  {P0CoordAvail}

RegisterFor2PC: Services  $\rightarrow$  {WaitingFor2PC}

CompletionCoordinator: Services  $\rightarrow$  {CCAvailable}

SendOperationalMessage: Services  $\rightarrow$  {OPMsgSend}

OPPartner: Services  $\rightarrow$  Services

StartTransaction: Services  $\rightarrow$  Boolean

SendMsg: Messages  $\rightarrow$  {PhaseZeroSend, PhaseZeroReceived, PrepareSend, PrepareReceived, CommitSend, CommitReceived, CompleteMsgSend, CompleteReceived}

ASM:

AtomicTransaction =<sub>def</sub>  
 WebService(WS-A)  
 Coordinator(C-A)  
 WebService(WS-B)  
 Coordinator(C-B)

WebService(Self) =<sub>def</sub>

**If** StartTransaction (Self)  
 ServiceState(Self) := CreateCC(MyCoord(Self))

**If** CoordinatorState(MyCoordinator(Self)) = ContextCreated  
**If** StartTransaction (Self)  
 ServiceState(Self) := RegisterForCompletion(MyCoord(Self))

**If** !StartTransaction (Self)  
 ServiceState(Self) := RegisterForPhaseZero(MyCoord(Self))

**If** CoordinatorState(MyCoord(Self)) = CCAvailable  
**choose** service  $\in$  Services  
 ServiceState(Self) := SendOperationalMessage(service)  
 OPPartner(Self) := service

**If** CoordinatorState(MyCoord(Self)) = P0CoordAvail  
 ServiceState(Self) := RegisterFor2PC(MyCoord(Self))

**If** CoordinatorState(MyCoord(Self)) = PhaseZeroSend  
 ServiceState(Self) := SendMsg(PhaseZeroComplete)

**If** CoordinatorState(MyCoord(Self)) = PrepareSend  
 ServiceState(Self) := SendMsg(Prepared)

**If** CoordinatorState(MyCoord(Self)) = CommitSend  
 ServiceState(Self) := SendMsg(Committed)

**If** CoordinatorState(MyCoord(OPPartner(Self))) = 2PCCoordAvail  
 ServiceState(Self) := SendMsg(Complete)

**If** !StartTransaction(Self)  
**For all** service  $\in$  Services  
**If** OPPartner(service) = Self **and** ServiceState(service) = OPMsgSend  
 OPPartner(Self) := service

```

        ServiceState(Self) := CreateCC(MyCoord(Self))
If CoordinatorState(MyCoord(Self)) = CompleteReceived
    ServiceState(Self) := CompleteReceived

Coordinator(Self) =def
For all service ∈ Services
    If MyCoordinator(service) = Self
        If ServiceState(service) = WaitingForCoordinator
            CoordinatorState(Self) := CreateTransactionContext(service)
        If ServiceState(service) = WaitingForCompletion
            CoordinatorState(Self) := CompletionCoordinator(service)
        If ServiceState(service) = WaitingForPhaseZero
            choose OPService ∈ Services
                where OPPartner(OPService) = service
                    If CoordinatorState(MyCoord(OPService)) != P0CoordAvail
                        CoordinatorState(Self) := RegisterForPhaseZero
                            (MyCoord(OPService))
                    If CoordinatorState(MyCoord(OPService)) = P0CoordAvail
                        CoordinatorState(Self) := P0CoordAvail

        If ServiceState(service) = WaitingFor2PC
            choose OPService ∈ Services
                where OPPartner(OPService) = service
                    If CoordinatorState(MyCoord(OPService)) != 2PCCoordAvail
                        CoordinatorState(Self) := RegisterFor2PC
                            (MyCoord(OPService))
                    If CoordinatorState(MyCoord(OPService)) = 2PCCoordAvail
                        CoordinatorState(Self) := 2PCCoordAvail
                    If CoordinatorState(MyCoord(OPService)) = PhaseZeroSend
                        CoordinatorState(Self) := SendMsg(PhaseZero)
                    If CoordinatorState(MyCoord(OPService)) = PrepareSend
                        CoordinatorState(Self) := SendMsg(Prepare)
                    If CoordinatorState(MyCoord(OPService)) = CommitSend
                        CoordinatorState(Self) := SendMsg(Commit)

            choose OPService ∈ Services
                where OPPartner(service) = OPService
                    If CoordinatorState(MyCoord(OPService)) = WaitingForPhaseZero
                        CoordinatorState(Self) :=
                            PhaseZeroCoordinator(MyCoord(OPService))
                    If CoordinatorState(MyCoord(OPService)) = PhaseZeroReceived
                        CoordinatorState(Self) := SendMsg(Prepare)
                    If CoordinatorState(MyCoord(OPService)) = PrepareReceived
                        CoordinatorState(Self) := SendMsg(Commit)
                    If CoordinatorState(MyCoord(OPService)) = CommitReceived
                        CoordinatorState(Self) := SendMsg(Completed)

        If ServiceState(service) = CompleteMsgSend
            CoordinatorState(Self) := SendMsg(PhaseZero)
        If ServiceState(service) = PhaseZeroReceived
            CoordinatorState(Self) := SendMsg(PhaseZeroComplete)
        If ServiceState(service) = PrepareReceived
            CoordinatorState(Self) := SendMsg(Prepared)
        If ServiceState(service) = CommitReceived
            CoordinatorState(Self) := SendMsg(Committed)

```

**Port-Typen bei Geschäftsprozessen (Grafik Seite 232):**

BusinessActivityCoordinator ∈ **Object**  
WS-CoordinationInterfaces ∈ **Container**  
WS-TransactionInterfacesNeededForChaining ∈ **Container**  
WS-CoordinationInterfacesNeededForChaining ∈ **Container**  
WS-TransactionInterfaces ∈ **Container**  
ActivationCoordinatorPortType ∈ **Port**  
RegistrationCoordinatorPortType ∈ **Port**

BusinessAgreementParticipantPortType ∈ **Port**  
BusinessAgreementWithCompleteParticipantPortType ∈ **Port**

Registration ParticipantPortType ∈ **Port**

BusinessAgreementCoordinatorPortType ∈ **Port**  
BusinessAgreementWithCompleteCoordinatorPortType ∈ **Port**

**Relationen:**

ActivationCoordinatorPortType portOf BusinessActivityCoordinator  
RegistrationCoordinatorPortType portOf BusinessActivityCoordinator  
BusinessAgreementParticipantPortType portOf BusinessActivityCoordinator  
BusinessAgreementWithCompleteParticipantPortType portOf BusinessActivityCoordinator  
Registration ParticipantPortType portOf BusinessActivityCoordinator  
BusinessAgreementCoordinatorPortType portOf BusinessActivityCoordinator  
BusinessAgreementWithCompleteCoordinatorPortType portOf BusinessActivityCoordinator

ActivationCoordinatorPortType partOf WS-CoordinationInterfaces  
RegistrationCoordinatorPortType partOf WS-CoordinationInterfaces

BusinessAgreementParticipantPortType partOf WS-TransactionInterfacesNeededForChaining  
BusinessAgreementWithCompleteParticipantPortType  
partOf WS-TransactionInterfacesNeededForChaining

Registration ParticipantPortType partOf WS-CoordinationInterfacesNeededForChaining

BusinessAgreementCoordinatorPortType partOf WS-TransactionInterfaces  
BusinessAgreementWithCompleteCoordinatorPortType partOf WS-TransactionInterfaces

**Beispielhafte Ausführung des Geschäftsprozessprotokolles (Grafik Seite 233)\*:**

1) statisch:

WebServiceA ∈ **Object**  
WebServiceB ∈ **Object**  
WebServiceC ∈ **Object**  
CoordinatorR ∈ **Object**

**Relationen:**

invokeOp (WebServiceA, CreateCC, CoordinatorR)  
invokeOp (CoordinatorR, A1, WebServiceA)  
sendMsg (WebServiceA, OperationalMessage, WebServiceB)

sendMsg (WebServiceA, OperationalMessage, WebServiceC)

invokeOp (WebServiceB, RegisterForBusinessAgreement, CoordinatorR)

invokeOp (CoordinatorR, BusinessAgreementCoordinator, WebServiceB)

invokeOp (WebServiceC, RegisterForBusinessAgreement, CoordinatorR)

invokeOp (CoordinatorR, BusinessAgreementCoordinator, WebServiceC)

sendMsg (WebServiceB, Completed, CoordinatorR)

sendMsg (WebServiceC, Faulted, CoordinatorR)

sendMsg (CoordinatorR, Compensate, WebServiceB)

sendMsg (CoordinatorR, Forget, WebServiceC)

## 2) dynamisch:

### Variablen:

WS-A  $\in$  Services

WS-B  $\in$  Services

WS-C  $\in$  Services

CoordR  $\in$  Coordinators

### Funktionen:

StartExecution: Services  $\rightarrow$  Boolean

CreateCC: Services  $\rightarrow$  Boolean

CCRespMsg: Services  $\rightarrow$  Messages

CreateCCRespMsg: Services  $\rightarrow$  Messages

OpMsg: Services  $\rightarrow$  Boolean

RegisterForBusinessAgreement: Services  $\rightarrow$  Boolean

BACoord: Services  $\rightarrow$  Message

CreateBACoord: Services  $\rightarrow$  Message

Execute: Services  $\rightarrow$  Message

Completed: Services  $\rightarrow$  Boolean

### ASM:

BusinessActivity =<sub>def</sub>

WebService(WS-A)

WebService(WS-B)

WebService(WS-C)

Coordinator

WebService(Self) =<sub>def</sub>

**If** StartExecution

    CreateCC(Self) := **true**

**If** CCRespMsg(WebService) != **undef**

    OpMsg(WS-B) := **true**

    OpMsg(WS-C) := **true**

**If** OpMsg(Self)

    RegisterForBusinessAgreement(Self) := **true**

**If** BACoord(WebService) != **undef**

    Completed(Self) = Execute(Self)

**If** !Completed(Self)

    Forget(Self)

**choose** WebService  $\in$  Services

        Compensate(WebService)

Coordinator =<sub>def</sub>  
**For all** Webservice ∈ Services  
**If** CreateCC(WebService)  
    CCRespMsg(WebService) := CreateCCRespMsg(WebService)  
**If** RegisterForBusinessAgreement(Self)  
    BACoord(WebService) := CreateBACoord(WebService)

**RosettaNet PIP 3A2 (Grafik Seite 237):**

Customer ∈ **Object**  
Start ∈ **Point**  
RequestResponseActivity ∈ **Container**  
RequestPriceAndAvailability ∈ **Activity**  
SecureFlow(1) ∈ **Container**  
SendPriceAndAvailabilityRequest ∈ **Activity**  
Failed ∈ **Point**  
End ∈ **Point**  
ProductSupplier ∈ **Object**  
SecureFlow(2) ∈ **Container**  
SendPriceAndAvailabilityResponse ∈ **Activity**  
ProcessPriceAndAvailabilityRequest ∈ **Activity**

**Relationen:**

RequestPriceAndAvailability partOf RequestResponseActivity  
SendPriceAndAvailabilityRequest partOf SecureFlow(1)

Start partOf Customer  
RequestResponseActivity partOf Customer  
SecureFlow(1) partOf Customer  
Failed partOf Customer  
End partOf Customer

PriceAndAvailabilityResponse partOf SecureFlow(2)  
SecureFlow(2) partOf ProductSupplier  
ProcessPriceAndAvailabilityRequest partOf ProductSupplier

Start predOf RequestResponseActivity  
label((RequestResponseActivity predOf Failed), Fail)  
label((RequestResponseActivity predOf End), Success)  
RequestResponseActivity predOf SecureFlow(1)  
SecureFlow(1) predOf ProcessPriceAndAvailabilityRequest  
ProcessPriceAndAvailabilityRequest predOf SecureFlow(2)  
SecureFlow(2) predOf RequestResponseActivity

**RosettaNet Protokoll und Nachrichtenkomponenten (Grafik Seite 239):**

MIMEMultipart/RelatedEnvelope ∈ **Activity**  
Headers ∈ **Container**  
PreambleHeader ∈ **Activity**  
DeliveryHeader ∈ **Activity**  
ServiceHeader ∈ **Activity**  
BusinessProcessPayload ∈ **Activity**

ServiceContent ∈ **Activity**  
AttachmentN ∈ **Activity**  
-(1) ∈ **undefObj**  
Attachment1 ∈ **Activity**  
HTTPS ∈ **Activity**  
SSL ∈ **Activity**  
HTTP ∈ **Activity**  
SMTP ∈ **Activity**  
Other(Future) ∈ **Activity**  
TransportProtocols(TCP/IPAndOthers) ∈ **Activity**

**Relationen:**

new AttachmentN  
new -(1)  
new Attachment1

PreambleHeader equal DeliveryHeader  
DeliveryHeader equal ServiceHeader

PreambleHeader partOf Headers  
DeliveryHeader partOf Headers  
ServiceHeader partOf Headers  
Headers partOf MIMEMultipart/RelatedEnvelope

ServiceContent partOf BusinessProcessPayload  
AttachmentN partOf BusinessProcessPayload  
-(1) partOf BusinessProcessPayload  
Attachment1 partOf BusinessProcessPayload  
BusinessProcessPayload partOf MIMEMultipart/RelatedEnvelope

HTTPS equal SSL  
SSL equal HTTP  
HTTP equal SMTP  
SMTP equal Other(Future)  
Other(Future) equal TransportProtocols(TCP/IPAndOthers)

## Kapitel 8

### **Client-Konversation mit mehreren WebServices (Grafik Seite 246):**

Approval(WebService) ∈ **Object**  
Customer(Client) ∈ **Object**  
Supplier(WebService) ∈ **Object**  
AnotherSupplier(WebService) ∈ **Object**

**Relationen:**

entryPoints (Supplier(WebService))  
entryPoints (AnotherSupplier(WebService))

invokeOp (Customer(Client), NotifyPayment, Approval(WebService))  
invokeOp (Customer(Client), RequestQuote, AnotherSupplier(WebService))  
invokeOp (Customer(Client), RequestQuote, Supplier(WebService))  
invokeOp (Customer(Client), OrderGoods, Supplier(WebService))

invokeOp (Customer(Client), MakePayment, Supplier(WebService))

**zusammengesetzte Services (Grafik Seite 247):**

Customer ∈ **Activity**  
SupplyChain ∈ **Object**  
InventoryPlanning ∈ **Object**  
Accounting ∈ **Object**  
Procurement ∈ **Object**  
Approval ∈ **Object**  
-(1) ∈ **undefObj**  
-(2) ∈ **undefObj**  
Supplier ∈ **Object**  
AnotherSupplier ∈ **Object**

**Relationen:**

SupplyChain partOf Customer  
InventoryPlanning partOf Customer  
Accounting partOf Customer  
Procurement partOf Customer  
Approval partOf Customer  
-(1) partOf Customer  
-(2) partOf Customer

SupplyChain comm InventoryPlanning  
SupplyChain comm Accounting  
SupplyChain comm Procurement

InventoryPlanning comm -(1)  
InventoryPlanning comm -(2)

Procurement comm Approval  
Procurement comm Supplier  
Procurement comm AnotherSupplier

**Implementation von zusammengesetzten WebServices (Grafik Seite 249):**

CompanyA ∈ **Activity**  
(Composite)WebService ∈ **Object**  
WebServicesMiddleware ∈ **Middle**  
InternalServiceInterface ∈ **Object**  
ImplementationOfTheCompositionLogic ∈ **Object**  
ConventionalMiddleware ∈ **Activity**  
CompanyB ∈ **Activity**  
WebService(1) ∈ **Object**  
WebService(2) ∈ **Object**  
CompanyC ∈ **Activity**  
WebService(3) ∈ **Object**  
WebService(4) ∈ **Object**  
CompanyD ∈ **Activity**  
WebService(5) ∈ **Object**

**Relationen:**

(Composite)WebService partOf CompanyA  
WebServicesMiddleware partOf CompanyA  
InternalServiceInterface partOf CompanyA  
ImplementationOfTheCompositionLogic partOf CompanyA  
ConventionalMiddleware partOf CompanyA

WebService(1) partOf CompanyB  
WebService(2) partOf CompanyB

WebService(3) partOf CompanyC  
WebService(4) partOf CompanyC

WebService(5) partOf CompanyD

(Composite)WebService comm WebServicesMiddleware  
WebServicesMiddleware comm ConventionalMiddleware  
ImplementationOfTheCompositionLogic comm ConventionalMiddleware

allowsVia(ImplementationOfTheCompositionLogic, {ConventionalMiddleware,  
WebServicesMiddleware}, WebService(1))

allowsVia(ImplementationOfTheCompositionLogic, {ConventionalMiddleware,  
WebServicesMiddleware}, WebService(3))

allowsVia(ImplementationOfTheCompositionLogic, {ConventionalMiddleware,  
WebServicesMiddleware}, WebService(5))

**grundlegende Bestandteile einer WS-Composition Middleware (Grafik Seite 250):**

ServiceProvider ∈ **Activity**  
SchemaDesigner ∈ **Object**  
ServiceCompositionModelAndLanguage ∈ **Object**  
WebServiceCompositionMiddleware ∈ **Activity**  
DevelopmentEnvironment ∈ **Activity**  
SchemaDefinitions ∈ **Specification**  
Run-TimeEnvironment(CompositionEngine) ∈ **Activity**  
CompositionServiceExecutionData ∈ **Resource**  
OtherWebServiceMiddleware ∈ **Activity**  
ServicesOfferedByOtherProviders ∈ **Container**  
Supplier ∈ **Object**  
Warehouse ∈ **Object**  
Accounting ∈ **Object**

**Relationen:**

numberOf (SchemaDefinitions)

DevelopmentEnvironment partOf WebServiceCompositionSoftware  
SchemaDefinitions partOf WebServiceCompositionSoftware  
Run-TimeEnvironment(CompositionEngine) partOf WebServiceCompositionSoftware  
CompositionServiceExecutionData partOf WebServiceCompositionSoftware

WebServiceCompositionSoftware partOf ServiceProvider

OtherWebServiceMiddleware partOf ServiceProvider  
SchemaDesigner partOf ServiceProvider  
ServiceCompositionModelAndLanguage partOf ServiceProvider

Supplier partOf ServicesOfferedByOtherProviders  
Warehouse partOf ServicesOfferedByOtherProviders  
Accounting partOf ServicesOfferedByOtherProviders

SchemaDesigner predOf ServiceCompositionModelAndLanguage  
ServiceCompositionModelAndLanguage predOf DevelopmentEnvironment  
DevelopmentEnvironment predOf SchemaDefintions  
SchemaDefintions predOf Run-TimeEnvironment(CompositionEngine)  
Run-TimeEnvironment(CompositionEngine) comm CompositionServiceExecutionData  
Run-TimeEnvironment(CompositionEngine) comm OtherWebServiceMiddleware  
OtherWebServiceMiddleware comm Supplier  
OtherWebServiceMiddleware comm Warehouse  
OtherWebServiceMiddleware comm Accounting

**Implementation zusammengesetzter WS über WS-Middleware (Grafik Seite 251):**

CompanyA ∈ **Activity**  
(Composite)WebService ∈ **Object**  
WebServicesMiddleware ∈ **Middle**  
ServiceCompositionSupport(ModelingAndExecution) ∈ **Object**  
OtherTiers ∈ **Container**  
CompanyB ∈ **Activity**  
CompanyC ∈ **Activity**  
CompanyD ∈ **Activity**  
WebService(1) ∈ **Object**  
WebService(2) ∈ **Object**  
WebService(3) ∈ **Object**  
WebService(4) ∈ **Object**  
WebService(5) ∈ **Object**

**Relationen:**

ServiceCompositionSupport(ModelingAndExecution) partOf WebServicesMiddleware  
WebServicesMiddleware partOf CompanyA  
(Composite)WebService partOf CompanyA  
OtherTiers partOf CompanyA

WebService(1) partOf CompanyB  
WebService(2) partOf CompanyB

WebService(1) partOf CompanyC  
WebService(2) partOf CompanyC

WebService(1) partOf CompanyD

(Composite)WebService comm WebServicesMiddleware  
OtherTiers comm WebServicesMiddleware

ServiceCompositionSupport(ModelingAndExecution) comm WebService(1)  
ServiceCompositionSupport(ModelingAndExecution) comm WebService(3)

ServiceCompositionSupport(ModelingAndExecution) comm WebService(5)

**Externe Interaktion vs. externe Implementation (Grafik Seite 252):**

Customer ∈ **Object**  
Supplier ∈ **Object**  
CompositionEngine ∈ **Activity**  
ConversationController ∈ **Activity**  
AnotherWebService ∈ **Container**  
YetAnotherWebService ∈ **Container**

**Relation:**

entryPoints (Customer)  
entryPoints (Supplier)

CompositionEngine partOf Supplier  
ConversationController partOf Supplier

CompositionEngine comm ConversationController  
ConversationController comm AnotherWebService  
ConversationController comm YetAnotherWebService

invokeOp (Customer, 1:RequestQuote, Supplier)  
invokeOp (Customer, 2:OrderGoods, Supplier)  
invokeOp (Customer, 3:ConfirmOrder, Supplier)  
invokeOp (Customer, 4:MakePayment, Supplier)

**Prozessdiagramm eines Lieferanten-WebServices (Grafik Seite 258)\*:**

1) statisch:

Customer ∈ **Object**  
LocalServiceOfferedByTheSupplier ∈ **Object**  
Warehouse ∈ **Object**  
Supplier ∈ **Object**  
-(1) ∈ **Point**  
ReceiveOrderGoods ∈ **Object**  
InvokeCheckLocalStock ∈ **Object**  
InvokeCheckShipAvailable ∈ **Object**  
SendCancelOrder ∈ **Object**  
SendConfirmOrder ∈ **Object**  
-(2) ∈ **Point**  
-(3) ∈ **Point**  
inStock ∈ **Boolean**  
shippingAvail ∈ **Boolean**

**Relationen:**

-(1) partOf Supplier  
ReceiveOrderGoods partOf Supplier  
InvokeCheckLocalStock partOf Supplier  
InvokeCheckShipAvailable partOf Supplier  
SendCancelOrder partOf Supplier  
SendConfirmOrder partOf Supplier

-(2) partOf Supplier

-(3) partOf Supplier

-(1) predOf ReceiveOrderGoods

ReceiveOrderGoods predOf InvokeCheckLocalStock

OrStatement (InvokeCheckLocalStock, inStock, SendConfirmOrder, InvokeCheckShipAvailable)

OrStatement (InvokeCheckShipAvailable, shippingAvail, SendConfirmOrder, SendCancelOrder)

SendConfirmOrder predOf -(2)

SendCancelOrder predOf -(3)

sendMsg (Customer, OrderGoods, ReceiveOrderGoods)

sendMsg (SendConfirmOrder, ConfirmOrder, Customer)

sendMsg (SendCancelOrder, CancelOrder, CancelOrder)

InvokeCheckLocalStock comm LocalServiceOfferedByTheSupplier

InvokeCheckShipAvailable comm Warehouse

## 2) dynamisch:

Variablen:

StartOrdering  $\in$  Boolean

suppl  $\in$  Suppliers

cust  $\in$  Customers

product  $\in$  Goods

OrderReceived  $\in$  Boolean

Funktionen:

OrderGoods: Suppliers x Goods  $\rightarrow$  Boolean

ReceiveOrderGoods: Goods  $\rightarrow$  Boolean

inStock: Goods  $\rightarrow$  Boolean

Send: Message x Customers  $\rightarrow$  Boolean

InvokeCheckLocalStock: Goods  $\rightarrow$  Boolean

InvokeCheckShipAvailable: Goods  $\rightarrow$  Boolean

ProductInStock: Goods  $\rightarrow$  Boolean

shippingAvail: Goods  $\rightarrow$  Boolean

ProductAvailable: Goods  $\rightarrow$  Boolean

Finished: Customers  $\rightarrow$  Boolean

ASM:

SupplierServiceI-E  $=_{\text{def}}$

**If** !Finished

Supplier

Customer

LocalService

Warehouse

Customer  $=_{\text{def}}$

**for all** product **in** Goods

**If** StartOrdering(product)

**choose** suppl  $\in$  Suppliers

OrderGoods(suppl, product) := **true**

Supplier  $=_{\text{def}}$

**for all** product **in** Goods **and** cust **in** Customers

**If** OrderGoods(Self, product)

**If** !OrderReceived

OrderReceived := ReceiveOrderGoods (product, cust)

```

        InvokeCheckLocalStock (product) := true
If OrderReceived
    If inStock(product)
        Finished(cust) := Send(ConfirmOrder, cust)
    If !inStock(product)
        If shippingAvail(product) = undef
            InvokeCheckShipAvailable(product) := true
        If shippingAvail(product)
            Finished(cust) := Send(ConfirmOrder, cust)
        If !shippingAvail(product)
            Finished(cust) := Send(CancelOrder, cust)

LocalService =def
    for all product in Goods
        If InvokeCheckLocalStock(product)
            inStock(product) := ProductInStock (product)

Warehouse =def
    for all product in Goods
        If InvokeCheckShipAvailable(product)
            shippingAvail(product) := ProductAvailable (product)

```

**Statechart des Lieferanten-WebServices (Grafik Seite 260)\*:**

1) statisch:

-(1) ∈ **Point**  
 Started ∈ **Object**  
 SearchingForProductsLocally ∈ **Object**  
 SearchingForProductsAtOtherSupplier ∈ **Object**  
 OrderCancelled ∈ **Object**  
 -(2) ∈ **Point**  
 OrderCompleted ∈ **Object**  
 -(3) ∈ **Point**  
 InvokeCheckLocalStock ∈ **Activity**  
 SendConfirmOrder ∈ **Activity**  
 InvokeCheckShipAvailable ∈ **Activity**  
 SendCancelOrder ∈ **Activity**  
 inStock ∈ **Boolean**  
 shippingAvail ∈ **Boolean**

**Relationen:**

label ((-1) predOf Started), StartOnNewOrderRequest  
 Started predOf InvokeCheckLocalStock  
 InvokeCheckLocalStock predOf SearchingForProductsLocally  
 OrStatement (SearchingForProductsLocally, inStock, SendConfirmOrder,  
 InvokeCheckShipAvailable)  
 InvokeCheckShipAvailable predOf SearchingForProductsAtOtherSupplier  
 OrStatement (SearchingForProductsAtOtherSupplier, shippingAvail, SendConfirmOrder,  
 SendCancelOrder)  
 SendConfirmOrder predOf OrderCompleted  
 OrderCompleted predOf -(3)  
 SendCancelOrder predOf OrderCancelled  
 OrderCancelled predOf -(2)

## 2) dynamisch:

Variablen:

product  $\in$  Goods  
ConfirmOrder  $\in$  Messages  
CancelOrder  $\in$  Messages

Funktionen:

NewOrderRequest: Goods  $\rightarrow$  Boolean  
inStock: Goods  $\rightarrow$  Boolean  
shippingAvail: Goods  $\rightarrow$  Boolean  
CheckLocalStock: Goods  $\rightarrow$  Boolean  
Order: Customers  $\rightarrow$  Boolean  
Send: Messages x Customers  $\rightarrow$  Boolean

ASM:

SupplierProcessSC  $\stackrel{=def}{}$

```
for all product in Goods
  If NewOrderRequest(product)
    If inStock(product) = undef
      inStock(product) := CheckLocalStock(product)
    If inStock(product)
      Order(cust) := Send(ConfirmOrder, cust)
    If !inStock(product)
      If shippingAvail(product) = undef
        shippingAvail(product) := CheckShipAvailable(product)
      If shippingAvail(product)
        Order(cust) := Send(ConfirmOrder, cust)
      If !shippingAvail(product)
        Order(cust) := Send(CancelOrder, cust)
```

### **Petrinetzdarstellung des Lieferantenprozesses (Grafik Seite 261):**

Da es sich hierbei um ein Petrinetz handelt, werden Objekte hier als Kreise (Zustände) betrachtet. Die Dynamik dieses Prozesses entspricht der auf Seite 260, da es sich hier nur um zwei unterschiedliche Darstellungsmöglichkeiten des selben Sachverhaltes handelt.

Start  $\in$  **Object**

InvokeCheckLocalStock  $\in$  **Activity**

LocalSystemAccessed  $\in$  **Object**

InvokeCheckShipAvailable  $\in$  **Activity**

DoNothing(1)  $\in$  **Activity**

ExternalSupplierAccessed  $\in$  **Object**

SendCancelOrder  $\in$  **Activity**

Complete(Cancel)  $\in$  **Object**

DoNothing(2)  $\in$  **Activity**

ReadyToSendConfirmation  $\in$  **Object**

SendConfirmOrder  $\in$  **Activity**

Complete(Confirm)  $\in$  **Object**

### **Relationen:**

Start  $\text{predOf}$  InvokeCheckLocalStock

InvokeCheckLocalStock  $\text{predOf}$  LocalSystemAccessed

label ((LocalSystemAccessed  $\text{predOf}$  DoNothing(1)), inStock=true)

label ((LocalSystemAccessed  $\text{predOf}$  InvokeCheckShipAvailable), inStock=false)

DoNothing(1)  $\text{predOf}$  ReadyToSendConfirmation

InvokeCheckShipAvailable predOf ExternalSupplierAccessed  
 label ((ExternalSupplierAccessed predOf DoNothing(2)), shippingAvail=true)  
 DoNothing(2) predOf ReadyToSendConfirmation  
 label ((ExternalSupplierAccessed predOf SendCancelOrder), shippingAvail=false)  
 SendCancelOrder predOf Complete(Cancel)  
 ReadyToSendConfirmation predOf SendConfirmOrder  
 SendConfirmOrder predOf Complete(Confirm)

**Prozesshierarchie des Lieferanten-Prozesses (Grafik Seite 263):**

ProcessOrder(Sequence) ∈ **Activity**  
 ReceiveOrderGoods ∈ **Activity**  
 InvokeCheckLocalStock ∈ **Activity**  
 DiscriminateBasedOnLocalSearch(Choice) ∈ **Activity**  
 SendConfirmOrder(1) ∈ **Activity**  
 SearchExternalSupplier(Sequence) ∈ **Activity**  
 InvokeCheckShipAvailable ∈ **Activity**  
 DiscriminateBasedOnExternalSearch(Choice) ∈ **Activity**  
 SendConfirmOrder(2) ∈ **Activity**  
 SendCancelOrder ∈ **Activity**

**Relationen:**

ProcessOrder(Sequence) parentOf {ReceiveOrderGoods, InvokeCheckLocalStock,  
 DiscriminateBasedOnLocalSearch(Choice)}  
 label((DiscriminateBasedOnLocalSearch(Choice) parentOf {SendConfirmOrder(1)}),  
 inStock=true)  
 label((DiscriminateBasedOnLocalSearch(Choice) parentOf  
 {SearchExternalSupplier(Sequence)}), inStock=false)  
 SearchExternalSupplier(Sequence) parentOf {InvokeCheckShipAvailable,  
 DiscriminateBasedOnExternalSearch(Choice)}  
 label((DiscriminateBasedOnExternalSearch(Choice) parentOf {SendCancelOrder}),  
 shippingAvail=false)  
 label((DiscriminateBasedOnExternalSearch(Choice) parentOf {SendConfirmOrder(2)}),  
 shippingAvail=true)

**Datenflussansatz für die Datenübertragung zwischen Komponenten (Grafik Seite 267):**

-(1) ∈ **Point**  
 A ∈ **Object**  
 B ∈ **Object**  
 C ∈ **Object**  
 -(2) ∈ **Point**

**Relationen:**

-(1) predOf A  
 A predOf B  
 B predOf C  
 C predOf -(2)  
 sendMsg (A, Quantity, B)  
 sendMsg (A, Quantity, C)

sendMsg (B, Price, C)

**Serviceauswahl anhand von UDDI-Einträgen (Grafik Seite 268)\*:**

Zwischen UDDIRegistry und InvokeGetBindingDetail hier “comm” anstatt des ursprünglichen “predOf”

1) statisch:

Supplier ∈ **Object**

-(1) ∈ **Point**

ReceiveOrderGoods ∈ **Object**

InvokeCheckLocalStock ∈ **Object**

InvokeGetBindingDetail ∈ **Object**

UDDIRegistry ∈ **Resource**

InvokeCheckShipAvailable ∈ **Object**

SendCancelOrder ∈ **Object**

SendConfirmOrder ∈ **Object**

-(2) ∈ **Point**

-(3) ∈ **Point**

inStock ∈ **Boolean**

shippingAvail ∈ **Boolean**

**Relationen:**

-(1) partOf Supplier

ReceiveOrderGoods partOf Supplier

InvokeCheckLocalStock partOf Supplier

InvokeCheckShipAvailable partOf Supplier

SendCancelOrder partOf Supplier

SendConfirmOrder partOf Supplier

-(2) partOf Supplier

-(3) partOf Supplier

-(1) predOf ReceiveOrderGoods

ReceiveOrderGoods predOf InvokeCheckLocalStock

OrStatement (InvokeCheckLocalStock, inStock, SendConfirmOrder, InvokeGetBindingDetail)

InvokeGetBindingDetail predOf InvokeCheckShipAvailable

InvokeGetBindingDetail comm UDDIRegistry

OrStatement (InvokeCheckShipAvailable, shippingAvail, SendConfirmOrder, SendCancelOrder)

SendConfirmOrder predOf -(2)

SendCancelOrder predOf -(3)

2) dynamisch:

Variablen:

cust ∈ Customers

product ∈ Goods

OrderReceived ∈ Boolean

GotDetails ∈ Boolean

Funktionen:

ReceiveOrderGoods: Goods → Boolean

inStock: Goods → Boolean

Send: Message x Customers → Boolean

InvokeCheckLocalStock: Goods → Boolean  
 InvokeCheckShipAvailable: Goods → Boolean  
 shippingAvail: Goods → Boolean  
 Finished: Customers → Boolean

ASM:

```

SupplierService =def
  If !Finished
    Supplier
  
```

```

Supplier =def
  For all cust in Customers and product in Goods
    OrderReceived := ReceiveOrderGoods (product, cust)
    If OrderReceived and inStock(product) = undef
      inStock(product) := InvokeCheckLocalStock(product)
    If inStock(product)
      Finished(cust) := Send(ConfirmOrder, cust)
    If !inStock(product)
      If !GotDetails
        GotDetails := InvokeGetBindingDetail(product)
      If GotDetails
        If shippingAvail(product) = undef
          shippingAvail(product) :=
            InvokeCheckShipAvailable(product)
        If shippingAvail(product)
          Finished(cust) := Send(ConfirmOrder, cust)
        If !shippingAvail(product)
          Finished(cust) := Send(CancelOrder, cust)
  
```

**Dynamic binding (Grafik Seite 269):**

-(1) ∈ Point  
 BookAPlane ∈ Object  
 BookABoat ∈ Object  
 BookACar ∈ Object  
 -(2) ∈ undefObj  
 -(3) ∈ Point  
 -(4) ∈ Or  
 -(5) ∈ Or

**Relationen:**

-(1) predOf -(4)  
 -(4) predOf BookAPlane  
 -(4) predOf BookABoat  
 -(4) predOf BookACar  
 -(4) predOf -(2)  
 BookAPlane predOf -(5)  
 BookABoat predOf -(5)  
 BookACar predOf -(5)  
 -(2) predOf -(5)  
 -(5) predOf -(3)

**Beispiel für eine atomare Region (Grafik Seite 270):**

-(1) ∈ **Point**  
A ∈ **Object**  
B ∈ **Object**  
C ∈ **Object**  
D ∈ **Object**  
-(2) ∈ **Point**  
AtomicRegion ∈ **Container**

**Relationen:**

-(1) predOf A  
A predOf B  
B predOf C  
C predOf D  
D predOf -(2)

A partOf AtomicRegion  
B partOf AtomicRegion

**ACID-Teiltransaktionen (Grafik Seite 272):**

ForwardExecution ∈ **Object**  
SubtransactionS1 ∈ **Object**  
SubtransactionS2 ∈ **Object**  
-(1) ∈ **undefObj**  
SubtransactionSn ∈ **Object**  
CompensationFlow ∈ **Container**  
CompensatingSubtransactionCS1 ∈ **Object**  
CompensatingSubtransactionCS2 ∈ **Object**  
CompensatingSubtransactionCSn ∈ **Object**  
-(2) ∈ **undefObj**

**Relationen:**

SubtransactionS1 partOf ForwardExecution  
SubtransactionS2 partOf ForwardExecution  
-(1) partOf ForwardExecution  
SubtransactionSn partOf ForwardExecution

CompensatingSubtransactionCS1 partOf CompensationFlow  
CompensatingSubtransactionCS2 partOf CompensationFlow  
-(2) partOf CompensationFlow  
CompensatingSubtransactionCSn partOf CompensationFlow

SubtransactionS1 predOf SubtransactionS2  
SubtransactionS2 predOf -(1)  
-(1) predOf SubtransactionSn

CompensatingSubtransactionCSn predOf -(2)  
-(2) predOf CompensatingSubtransactionCS2  
CompensatingSubtransactionCS2 predOf CompensatingSubtransactionCS1

## Fehlerbehandlung innerhalb eines Prozesses (Grafik Seite 274)\*:

### 1) statisch:

Supplier  $\in$  **Object**  
receive\_orderGoods  $\in$  **Object**  
invoke\_checkLocalStock  $\in$  **Object**  
invoke\_checkShipAvailable  $\in$  **Object**  
send\_cancelOrder  $\in$  **Object**  
send\_confirmOrder  $\in$  **Object**  
some\_exceptionhandling\_logic\_here  $\in$  **Object**  
-(1)  $\in$  **Or**  
-(2)  $\in$  **Or**  
-(3)  $\in$  **Or**

### **Relationen:**

start (receive\_orderGoods)  
end (send\_cancelOrder)  
end (send\_confirmOrder)  
end (-(3))  
{receive\_orderGoods, invoke\_checkLocalStock, invoke\_checkShipAvailable} partOf Supplier  
{send\_cancelOrder, send\_confirmOrder, some\_exceptionhandling\_logic\_here} partOf Supplier  
receive\_orderGoods predOf invoke\_checkLocalStock  
invoke\_checkLocalStock predOf -(1)  
-(1) predOf invoke\_checkShipAvailable  
-(1) predOf send\_confirmOrder  
-(1) predOf some\_exceptionhandling\_logic\_here  
invoke\_checkShipAvailable predOf -(2)  
-(2) predOf send\_cancelOrder  
-(2) predOf send\_confirmOrder  
some\_exceptionhandling\_logic\_here predOf -(3)  
-(3) predOf invoke\_checkLocalStock

### 2) dynamisch:

#### Variablen:

OrderedGoods  $\in$  Goods  
inStock  $\in$  Boolean  
shippingAvail  $\in$  Boolean  
Errors\_Occured  $\in$  Boolean  
orderGoods  $\in$  Message  
confirmOrder  $\in$  Message  
cancelOrder  $\in$  Message  
repeat  $\in$  Boolean

#### Funktionen:

checkLocalStock: Goods x ... x Goods  $\rightarrow$  Boolean  
Receive: Message  $\rightarrow$  Goods  
Send: Message  $\rightarrow$  Boolean  
ExceptionHandling:  $\rightarrow$  Boolean

#### ASM:

Supplier =<sub>def</sub>  
  **If** OrderedGoods = **undef or Repeat**  
    OrderedGoods := Receive(orderGoods)  
  **If** OrderedGoods != **undef**  
    inStock := checkLocalStock(OrderedGoods)  
  **If** !inStock **and** !Errors\_Occured

```

        shippingAvail := checkShipAvailable(OrderedGoods)
If (inStock and !Errors_Occured) or shippingAvail
        Send(confirmOrder)
If !shippingAvail
        Send(cancelOrder)
If Errors_Occured
        Repeat := ExceptionHandling()

```

**Koordinationsprotokoll eines Bestellvorganges (Grafik Seite 277):**

Entspricht der Grafik auf Seite 205

Dementsprechend ist die sie beschreibende ASM auch die gleiche.

**Händlersicht auf das Koordinationsprotokoll (Grafik Seite 278)\*:**

1) statisch

Der dazugehörige "Customer View" entspricht der Grafik Seite 207

-(1) ∈ **Point**

-(2) ∈ **Point**

-(3) ∈ **Point**

Customer ∈ **Container**

RequestQuote(ToSupplier) ∈ **Object**

OrderGoods(ToSupplier) ∈ **Object**

MakePayment(ToSupplier) ∈ **Object**

Supplier ∈ **Container**

CheckShipAvailable(ToWarehouse) ∈ **Object**

ConfirmOrder(ToCustomer) ∈ **Object**

CancelOrder(ToCustomer) ∈ **Object**

OrderShipment(ToWarehouse) ∈ **Object**

Warehouse ∈ **Container**

ConfirmShipment(ToSupplier) ∈ **Object**

WarehouseConfirms ∈ **Boolean**

**Relationen:**

RequestQuote(ToSupplier) partOf Customer

OrderGoods(ToSupplier) partOf Customer

MakePayment(ToSupplier) partOf Customer

CheckShipAvailable(ToWarehouse) partOf Supplier

ConfirmOrder(ToCustomer) partOf Supplier

CancelOrder(ToCustomer) partOf Supplier

OrderShipment(ToWarehouse) partOf Supplier

ConfirmShipment(ToSupplier) partOf Warehouse

-(1) predOf RequestQuote(ToSupplier)

RequestQuote(ToSupplier) predOf OrderGoods(ToSupplier)

OrderGoods(ToSupplier) predOf CheckShipAvailable(ToWarehouse)

OrStatement

(CheckShipAvailable(ToWarehouse), WarehouseConfirms, ConfirmOrder(ToCustomer),  
CancelOrder(ToCustomer))

CancelOrder(ToCustomer) predOf -(2)

ConfirmOrder(ToCustomer) predOf MakePayment(ToSupplier)

MakePayment(ToSupplier) predOf OrderShipment(ToWarehouse)

OrderShipment(ToWarehouse) predOf ConfirmShipment(ToSupplier)  
ConfirmShipment(ToSupplier) predOf -(3)

2) dynamisch:

Variablen:

ProcurementConversation  
ShippingAvail ∈ Boolean  
ShipmentConfirmed ∈ Boolean  
ShipmentOrdered ∈ Boolean  
OrderCancelled ∈ Boolean

Funktionen:

ProcurementConversation  
CheckShipAvailable: Message x Message x Message → Boolean  
OrderShipment: Message x Message x Message → Boolean  
ConfirmShipment: Message x Message x Message → Boolean  
CancelOrder: Message x Message → Boolean

ASM:

ProcurementProtocolSV =<sub>def</sub>  
    **If** !OrderCancelled **and** !ShipmentConfirmed  
        CustomerSC  
        SupplierPPSV  
        Warehouse

SupplierPPSV =<sub>def</sub>  
    SupplierPP  
    **If** !ShippingAvail  
        OrderCancelled := CancelOrder(Goods, CustomerMsg)

WarehousePPSV =<sub>def</sub>  
    **If** ShipmentOrdered  
        ShipmentConfirmed :=  
            ConfirmShipment(SupplierMsg, Goods, CustomerMsg)

**Beschreibung der Rolle des Händlers im Koordinationsprotokoll (Grafik Seite 279)\*:**

1) statisch:

-(1) ∈ **Point**  
ReceiveRequestQuote ∈ **Object**  
ReplyRequestQuote ∈ **Object**  
ReceiveOrderGoods ∈ **Object**  
InvokeCheckShipAvailable ∈ **Object**  
SendCancelOrder ∈ **Object**  
-(2) ∈ **Point**  
SendConfirmOrder ∈ **Object**  
ReceiveMakePayment ∈ **Object**  
SendOrderShipment ∈ **Object**  
ReceiveConfirmShipment ∈ **Object**  
-(3) ∈ **Point**  
shippingAvail ∈ **Boolean**



**If** !ShippingAvail  
     OrderCancelled := SendCancelOrder(GoodsOrdered, CustomerMsg)  
**If** OrderConfirmed **and** !GoodsPaid  
     GoodsPaid := ReceiveMakePayment(GoodsOrdered, CustomerMsg)  
**If** GoodsPaid **and** !ShipmentOrdered  
     ShipmentOrdered := SendOrderShipment(GoodsOrdered, CustomerMsg)  
**If** ShipmentOrdered  
     ShipmentConfirmed := ReceiveConfirmShipment(GoodsOrdered,  
   CustomerMsg)

**Implementation der Lieferanten-Rolle (Grafik Seite 280)\*:**

1) statisch:

Supplier ∈ **Object**

-(1) ∈ **Point**

ReceiveRequestQuote ∈ **Object**

InvokeLookupQuote ∈ **Object**

ReplyRequestQuote ∈ **Object**

ReceiveOrderGoods ∈ **Object**

InvokeCheckShipAvailable ∈ **Object**

SendCancelOrder ∈ **Object**

-(2) ∈ **Point**

SendConfirmOrder ∈ **Object**

ReceiveMakePayment ∈ **Object**

InvokeCollectPayment ∈ **Object**

SendOrderShipment ∈ **Object**

ReceiveConfirmShipment ∈ **Object**

-(3) ∈ **Point**

shippingAvail ∈ **Boolean**

**Relationen:**

-(1) partOf Supplier

ReceiveRequestQuote partOf Supplier

InvokeLookupQuote partOf Supplier

ReplyRequestQuote partOf Supplier

ReceiveOrderGoods partOf Supplier

InvokeCheckShipAvailable partOf Supplier t

SendCancelOrder partOf Supplier

-(2) partOf Supplier

SendConfirmOrder partOf Supplier

ReceiveMakePayment partOf Supplier

InvokeCollectPayment partOf Supplier

SendOrderShipment partOf Supplier

ReceiveConfirmShipment partOf Supplier

-(3) partOf Supplier

-(1) predOf ReceiveRequestQuote

ReceiveRequestQuote predOf InvokeLookupQuote

InvokeLookupQuote predOf ReplyRequestQuote

ReplyRequestQuote predOf ReceiveOrderGoods

ReceiveOrderGoods predOf InvokeCheckShipAvailable

OrStatement (InvokeCheckShipAvailable, shippingAvail, SendConfirmOrder, SendCancelOrder)

SendCancelOrder predOf -(2)

SendConfirmOrder predOf ReceiveMakePayment  
 ReceiveMakePayment predOf InvokeCollectPayment  
 InvokeCollectPayment predOf SendOrderShipment  
 SendOrderShipment predOf ReceiveConfirmShipment  
 ReceiveConfirmShipment predOf -(3)

2) dynamisch:

Variablen:

Quote ∈ Message  
 Replied ∈ Boolean  
 Received ∈ Boolean  
 LookedUp ∈ Boolean  
 PaymantCollected ∈ Boolean  
 GoodsOrdered ∈ Message  
 ShippingAvail ∈ Boolean  
 CustomerMsg ∈ Message  
 OrderConfirmed ∈ Boolean  
 OrderCancelled ∈ Boolean  
 GoodsPaid ∈ Boolean  
 ShipmentConfirmed ∈ Boolean  
 ShipmentOrdered ∈ Boolean

Funktionen:

ReceiveRequestQuote: → Boolean  
 ReplyRequestQuote: Message → Boolean  
 ReceiveOrderGoods: Message → Message  
 ReceiveMakePayment: Message x Message → Boolean  
 SendConfirmOrder: Message x Message → Boolean  
 InvokeCheckShipAvailable: Message x Message → Boolean  
 InvokeLookupQuote: → Message  
 SendOrderShipment: Message x Message x Message → Boolean  
 ReceiveConfirmShipment: Message x Message x Message → Boolean  
 SendCancelOrder: Message x Message → Boolean

ASM:

ProcurementProtocolSV =<sub>def</sub>  
     **If** !OrderCancelled **and** !ShipmentConfirmed  
         Supplier

Supplier =<sub>def</sub>  
     **If** !Received  
         Received := ReceiveRequestQuote()  
     **If** Received **and** Quote = **undef**  
         Quote := InvokeLookupQuote()  
     SupplierPPSVR

**Routing-Problem der Kompositionskomponente (Grafik Seite 283):**

ServiceProvider ∈ **Activity**

InstanceOfACompositionSchema ∈ **Object** (könnte z.B. den auf Seite 258 beschriebenen Supplier beinhalten)

CompositionEngine ∈ **Object**

Object(WebServiceImplementation) ∈ **Object**

ConversationController ∈ **Object**

AnotherWebService ∈ **Object**

**Relationen:**

numberOf (InstanceOfACompositionSchema)  
numberOf (Object(WebServiceImplementation))  
InstanceOfACompositionSchema partOf Supplier  
CompositionEngine partOf Supplier  
Object(WebServiceImplementation) partOf Supplier  
ConversationController partOf Supplier

ConversationController comm AnotherWebService  
ConversationController predOf CompositionEngine  
ConversationController predOf Object(WebServiceImplementation)  
CompositionEngine predOf InstanceOfACompositionSchema

**Umfang der BPEL-Spezifikation (Grafik Seite 285)\*:**

(Abänderung von Seite 258)

1) statisch:

Customer ∈ **Object**  
LocalServiceOfferedByTheSupplier ∈ **Object**  
Warehouse ∈ **Object**  
Supplier ∈ **Object**  
-(1) ∈ **Point**  
ReceiveOrderGoods ∈ **Object**  
InvokeCheckLocalStock ∈ **Object**  
InvokeCheckShipAvailable ∈ **Object**  
SendCancelOrder ∈ **Object**  
SendConfirmOrder ∈ **Object**  
-(2) ∈ **Point**  
-(3) ∈ **Point**  
Roles ∈ **Container**  
PortTypes ∈ **Container**  
-(4) ∈ **Port**  
-(5) ∈ **Port**  
-(6) ∈ **Port**  
-(7) ∈ **Port**  
-(8) ∈ **Port**  
inStock ∈ **Boolean**  
shippingAvail ∈ **Boolean**

**Relationen:**

Customer partOf Roles  
Warehouse partOf Roles  
LocalServiceOfferedByTheSupplier partOf Roles

-(4) partOf PortTypes  
-(5) partOf PortTypes  
-(6) partOf PortTypes  
-(7) partOf PortTypes  
-(8) partOf PortTypes  
  
-(4) portOf Customer

-(5) portOf Warehouse  
-(6) portOf LocalServiceOfferedByTheSupplier  
-(7) portOf Supplier  
-(8) portOf Supplier

-(4) comm -(7)  
-(5) comm -(7)  
-(6) comm -(8)

-(1) partOf Supplier  
ReceiveOrderGoods partOf Supplier  
InvokeCheckLocalStock partOf Supplier  
InvokeCheckShipAvailable partOf Supplier  
InvokeCancelOrder partOf Supplier  
InvokeConfirmOrder partOf Supplier  
-(2) partOf Supplier  
-(3) partOf Supplier

-(1) predOf ReceiveOrderGoods  
ReceiveOrderGoods predOf InvokeCheckLocalStock  
OrStatement (InvokeCheckLocalStock, inStock, InvokeConfirmOrder,  
    InvokeCheckShipAvailable)  
OrStatement (InvokeCheckShipAvailable, shippingAvail, InvokeConfirmOrder,  
InvokeCancelOrder)  
InvokeConfirmOrder predOf -(2)  
InvokeCancelOrder predOf -(3)

2) dynamisch (für den Supplier, da der Rest nur Strukturen beschreibt und durch den statischen Teil schon abgedeckt wird):

Variablen:

OrderedGoods ∈ Goods  
inStock ∈ Boolean  
shippingAvail ∈ Boolean  
orderGoods ∈ Message  
confirmOrder ∈ Message  
cancelOrder ∈ Message  
Confirmed ∈ Boolean  
Cancelled ∈ Boolean

Funktionen:

InvokeCheckLocalStock: Goods x ... x Goods → Boolean  
Receive: Message → Goods  
Send: Message → Boolean  
InvokeConfirmOrder: Goods → Boolean  
InvokeCancelOrder: Goods → Boolean

ASM:

Supplier =<sub>def</sub>  
    **If** OrderedGoods = **undef**  
        OrderedGoods := Receive(orderGoods)  
    **If** OrderedGoods != **undef**  
        inStock := InvokeCheckLocalStock(OrderedGoods)  
    **If** !inStock **and** shippingAvail = **undef**  
        shippingAvail := checkShipAvailable(OrderedGoods)

**If** (inStock **or** shippingAvail) **and** !Confirmed  
     Confirmed := InvokeConfirmOrder(orderGoods)  
**If** !shippingAvail **and** !Cancelled  
     Cancelled := InvokeCancelOrder(orderGoods)

**Prozessstrukturen in BPEL (Grafik Seite 287)\*:**

1) statisch:

Entspricht der Struktur nach der Grafik auf Seite 263 –nur einige Elemente wurden umbenannt–diese sind im Folgenden rot markiert.

ProcessOrder(Sequence) ∈ **Activity**  
 ReceiveOrderGoods ∈ **Activity**  
 InvokeCheckLocalStock ∈ **Activity**  
 ChooseLocal(Switch) ∈ **Activity**  
 InvokeConfirmOrder(1) ∈ **Activity**  
 SearchExternal (Sequence) ∈ **Activity**  
 InvokeCheckShipAvailable ∈ **Activity**  
 ChooseExternal(Switch) ∈ **Activity**  
 InvokeConfirmOrder(2) ∈ **Activity**  
 InvokeCancelOrder ∈ **Activity**

**Relationen:**

ProcessOrder(Sequence) parentOf {ReceiveOrderGoods, InvokeCheckLocalStock,  
     ChooseLocal(Switch)}  
 label((ChooseLocal(Switch) parentOf {InvokeConfirmOrder(1)}), inStock=true)  
 label((ChooseLocal(Switch) parentOf {SearchExternal(Sequence)}), inStock=false)  
 SearchExternal(Sequence) parentOf {InvokeCheckShipAvailable, ChooseExternal(Switch)}  
 label((ChooseExternal(Switch) parentOf {InvokeCancelOrder}), shippingAvail=false)  
 label((ChooseExternal(Switch) parentOf {InvokeConfirmOrder(2)}), shippingAvail=true)

2) dynamisch:

Variablen:

inStock ∈ Boolean  
 shippingAvail ∈ Boolean  
 product ∈ Goods  
 OrderMsgs ∈ Messages  
 finished ∈ Boolean

Funktionen:

ReceiveOrderGoods: Message → Goods  
 InvokeCheckLocalStock: Goods → Boolean  
 InvokeConfirmOrder: Messages → Boolean  
 InvokeCheckShipAvailable: Goods → Boolean  
 InvokeCancelOrder: Messages → Boolean

ASM:

ProcessOrder =<sub>def</sub>  
     **If** product = **undef**  
         product := ReceiveOrderGoods(OrderMsgs)  
     **If** inStock = **undef**  
         inStock := InvokeCheckLocalStock(product)  
 ChooseLocal

```

ChooseLocal =def
    If !inStock
        searchExternal
    If inStock
        finished := InvokeConfirmOrder(OrderMsgs)

searchExternal =def
    If shippingAvail = undef
        shippingAvail := InvokeCheckShipAvailable(product)
    If shippingAvail != undef
        chooseExternal

chooseExternal =def
    If shippingAvail
        finished := InvokeConfirmOrder(OrderMsgs)
    If !shippingAvail
        finished := InvokeCancelOrder(OrderMsgs)

```

**Partnerdefinitionen in BPEL (Grafik Seite 290):**

Customer ∈ **Object**  
Warehouse ∈ **Object**  
LocalServiceOfferedByTheSupplier ∈ **Object**  
Supplier ∈ **Object**  
PartnerLinkType\_OrderLT ∈ **Container**  
PortType\_SupplierPT  
-(1) ∈ **Port**  
-(2) ∈ **Port**  
-(3) ∈ **Port**  
-(4) ∈ **Port**  
-(5) ∈ **Port**

**Relationen:**

-(1) portOf Customer  
-(2) portOf Warehouse  
-(3) portOf LocalServiceOfferedByTheSupplier  
-(4) portOf Supplier  
-(5) portOf Supplier

-(4) partOf PortType\_SupplierPT  
-(5) partOf PortType\_SupplierPT

-(1) partOf PartnerLinkType\_OrderLT  
-(4) partOf PartnerLinkType\_OrderLT

-(1) rel -(4)  
-(2) rel -(4)  
-(3) rel -(5)

### **Fehlerbehandlung in BPEL (Grafik Seite 291):**

Erweiterung der Grafik auf Seite 287 –im Folgenden werden nur die neu hinzugekommenen Elemente notiert während die alte Grafik als “Orchestration” abekürzt aufgeführt wird.

Orchestration

FaultHandlerForFaultF ∈ **Activity**

-(1) ∈ **Error**

ScopeOfTheSearchExternalActivity ∈ **Container**

#### **Relationen:**

-(1) equal InvokeConfirmOrder

FaultHandlerForFaultF partOf SearchExternal(Sequence)

SearchExternal(Sequence) partOf ScopeOfTheSearchExternalActivity

InvokeCheckShipAvailable partOf ScopeOfTheSearchExternalActivity

ChooseExternal(Switch) partOf ScopeOfTheSearchExternalActivity

InvokeConfirmOrder partOf ScopeOfTheSearchExternalActivity

-(1) partOf ScopeOfTheSearchExternalActivity

InvokeCancelOrder partOf ScopeOfTheSearchExternalActivity

### **Korrelation zwischen Nachrichten (Grafik Seite 293):**

Warehouse ∈ **Object**

Supplier ∈ **Object**

MessageCheckAvailability ∈ **Activity**

OrderId(1) ∈ **Object**

RequestedDeliveryDate ∈ **Object**

DeliveryLocation ∈ **Object**

-(1) ∈ **undefObj**

MessageAvailability ∈ **Activity**

OrderId(2) ∈ **Object**

ShippingAvail ∈ **Object**

#### **Relationen:**

OrderId(1) partOf MessageCheckAvailability

RequestedDeliveryDate partOf MessageCheckAvailability

DeliveryLocation partOf MessageCheckAvailability

-(1) partOf MessageCheckAvailability

OrderId(2) partOf MessageAvailability

ShippingAvail partOf MessageAvailability

Supplier predOf MessageCheckAvailability

MessageCheckAvailability predOf Warehouse

Warehouse predOf MessageAvailability

MessageAvailability predOf Supplier

## Kapitel 9

### **Closed Community with Direct Interaction (Grafik Seite 304):**

SmallClosedCommunity ∈ **Container**  
CompanyA ∈ **Activity**  
CompanyB ∈ **Activity**  
CompanyC ∈ **Activity**  
WS(1) ∈ **Object**  
WS(2) ∈ **Object**  
WS(3) ∈ **Object**  
WS(4) ∈ **Object**  
WS(5) ∈ **Object**  
WS(6) ∈ **Object**  
WS(7) ∈ **Object**

#### **Relationen:**

WS(1) partOf CompanyA  
WS(2) partOf CompanyA  
CompanyA partOf SmallClosedCommunity

WS(3) partOf CompanyB  
WS(4) partOf CompanyB  
CompanyB partOf SmallClosedCommunity

WS(5) partOf CompanyC  
WS(6) partOf CompanyC  
WS(7) partOf CompanyC  
CompanyC partOf SmallClosedCommunity

label((CompanyA comm CompanyB), HTTP\_XML\_SOAP)  
label((CompanyA comm CompanyC), HTTP\_XML\_SOAP)  
label((CompanyC comm CompanyB), HTTP\_XML\_SOAP)

### **Interaktionsvermittlung mit Hubs (Grafik Seite 306):**

WS(1) ∈ **Object**  
WS(2) ∈ **Object**  
WS(3) ∈ **Object**  
WS(4) ∈ **Object**  
Hub ∈ **Object**  
PropertiesOfHubClients ∈ **Resource**

#### **Relationen:**

WS(1) comm Hub  
WS(2) comm Hub  
WS(3) comm Hub  
WS(4) comm Hub  
PropertiesOfHubClients comm Hub

### **Kontrolle von Middlewarekomponenten durch ein WSMS (Grafik Seite 310):**

SOAPMessages ∈ **Container**  
MiddlewareComponent ∈ **Activity**  
SOAPRouter&ConversationController ∈ **Object**  
HorizontalProtocolHandlers ∈ **Object**  
CompositionEngine ∈ **Object**  
UnderlyingApplicationObjects ∈ **Object**  
WebServiceMgmtSystem ∈ **Activity**  
SOAPMgmt ∈ **Object**  
ConversationMgmt ∈ **Object**  
CompositionMgmt ∈ **Object**  
ApplicationMgmt ∈ **Object**  
MgmtData&ControlActions ∈ **Container**

#### **Relationen:**

SOAPRouter&ConversationController partOf MiddlewareComponent  
HorizontalProtocolHandlers partOf MiddlewareComponent  
CompositionEngine partOf MiddlewareComponent  
UnderlyingApplicationObjects partOf MiddlewareComponent

SOAPMgmt partOf WebServiceMgmtSystem  
ConversationMgmt partOf WebServiceMgmtSystem  
CompositionMgmt partOf WebServiceMgmtSystem  
ApplicationMgmt partOf WebServiceMgmtSystem

SOAPMessages predOf SOAPRouter&ConversationController  
SOAPRouter&ConversationController predOf HorizontalProtocolHandles  
SOAPRouter&ConversationController predOf CompositionEngine  
SOAPRouter&ConversationController predOf UnderlyingApplicationObjects

SOAPRouter&ConversationController comm MgmtData&ControlActions  
SOAPRouter&ConversationController comm MgmtData&ControlActions  
CompositionEngine comm MgmtData&ControlActions  
UnderlyingApplicationObjects comm MgmtData&ControlActions

MgmtData&ControlActions comm SOAPMgmt  
MgmtData&ControlActions comm ConversationMgmt  
MgmtData&ControlActions comm CompositionMgmt  
MgmtData&ControlActions comm ApplicationMgmt

### **Externe Architektur des WS-Managements (Grafik Seite 315):**

WebService(1) ∈ **Object**  
WebService(2) ∈ **Object**  
Trusted3rdParty ∈ **Object**

#### **Relationen:**

entryPoints (WebService(1))  
entryPoints (WebService(2))  
entryPoints (Trusted3rdParty)  
label((WebService(1) comm WebService(2)), MgmtProtocols(LimitedVisibility))  
label((WebService(1) comm Trusted3rdParty), MgmtProtocols(CompleteVisibility))  
label((WebService(2) comm Trusted3rdParty), MgmtProtocols(CompleteVisibility))

### **Management Outsourcing (Grafik Seite 317):**

Enterprise ∈ **Activity**

EnterpriseResources(ExposedThroughWSInterfaces) ∈ **Object**

MSP ∈ **Activity**

MgmtSystem(ImplementedAsWS) ∈ **Object**

#### **Relationen:**

EnterpriseResources(ExposedThroughWSInterfaces) partOf Enterprise

MgmtSystem(ImplementedAsWS) partOf MSP

label((EnterpriseResources(ExposedThroughWSInterfaces) comm  
MgmtSystem(ImplementedAsWS)), SOAP-BasedMgmtProtocols)

### **Wechselwirkungen zwischen Systemen via XMLCIM (Grafik Seite 318):**

MgmtSystems ∈ **Object**

CIMDataModelRepository ∈ **Resource**

ManagedResources ∈ **Object**

#### **Relationen:**

label((MgmtSystems comm CIMDataModelRepository), XMLCIM\_XMLOverHTTP)

label((CIMDataModelRepository comm ManagedResources),  
ProprietaryProtocolsToPopulateRepository)

### **Austausch von Managementinformationen mit Hilfe von XMLCIM (Grafik Seite 319):**

WebServiceMgmtSystem(1) ∈ **Object**

CIMRepository(1) ∈ **Resource**

WebServiceMgmtSystem(2) ∈ **Object**

CIMRepository(2) ∈ **Resource**

WebService(1) ∈ **Object**

WebService(2) ∈ **Object**

#### **Relationen:**

CIMRepository(1) comm WebService(1)

((CIMRepository(1) comm WebServiceMgmtSystem(1)), XMLCIM)

((CIMRepository(1) comm WebServiceMgmtSystem(2)), XMLCIM)

CIMRepository(2) comm WebService(2)

((CIMRepository(2) comm WebServiceMgmtSystem(2)), XMLCIM)

((CIMRepository(2) comm WebServiceMgmtSystem(1)), XMLCIM)

**Bibliographie:**

- [WS]: „Web Services“ von Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju  
[GUR99]: Yuri Gurevich: Sequential Abstract State Machines Capture Sequential Algorithms  
[WR]: Wolfgang Reisig: On Gurevich's theorem on sequential algorithms