Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2250

Abstrakte Sichten auf BPEL Prozesse

Jiayang Cai

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. David Schumm
begonnen am:	16. November 2009
beendet am:	18. Mai 2010
CR-Klassifikation:	D.2.2, H.4.1, H.5.2

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Geschäftsprozesse in Unternehmen werden wegen die aufwachsenden Konkurrenzen und ständig veränderte Interaktionen mit dem Partnern immer komplexer. Viele Unternehmen schreiben die rote Zahlen in Betrieb wegen die Chaos und Unklarheit in einen unzuverlässig Geschäftsprozess. Ein agiler Geschäftsprozess schafft den besser Flexibilität in Marktveränderung und Konkurrenzfähigkeit in den Branche für den Unternehmen. Das effizient Management und Optimierung von Geschäftsprozess werden dann als den wichtigen Ressourcen in großem Unternehmen nach den strategische Zielsetzung und Entwicklungsplanung betrachtet.

Eine Geschäftsprozess können mehrere Hunderte von Aktivitäten und viele relevante auf hierarchische Managementebenen eingeordnete Geschäftsinformation wegen die hoch Geschäftskomplexität und Vielfalt von die Web-Services [WCL+05] basierte Interaktion beinhalten. Eine geschäftlichen Prozessanalyse wird immer aufwändig, wenn das privat Geschäftsprozess mit mehrere extern Prozesses z.B durch Business-to-Business (B2B) [SG07] oder Business Process Outsourcing (BPO) [Sch08a] kommunizieren und zusammenwirken müssen. Deswegen wird die anwendungsspezifische Vereinfachung von Geschäftsprozess sehr hilfreich. Es wird so benötigt, dass die private Geschäftsprozess nach entsprechen Kriterien z.B die geschäftlich Sicherheitsstufen verarbeitet und klassifiziert werden können. Eine sinnvolle abstrakte Sicht bzw. »Process View« [Str09] von private Basisprozess sollen einen bessere anwendungsspezifische Businessanalyse unterstützt.

WS-BPEL [AAA+] ist einen standardisierte Beschreibungssprache für Modellierung, Simulieren und Management von Geschäftsprozess in einen automatisierte Umgebung. Es wird viel Werkzeugen und Systeme angeboten, die die Analysieren und Management von BPEL-Prozess unterstützen. Eine Erweiterung von Werkzeug wird dabei sehr notwendig, die dieser Funktion sowie einen abstrakte generierte Sicht von Basisprozess realisiert. In Arbeit von [Str09] wird die generelle Konzept von »Process View« auf BPEL-Prozess verdeutlicht, die Realisierbarkeit von praktische Implementierung wird durch einen ausführbare Prototyp in Java verwiest.

Nach den Zielsetzung dieser Arbeit wird der Prototyp weiterhin fortgesetzt, um einen installierbare Applikation zu erschaffen, die nach Angabe der entsprechende Verarbeitungsoperationen und Aktivitätsauswahl einen abstrakte Prozess aus den ursprünglich Prozess generiert. Einen modifizierte Applikationspaket können in einen Anwendungsserver integriert werden, um dieser Funktion ohne Installation in Webbrowser zu aufrufen.

1.1. Motivation

Die Motivation dieser Arbeit ist den Vervollständigung und Erweiterung die grundlegende View-Operationen auf den Basis von [Str09] , mit den die Basisgeschäftsprozess praktisch verarbeiten und abstrakte Prozesses flexibel generieren werden können. Es wurde in den Praktische Arbeit einen Jar-Paket [Mica] entwickelt, die einen basische BPEL-Prozess und entsprechende Verarbeitungsregeln bzw. Rules-Dokument [Str09] als Eingaben einlest. Eine erwünscht Teilprozess bzw. Prozessfragment oder abstrakte Prozess wird erzeugt, die die Komplexität von ursprünglich Prozess nach Anforderung reduziert und die kritische Informationen wie geschäftliche Details verborgen.

1.2. Gliederung der Arbeit

In Kapitel 1 werden die Themengebiet und Aufgabestellung der Arbeit eingeführt, die Motivation und Zielsetzung werden dabei erklärt. Die grundlegende Theorie und technische Ansätze werden in Kapitel 2 vorgestellt, die relevant für den Verständnissen und Implementierung der praktische Aufgabe der Arbeit sind. In Kapitel 3 werden die Konzept und Operationen von Prozessverarbeitung für einen abstrakte Sicht von Basisprozess prinzipiell erzählt. Die praktische Implementierung von erweiterte View-Operationen und entsprechende Algorithmen werden in Kapitel 4 mit anschauliche Anwendungsfällen in Details erläutert. In Kapitel 5 wird die Schlussfolgerung der Arbeit kurz zusammengefasst und einen Ausblick für den »Process View« in weitere Anwendungsaspekten gegeben.

2. Grundlagen und Technik

In diesem Kapitel werden die wichtige und grundlegenden Theorie und Technologien vermittelt, auf den dieser Studienarbeit basiert. Es wird zuerst die fundamentale Definition und Prinzipien in den Gebieten von Web Services und BPEL-Prozesse erläutert, dann werden die eingesetzte Implementierungstools bzw. XML und Java-Programmierschnittstellen erklärt, die in den Praktischer Aufgabenteil angewendet wurde. Die Informationen sollen für einen besser Verständnis über die ganze Arbeitsumfang geliefert werden.

In Abschnitt 2.1 wird Service-orientierte Architektur mit weltweite bekannte Abkürzung SOA vorgestellt und deren Implementierung bzw. Web Service Technologie erzählt. Es wird auf den wichtige Begriffe und Konzepte eingegangen, die als den Entwurfsbausteinen in den heutiger Service-orientierte Geschäften und Realisierungen betrachtet werden.

Geschäftsprozess wird durch Komposition von mehrere Services modelliert und auch rekursiv als einen vollständige Services betrachtet. In Abschnitt 2.2 auf Seite 14 wird die Geschäftsprozessmanagement oder genannt als Business Prozess Management erläutert. Web Services Business Process Exection Language (WS-BPEL) [AAA+] implementiert die ausführbare Geschäftsprozess auf den Rechnerumgebung und wird als einen effektive Sprache für Prozessmodellierung entwickelt. WS-BPEL und die entsprechende Spezifikation wird in dieser Abschnitt erzählt.

In Abschnitt 2.3 auf Seite 20 wird die wichtige Konzept von eXtensible Markup Language (XML) gegeben, die selber eine Metasprache für den Datendokument wegen die hoch Flexibilität und Struktureigenschaft ist. Die weiter Technik wie XML Path Language wird vorgestellt, die die Aufgaben in dieser Arbeit wechselseitige verbindet.

In den Praktischer Arbeit wurde die verschiedenen Schnittstellen angewendet, die auf den XML-Dokument verarbeiten können. In Abschnitt 2.4 auf Seite 23 wird die drei wichtige Programmierschnittstellen bzw. Parser vorgestellt und die entsprechende Übersicht in den Anwendung kurz erstellt.

2.1. SOA und Web Services

Services-orientierte Architektur ist einen abstrakte Konzept und architektonisch Paradigma für den Entwurf und Realisierung bzw. Aufrufen von Services. Eine deutliche Vorteil von SOA ist die Unabhängigkeit von jeweils Implementierung von Services, dieser ermöglicht

eine funktional Zerlegung der Anwendung und einen erfolgreicher prozessorientierte Betrachtungsweise [DJMZ05]. SOA wird immer mehr im Unternehmen eingesetzt, die einzelne Services sind getriebene Plattform- und zugehöriger Anwendung-unabhängig, die aus geschäftlichen Anwendungen im Unternehmen durch SOA zerlegt wird. Es bietet mehrere Flexibilität bei Integration und neue erstellte Funktionalitäten an.

2.1.1. Services

Services sind die Kernelement von SOA. Viele Details von Services wird in [WCL+05] ausführlich erklärt. Die folgende Definitionen von Services und Web-Services in weiter Abschnitten werden in den Prinzip aus [WCL+05] entnommen. In jeder Tag wird verschiedene Dienstleistung bzw. Services von den Menschen oder geschäftlichen Organisation aufgerufen, die in den umfangreiche Bereiche die verschieden Bedürfnis und Anforderungen erfüllen. In den technische Ebenen wird eine Services als einen Software betrachtet, die einen konkrete Ziel oder Aufgabe nach bestimmte Service-Management durch Aufrufen und Kommunikation entsprechende Informationen bzw. Nachrichten realisiert können.

Services wird als einen kompakte Bündnis von mehrere angeboten Funktionalitäten gesehen. Es könnte kompositorisch gestalten werden, d.h, es wird aus mehrere Komponenten bestehen und als einen großer aber agiler Service ausgeliefert. Einen Geschäftsprozess wird auch als einen Orchestrierung von Services in geschäftliche Dimension entwickelt und für den Partnern angeboten. Die angebotene Services werden durch verschieden Technologien oder deren Mischung entwickelt und in die Öffentlichkeit durch einen eindeutige und sinnvolle Beschreibung registriert, deren Beschreibung prinzipiell auf Metasprache z.B. XML basiert. Die Beschreibungssprache von Services wie in [CMRW07] wird in weiter Abschnitte über Web Services generell aufgeklärt.

Services verborgen die technische Details und Hintergrunde für den Benutzern, die den konkrete Implementierungstechnik und Softwarekomponenten angeben. Services bieten eine anwendungsspezifische Abstraktion von Programmierung-Modell an, die Endbenutzern rufen die Methoden oder Funktionen in den Softwarekomponenten nach abstrakte Spezifikation ohne die genau Wissen über die Implementierung von Softwarekomponenten auf, deswegen wird die Anwendungen, die die Services realisieren sollen, mit den Aufrufen oder andere Anwendungen lose gekoppelt. Es benötigt deutliche wenig Bedingungen unter entsprechende Kommunikationsmethode sowie asynchrone Zusammenarbeit von geschäftlichen Anwendungen durch Services. Es schafft einen besser Integration und effektive Zusammenarbeitsfähigkeit von Anwendungen im Unternehmen, die auf verschieden Programmierung-Modellen und Softwarekomponenten aufgebaut.

Services stehen in einen bestimmte Ort in den Netzwerk zu Verfügung. Die Services senden und bekommen die Nachrichten durch globale Internet oder der lokale Netzwerk nach den entsprechenden Servicespezifikationen. Die Servicespezifikationen beschreiben die funktionale Informationen, die die Funktionalitäten bzw. Operationen und entsprechende Qualitätsanforderung von Services angeben. Solche Information wird für den Suchen und Auswahl von Services für den Anwender hilfreich. Die Rechnerumgebung, die die Services

anbietet, implementiert die Operationen mit bekommenden Nachrichten und sind für den Qualitätsanforderungen des Aufrufes verantwortlich. Die funktional Sicht von Services wird in einen abstrakte Sprache sowie Web Services Description Language (WSDL) in [CMRW07] definiert und besetzt eine hoch Erweiterbarkeit. Solche Beschreibungsdokument von Services wird veröffentlicht und leicht gefunden für den Servicebenutzern.

2.1.2. Services-orientierte Architektur

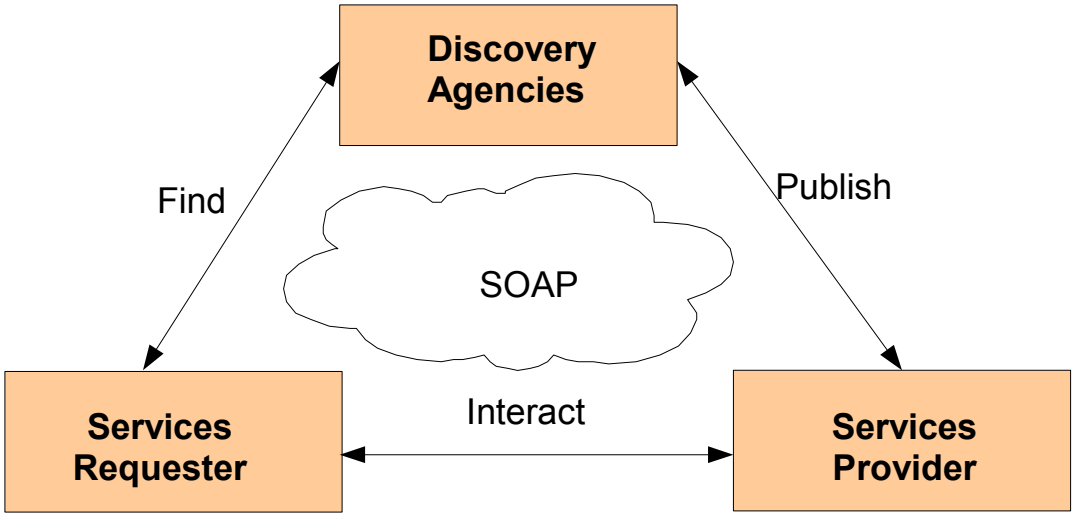


Abbildung 2.1.: SOA Dreieck

Die Abbildungen 2.1 aus [CFNO02] stellt das grundlegend SOA-Prinzip als eine Dreiecke dar, die die Relation von die drei Akteuren sowie Service-Anfordern (Services Requester), Service-Anbieter (Services Provider) und Service-Registrierung (Discovery Agencies) anschaulich erklärt. Der Services-Anbieter ist die Eigentümer von Services aus den geschäftlichen Sicht. Es ist auch einen IT-Plattform oder einen Server-Schichten wie in Client-Server aus den architektonisch Sicht, die alle Zugreifen nach Services unterstützen. Services-Anbieter können auch als einen Rechnerumgebung von Durchführung des Services oder als einen Service-Container betrachten werden. Aus den geschäftlichen Sicht ist die Services-Anfordern einen Geschäft, das die mehrere Funktionen anlaufen sollen und erfolgreich den Zielen mit den

bestimmte Qualität erreichen müssen. Es ist aus den architektonisch Sicht einen Anwendungsprogramm wie einen Client-Rolle in Client-Server-Architektur, das einen Verbindung mit Service-Anbieter initialisieren und weiter die Services aufrufen. Einen Services-Anbieter können auch einen Rollen sowie Services-Anfordern in einen andere Services-Szenario spielen, ein Service ruft einen anderer Service durch eine Anwendungsprogramm oder personelle Interaktion auf. Service-Registrierung können auch als einen Suchagent gesehen werden, darin wird es einen Mengen von dem Servicebeschreibungen gespeichert. Die mehrere Services-Anbietern geben die eigene Servicebeschreibungen ab, solche Beschreibungsdokumente werden zentrale oder verteilte in Datenbank oder von einem Proxy-Server verwaltet. Durch Untersuchung und Zusammenpassung von Qualität des Services z.B WS-Policy [BBC+06] finden die Services-Anfordern den effektive Services-Anbietern mit entsprechenden gültigen Beschreibungsdokument.

In den SOA-Dreiecke wird drei Basisoperationen zwischen drei Akteuren kennzeichnet [CFNO02]. Diese Operationen realisieren die Vorteile von SOA-Strategie im Anwendung. Die Grundoperationen sind Publikation von Servicebeschreibung, Fund von Servicebeschreibung und Interaktion oder Bindung von Services. Durch Publikation von Services wird die Services und deren als WSDL-Dokument [CMRW07] beschriebene Spezifikation für den Services-Anfordern sichtbar. Die Speicherort wird klassifiziert nach die geschäftliche Gebieten und Sorten des Services, das reduziert die Komplexität für den Recherchieren von Services. Die Registrierung und Durchsuchung von Services bzw. deren Standard wird in [CHR+04] aufgeklärt, die Services-Anfordern suchen die Datenbank von Beschreibungsdaten durch oder abfragen entsprechend Type von erwünschte Services, deren Spezifikation und Beschreibung globale standardisiert wird. In den Entwicklungsphase von Anwendungslösung wird dieser Bindung durch zwei Methoden wie aufgeklärte in [CFNO02] realisiert, in den Designphase von Anwendung wird die Schnittstellbeschreibung von Services benötigt und in den Laufzeit wird die konkrete angegebene Operationsbeschreibung bzw. die Aufrufspezifikation von Services durchgesucht. Nach einen erfolgreiche Bindung mit Services wird die Funktionen von Services aufgerufen, die Services-Anfordern und Services-Anbieter tauschen die Nachrichten z.B SOAP [ML03] in Internet mit verschiedenen Protokollen z.B HTTP [FGM+99] durch einander. Die Interaktion können synchron oder asynchron sein.

2.1.3. Web Services Technologie

Web Services Technologie ist einen Basislösung von SOA [WCL+05], die von keine konkrete IT-Plattform und Programmierung-Modell abhängig werden. Web Services ist bisher einen effektive Implementierung und Realisierung von SOA-Prinzip, die bereits schon in Industrie als beste Praxislösung eingesetzt wurden. Web Services mit die standardisierte Spezifikationen sollen einen Interoperabilität für den gebietsübergreifende Anwendungsintegration erzielen. Web Services sind einen Technik zur Maschinen-Maschinen-Kommunikation [DJMZ05].

Die Entwicklungsgang von Web Services und entsprechende Basiskomponente wird von World Wide Web Consortium (W3C) [W3Cc] verwaltet und der Begriff von Web Services wird in Folgende so definiert:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. [BHM+04]*

Trotzdem Web Services sind einen bisherig gute Lösung für den Interoperabilität, es gabt zahlreiche nicht homogene Anwendungssystem, die aus unterschieden Programmierung-Modell entwickelt oder durch mehrere Varianten von Zwischenanwendungen integriert. Es benötigt hoch Aufwand bei Kommunikation zwischen die Infrastruktur und die hochrangige Anwendungen. Es wurde durch Zusammenarbeit von mehr technische Gemeinschaften und große IT-Spitzunternehmen in den Branche solche vertraute Standard und Richtlinien für den Spezifikationen von Web Services vorgestellt. Die Abbildungen 2.2 aus [BHM+04] anschaulichen die wichtige Standard und Spezifikationen durch einen Web Services Stack, die den Interoperabilitätsproblem lösen können. Durch standardisierte Spezifikation mit XML-Technologie werden die allgemeine Format und Protokoll definiert, es schafft einen höhere Qualität von Interoperabilität mit erwünschte Sicherheit und Effizient.

In dieser Abschnitt werden die alle Spezifikationen von Web Services nicht erzählt, die entsprechende Spezifikationen von Protokoll, Kommunikationssicherheit und Servicebeschreibung können in W3C [W3Cb] gefunden werden, die von IT-Spezialisten ausführliche standardisiert werden. Die Implementierung von dieser Arbeit wird auf den WSDL 1.1 [CW01] basiert, die die Web Services aus Anwendungsfällen von dieser Arbeit funktional beschreiben.

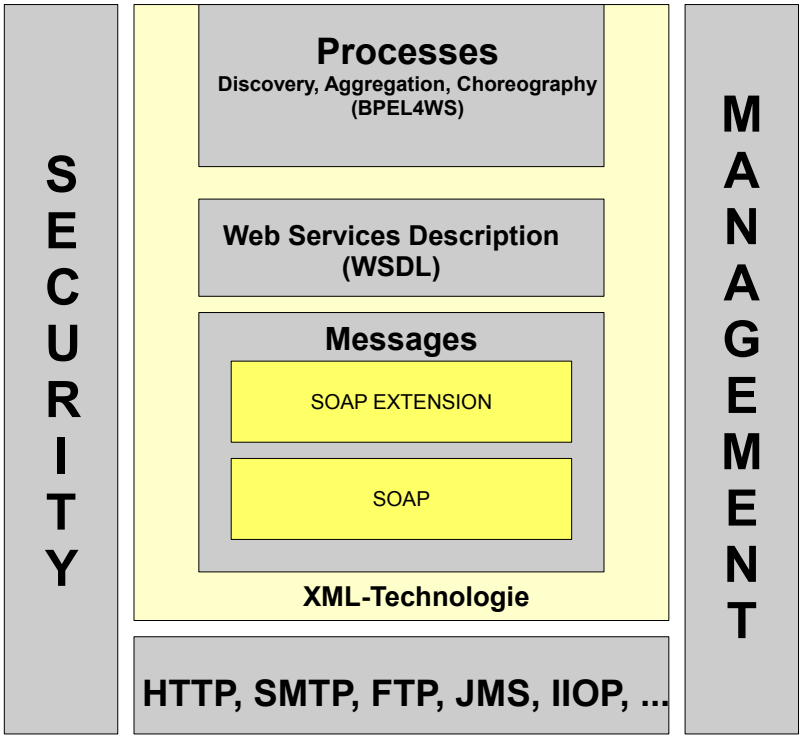


Abbildung 2.2.: Web Services Stack

2.2. Geschäftsprozessmanagement und BPEL

Die Web Services werden einzeln abgetrennt veröffentlicht und sind für den Publikum undurchsichtig wegen ihren hoch Geschlossenheit. Die Standards und Spezifikationen von Web Services definiert nur einen gemein Format für Interaktionen, es wird keine standardisierte geschäftliche Semantik in Businessanwendung für den Web Services definiert. Eine Orchestrierung wird durch Kombination von interne und externe Anwendung bzw. Web Services verknüpft. Eine Orchestrierung wird auch in Business Sicht als ausführbar Geschäftsprozess genannt. Ein komplex und funktionale Geschäftsprozess werden mehr Web Services in einen sinnvolle Ordnung verbinden und die Nachrichten-basierte Interaktion zwischen Web Services spezifizieren. In weitere Anschnitte werden die Modellierung und Managementaspekte von Geschäftsprozess erklärt, die Web Services Standard für Prozessbeschreibung sowie BPEL4WS [OAS] und deren Struktureigenschaft sowie Kernelement werden je nach Schwerpunk kurz erzählt.

2.2.1. Geschäftsprozessmanagement

Aufgrund den steigenden Konkurrenz in Gewerbe wird derer Geschäftsprozess einen mehrere wichtige Ressourcen im Unternehmen betrachtet. Ein Geschäftsprozess besteht aus einer zusammenhängenden abgeschlossenen Folge von Tätigkeit, die zur Erfülung einer betrieblichen Aufgabe notwendig sind [Ros06]. Eine flexible und effiziente Geschäftsprozess schafft den Unternehmen mehr Konkurrenzfähigkeiten. Der Geschäftsprozessmanagement oder »Business Process Management (BPM) « beschäftigt sich mit Planen, Modellierung, Verifizierung, Durchführung und Überwachung der Geschäftsprozess. Es wird im Trend mehrere Vorteilen gebracht, wenn die Unternehmen eigne Teilprozess nach externe Unternehmen ausgelagert, um eigne Kernkompetenzen zu konzentrieren. Die Auslagerung von Bereichen auf ein anderes Unternehmen, die beispielsweise für die Durchführung der Bankgeschäfte oder Finanzdienstleistungen wesentlich sind, wird als Business Prozess Outsourcing genannt [Sch08a].

In dieser Arbeit werden die betrieblichen Anwendungen und technische Implementierung von »Process View« aus [Str09] erweitert, die einen nach bestimmte Zweck verarbeitete Geschäftsprozess für den Partner oder Kunden z.B in einen Szenario von Outsourcing oder Prozessabfrage anbieten können.

Workflow Management

Workflow Modell ist einen Prozess Modell oder Teile davon, die in einen Rechnerumgebung automatisiert werden können. Eine Workflow Modell konzentriert auf den informationstechnischen Automation des Prozessmodelles. Es wird eine Instanz von Workflow Modell als Workflow so wie Prozess als Instanz von Prozessmodell genannt [LR00]. Unter Workflow Management versteht man die Zusammenfassung von alle Aufgaben, die den Optimierung, Modellierung, Spezifizierung, Simulieren, Ausführung und Monitoring des Workflows realisieren müssen. Ein Workflow-Management-System ist einen Anwendungssystem, die durch Management von den sequenzielle ausführbare Arbeitsaktivitäten und Aufrufen der entsprechende organisatorische und/oder informationstechnische Ressourcen den funktionale Automatisierungen von Geschäftsprozess unterstützen [H+94].

Die Abbildungen 2.3 aus [H+94] stellt die Workflow Reference Modell von Workflow Management Coalition (WFMC) [WFM] dar, die einen prinzipiell Architekturmodell für Workflow-Management-Systeme entwerft. Die Architektur identifiziert die wichtigen Komponenten und Interfaces. In dieser Arbeit wird die Prozess-Definitionswerkzeug als Aufgabegebiete eingesetzt. Die alle Elementbeschreibungen wurden in [H+94] ausführlich aufgeklärt.

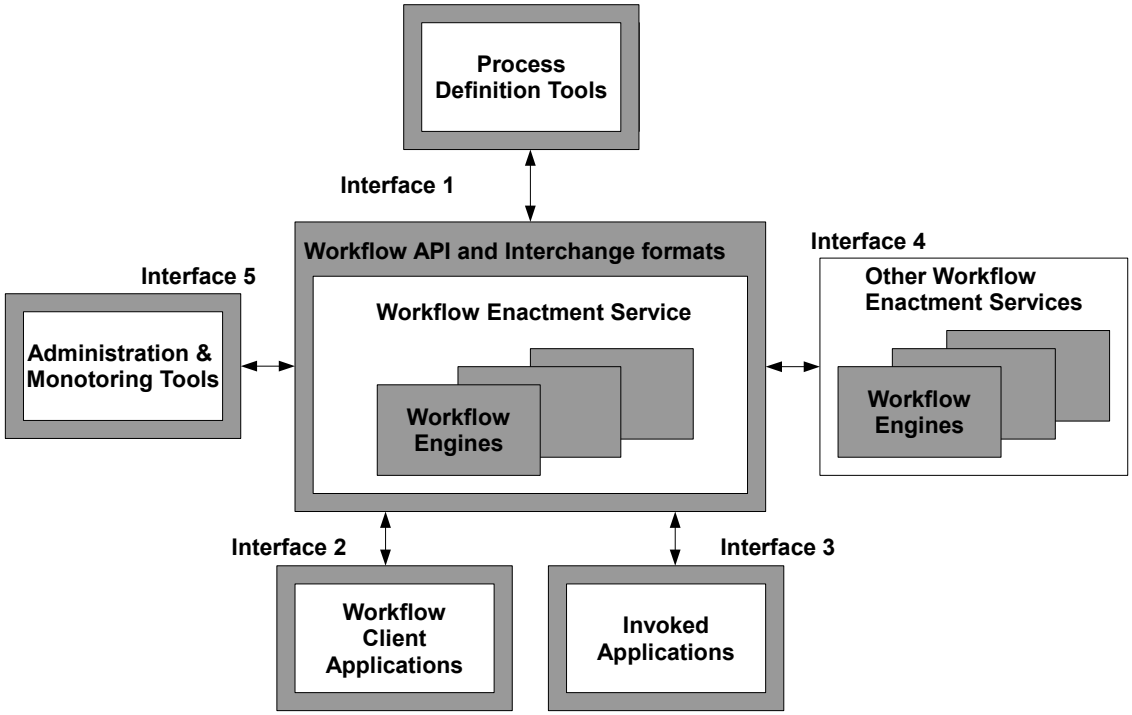


Abbildung 2.3.: Workflow Reference Modell

2.2.2. Business Process Execution Language (BPEL)

Business Process Execution Language für Web Services (oder Abkürzung als BPEL4WS) ist einen XML-basierte formale Spezifikationssprache, die den Geschäftsprozess als Orchestrierung von Web Services modelliert und deren geschäftliche Kommunikationsprotokolle beschreibt [WCL+05] [DJMZ05]. Aus historische Sicht wird BPEL4WS aus WSFL [L+01] und XLANG [Tha01] gemeinsam kombiniert und weiter entwickelt. Die Spezifikation von BPEL4WS wird von OASIS [OAS] standardisiert. Die Komposition aus mehrere Web Services und deren Semantik werden durch BPEL4WS als einen ausführbare Geschäftsprozess in den rechnerische Umgebung spezifiziert, die BPEL-Prozess werden in mehrere Laufzeitumgebungen ausführen und simulieren können, die die BPEL-Prozesse als neutrale Datenformat akzeptieren. Einer BPEL-Prozess können mit mehrere Partner durch Schnittstelle kommunizieren, ob Web Services von dem Partner durch BPEL4WS realisiert. Wegen die hoch Komplexität von Alltagsgeschäft, BPEL4WS bietet auch viele geschäftliche Interaktion-Protokolle an, die den komplexe Kommunikationsszenarien in den reale Welt unterstützen.

Die ausführliche Details von BPEL4WS wird in [AAA+] aufgeklärt, hiermit werden die wichtige Kernelemente von BPEL in Überblick erklärt. Einer BPEL-Prozess wird wie andere Spezifikationen von Web Services gut strukturiert. In Anhang wird einen einfache und

nutzbare Anwendungsfall von BPEL-Prozess sowie A.2.1 auf Seite 62 aus [IBM] entnommen. Es sollen einen voraussichtliche Verständnis für den Geschäftsprozessmodellierung bewirken, in folgende Abschnitte wird jeweils wichtiger Kernelement in den ausführbare Geschäftsprozess nach einander erzählt.

Partner Links

In BPEL4WS werden die Web Services als Partner Links modelliert, die mit den Geschäftsprozess durch den Schnittstellen bzw. <PortType> kommunizieren müssen. Jeweils <partnerLink> gehört zu einen bestimmte partnerLinkType von WSDL-Dokument des Geschäftsprozesses. Eine <partnerLinkType> beschreibt die Kommunikationsrelation zwischen zwei Web Services durch definierte Rollen von jeweils Teilnehmer. Eine BPEL-Prozess können mehrere <partnerLink> haben, die aus einen <partnerLinkType> des BPEL-Prozesses sind. Listing 2.1 zeigt, wie die Deklaration von <partnerLink> in BPEL-Prozess vereinbart wird.

Listing 2.1 Partner Links Deklaration aus [AAA+]

```
<partnerLinks>
    <partnerLink name="NCName"
      partnerLinkType="QName"
      myRole="NCName"?
      partnerRole="NCName"?
      initializePartnerRole="yes|no"? />+
</partnerLinks>
```

Variables

Das Geschäftsprozess kommuniziert mit dem Partnern durch Austauschen von Nachrichten, die in BPEL4WS als Variable in den jeweils Zustandsspezifikation des Prozesses benötigt werden. Das Datenformat von Interaktion wird in WSDL-Dokument von BPEL-Prozess als <wsdl:message> deklariert. Die Variablen in BPEL-Prozess verborgen die ausführliche Darstellung von solche »Messages« sowie Nachrichten, die den wichtigen Element für den ganz Geschäftsprozess sind. In den Prozess wird die alle Detailinformation bei Interaktion zwischen den <portType> druch Variable abgeschlossen. Solche Information können die bekommenden Nachrichten oder gesendete Nachrichten von den Partnern sein. Listing 2.2 zeigt, wie die Deklaration von <Variable> in BPEL-Prozess durch WSDL messageType vereinbart wird.

Listing 2.2 Variables Deklaration aus [AAA+]

```
<variables>
    <variable name="BPELVariableName"
       messageType="QName"?
       type="QName"?
       element="QName"?>+
       from-spec?
    </variable>
</variables>
```

Correlation Sets

Das Geschäftsprozess können mehrmals für jeder neu Anwendungsfall durch Instanziierung der Startaktivität jeweils neu instanziiert werden. Die Nachrichten werden nicht nur nach angepasste Schnittstellen geschickt, es müsst auch für den richtige und korrelierte Prozessinstanz identifiziert werden, den dieser anpasste Schnittstelle aufruft. Es wird benötigt so, dass jede Nachricht durch weiter sinnvolle Deklaration bzw. Korrelationseigenschaft mit entsprechenden Instanz eindeutige kombiniert. BPEL4WS bietet einen Deklarationsmethode für solche Korrelationsbedarf an, die die alle korrelierte Zusammenhang in den Prozessinstanz spezifizieren. Eine definierte und erweiterbare Eigenschaft für den Nachrichten in einen korrelierte Zusammenhang wird als <correlationSet> genannt. Die Deklaration von <correlationSet> wird wie die Variable in den BPEL-Prozess definiert. Der Name von einen <correlationSet> sind globale eindeutig und wird in Interaktion zwischen den Web Services nicht verändert.

Basis Aktivitäten

Die Aktivitäten in BPEL4WS realisieren die Logik und Semantik von Geschäftsprozess, es wird die funktional Operation oder geschäftliche Tätigkeit durch jeweils Aktivität ausgeführt. Alle Aktivitäten von BPEL4WS werden nach zwei Klassen klassifiziert, sowie elementare Aktivität und strukturierte Aktivität. In dieser Abschnitt werden alle elementare Aktivitäten kurz vorgestellt, die einen einzeln Arbeitsschritt von BPEL-Prozess beschreiben. In den weitere Abschnitte werden die alle strukturierte Aktivitäten erläutert.

<invoke> Es wird einen <invoke> Aktivität benutzt, um einen von Services-Anbieter gebotene Operationen aus Web Services zu aufrufen.

<receive> und <reply> Eine <receive> Ativität spezifiziert den entsprechende Information z.B <partnerLink> für den Aufnahme von Nachrichten, <portType> und <operation>, die für den Partner angeboten werden. Eine <reply> Aktivität sendet den Antworten für den vorige Abfrage, die z.B durch <receive> eine reinkommende Nachricht erzeugt wird.

<assign> In einen <assign> Aktivität werden den Wert der Variable überschrieben.

<throw> Eine <throw> Aktivität ruft einen intern Fehlerbehandlung auf, wenn einen Fehler in Prozess auftritt.

<wait> Eine <wait> Aktivität verzögert einer Prozess für einen Zeitintervall oder bis einen bestimmte Zeitpunkt.

<empty> Eine <empty> Aktivität bedeutet eine leere Operation, es wird benötigt in manche Situationen z.B Fehlerbehandlung.

<rethrow> Eine <rethrow> Aktivität wird in den Fehlerbehandlung benutzt, um einer aufgetauchte Fehler erneute zu behandeln.

<exit> Eine <exit> Aktivität beendet eine Prozessinstanz direkt.

<extensionActivity> Eine neue Aktivitätstype wird hinzugefügt in den Prozess durch direkt Platzierung in den <extensionActivity>.

Strukturierte Aktivitäten

Strukturierte Aktivitäten implementieren die logische Kontrollfluss von den Geschäftsprozess. Es können mehrere elementare Aktivitäten oder strukturierte Aktivitäten rekursive in einen strukturierte Aktivität abschließen werden. Es wird für den Mengen von Aktivitäten definiert, wie die Ausführung implementieren werden sollen.

<sequence> Die alle Aktivitäten in einen <sequence> Aktivität werden sequenziell nach einander ausgeführt

<if> Eine <if> Aktivität realisiert einen bedingte Verzweigung.

<while> Eine <while> Aktivität realisiert einen bedingte Wiederholung für den interne Aktivitäten.

<repeatUntil> Eine <repeatUntil> Aktivität realisert wie <while> Aktivität einen bedingte Wiederholung von interne Aktivitäten.

<pick> Eine <pick> Aktivität wartet einen erwünschte Angelegenheit auf, dann führt entsprechende Aktivitäten für den Auftauchen dieser Angelegenheit aus.

<flow> Eine <flow> Aktivität unterstützt den Synchronisierung und Nebenläufigkeit für den interne Aktivitäten durch Verknüpfung-Semantik.

<forEach> Eine <forEach> Aktivität können interne <scope> Aktivität mehrfach ausführen nach Zahldifferenz zwischen<finalCounterValue> und <startCounterValue> .

Scopes

In BPEL4WS wird eine Bündnis von mehrere Aktivitäten als <scope> konstruktiert, die einen transaktionale Eigenschaft besitzt. In einen <scope> wird die Kontexten von alle interne Aktivitäten spezifiziert, es definiert die interne »Variable«, »partnerLinks« und »CorrelationSet« usw. für einen lokale Bereich. Die interne Behandlungen z.b Fehlerbehandlung und Kompensationsbehandlung realisieren die wichtigen ACID-Eigenschaften von Transaktion [LR00]. Eine <scope> besetzt einen ursprüngliche Aktivität, die die normale Funktionalität von <scope> spezifiziert und realisiert.

Handlers

In BPEL4WS werden solche Behandlungen angeboten, die für komplexe Situationen in einen <scope> Aktivität den besondere Operationen z.B Reaktionen von Fehler und Rückführung von Zuständen ausführen können.

<compensationHandler> Eine <compensationhandler> Behandlung beinhaltet den Aktivitäten, die einen Kompensation implementieren sollen.

<faultHandlers> Eine <faultHandler> Behandlung führt einen reverse Aktionen von einen nicht erfolgreich Arbeitsschritten in einen <scope> Aktivität durch, die durch einen Fehler verursacht wird.

<terminationHandler> Eine <terminationHandler> Behandlung bietet eine Operation an, die einen <scope> Aktivität gezwungenermaßen terminieren können.

<eventHandlers> Eine <scope> Aktivität können mehrere <eventHandler> haben, die gleichzeitig laufen. Wenn eine entsprechende Angelegenheit auftritt, werden die Aktivitäten in den angepasste <eventHandler> ausgeführt.

2.3. XML-Technologie

Die Extensible Markup Language (XML) ist einen strukturierte Darstellungsformat von Textdaten. Die XML-Spezifikation wird von W3C [YBP+04] standardisiert, die einen Metasprache für den XML-Anwendung definiert. Ein XML-Dokument wird durch einen Baumstruktur dargestellt und besitzt einen gute Lesbarkeit als binäre Daten für den Menschen. Ein XML-Dokument mit standardisierte Textzeichen können in verschieden Plattform und Programmierung-Modell unabhängig ausgetauscht und verarbeitet. XML ist eine selber beschreibbare Metasprache und die Syntax wird einfach und logisch definiert. Die Web Services Spezifikationen basieren auf den XML-Spezifikation wegen die gute Performance von XML-Technologien. In folgende Abschnitte werden die wichtige Eigenschaft und Technologien von XML kurz aufgeklärt, die einen Zusammenhang mit dieser Arbeitsgebiet haben.

2.3.1. XML Namespaces

Die Struktureigenschaft und flexible Erweiterbarkeit von XML bietet so an, dass die Name von Element und Attribute in den Dokumentkontext beliebige definieren können, es gibt viel Vokabular in einen Bereich freiwillig zu auswählen. Es wird die Fehlern bei Namenszuweisung aufgetaucht, wenn zwei Vokabulare in einen Dokument kombiniert und die Name sich in Konflikt befinden. Es wird in XML einen XML Namespace [BHLT06] und mit Abkürzung »xmlns« in Dokument angeboten, die die Namensraum von Element und Attribute eindeutig beschränken. Einen XML Namensraum wird durch einen Uniform Resource Identifier [BLFM98] Adresse hingewiesen, die verfügbare Namen für Elemente und Attribute werden durch in [BHLT06] deklarierte Verfahren in den Namensraum zugewiesen. Eine erweiterte qualifiziert-Name besteht aus Präfix bzw. Namensraum-Präfix und lokale Name.

2.3.2. XML Schema

Die von W3C entwickelte XML Schema [FW04] definiert die Struktur einer XML-Dokument. XML Schema ist eine XML-basierte Dokumentdefinition, ein XML Schema Dokument wird auch als XML Schema Definition mit Abkürzung als XSD genannt. Aus einen XSD-Dokument werden mehrere validierte Instanzen bzw. XML-Dokument erzeugt. XML Schema ist einen verbesserte Version von DTD für den XML-Dokument, es können einen flexible Erweiterbarkeit für den Struktur und mehrere reichhaltigen Datentypendefinition unterstützen.

Einfaches Element

In einer XML Schema Dokument werden die alle Elementen definiert. Ein XML-Dokument besteht aus den Elementen, in den mehrere interne Elementen geschachtelt werden können. Ein einfaches Element ist einen XML-Element, die keine weiter Unterelement oder Attribute aber nur den Text beinhaltet. Der Text in einfach XML-Element können mehrere Datentypen sein, deswegen wird die Datentype von Text auch definiert in den Elementdefinition. In XML Schema werden bereits mehrere Datentypen vordefiniert, z.B »xs:string«, »xs:integer« und »xs:boolean«. Jedes Element in XML Schema besitzt einen eindeutige Name. Alle Attribute in XML-Dokument wird als einen einfach Element in XML Schema definiert, einen einfach Element in XML-Dokument besitzt nur Text aber keine Attribute, solche einfach Element für Attributen werden nur für komplexe Elementen definiert. Bei einfach Elementdefinition werden die Einschränkung von Wertzuweisung von Element und Attribute durch »xs:restriction« definiert.

Komplexes Element

Ein komplexes Element in XML-Dokument ist den Element, die andere Element oder Attribut beinhaltet. Ein komplex Element können leer aber mit Attribut sein. Ein komplexe

Element wird durch Kombination von mehrere einfach Elementen mit den Indikatoren sowie »xs:sequence« , »xs:all« oder »xs:choice« definiert.

2.3.3. XML Path Language (Xpath)

Die XML Path Language (Xpath) [CD+99] ist einen XML-orientierte Abfragesprache, die die Elementen in einen XML-Dokument sowie in einen Baumstruktur navigieren und adressieren können. Xpath wird von W3C entwickelt und als einen Grundstein für den XSLT Standard [W3C99] fungiert. Die Xpath benutzt einen Abfragebefehle wie andere Abfragesprache z.B SQL in Datenbank, um die Zielobjekten zu adressieren. Eine Xpath-Befehle bzw. Xpath-Ausdruck besteht aus einen Lokalisierung-Pfad mit »/« Zeichen, die wie die Ordnerpfade in Ordnerverwaltung die Adresspfade des Zielobjekts navigieren. Es wird optional bei mehrere Objekten einen Prädikaten nach den Pfade benötigt, um die zahlreiche Elementen zu beschränkten und Zielobjekt zu identifizieren.

Listing 2.3 Ein einfaches XML-Dokument

```
<A>
   <B1>
      <C1  value="1">
         <D1>
         </D1>
      </C1>
   </B1>
   <B2>
       <C2  value="2">
          <D2>
          </D2>
       </C2>
   </B2>
   <B3>
       <C3  value="3">
           <D3>
           </D3>
       </C3>
   </B3>
</A>
```

Knotenauswahl

Die Konten werden ausgewählt durch Verfolgen den Lokalisierung-Schritten, der Prozessor lest jeweils Schritt von links nach recht und zusammenpasst die Element nach die Gleichheit von Name oder Erfüllung von den Bedingungen. Eine Achse-schritt gibt die alle erreichbare Konten von den aktuelle Knote nach die Spezifikation der Achsen. In Xpath-Spezifikation bietet einen Mengen von Achse-Schritt an, die einen Vielfalt von Durchsuchung in Baumstruktur erfüllen können.Durch Kontentest wird die Elementauswahl in einen Achse-Schritt weiter eingeschränkt. Eine Knotentest ist einen Kontext-basierten Kondition in den XML-Dokument,

den die alle ausgewählte Elementen erfüllen müssen. Listing 2.4 zeigt beispielsweise die Kontenauswahl aus den einfach XML-Dokument wie Listing 2.3.

Listing 2.4 Beispiele von der Knotenauswahl

```
/A/B2/C2                    --- Das Element C2 in B2 wird selektiert.
/A/B1/following-sibling::*  --- Die nachkommende Brüder B2 und B3 werden ausgewählt.
```

Prädikate

Durch Prädikaten werden die Auswahl von Elementen weiter eingeschränkt wie beispielsweise in Listing 2.5. In Xpath werden viele Funktionen und entsprechende Operatoren angeboten, die auf mathematische Kontext und Zeichenketten beziehen. Eine Prädikaten werden die erreichbare Knoten filtern, die durch den Lokalisierung-Pfade identifiziert werden.

Listing 2.5 Beispiele von den Prädikaten

```
//*[@value='1']    --- Das Element mit Attribut value="1" sowie der Knote C1 wird selektiert.
//*[name()='B3']   --- Der Knote B3 wird ausgewählt.
```

2.4. XML-Parser

In der praktisch Arbeit wird die XML-basierte BPEL-Prozess verarbeitet. Eine durch BPEL4WS beschriebene Geschäftsprozess wird als einen strukturierte XML-Dokument in den Implementierungssoftware eingegeben. Das Programm nimmt das XML-Dokument ein und stellt die Information des Dokuments für den verschiedene Anwendung dar. In diese Abschnitt wird die Programmierung-Aspekt von Web Services bzw. BPEL4WS diskutiert. Die verschiedenen XML-Prozessoren sowie XML-Parser für den verschieden Programmiersprachen zur XML-Verarbeitung werden kurz erklärt.

2.4.1. DOM

Das Document Object Model (DOM) [W3Ca] ist einen von W3C standardisierte Spezifikation von Programmierschnittstelle für verschiedenen Programmiersprachen zur XML-Verarbeitung. Das XML-Dokument wird eingelesen und es wird in den Hauptspeicher einen DOM-Baum erzeugt, die die Informationen von XML-Dokuments als einen Baumstruktur darstellt. In den praktisch Implementierung wird der »process« Element in BPEL-Prozess als Root-Knoten von DOM-Baum traversiert. Es unterstützt den Zugriffen auf den DOM-Baum bzw. »Document«, die Struktur von »Document« können interaktive modifiziert werden. Eine Nachteil von DOM ist es, eine Speicherbedarf wird hoch, wenn die verarbeitet XML-Dokument sehr groß ist.

2.4.2. SAX

Simple API for XML (SAX) [Meg] wird benutzt, um die XML-Dokument sequenzielle zu durchlesen. Es wird keine Objektmodell wie DOM erzeugt. SAX ist Ereignis-basiert, es wird eine Mengen von Ereignissen von SAX definiert, die bei sequenzielle Einlesen von XML-Dokuments ausgelöst werden. Die Entwickler können individuell Behandlung für den Ereignisse definieren, z.B für den Elementen und Text. Wenn ein Ereignis auftritt, wird die Behandlung aufgerufen und entsprechende Ergebnis ausgegeben. SAX wird in meisten Fällen nur angewendet, um die Informationen in XML-Dokument zu anzeigen. Es unterstützt keinen interaktive Modifikationen durch Zugriffen von Objektmodell.

2.4.3. XML-Verarbeitung mit Java

In den praktische Implementierung dieser Arbeit werden die zwei Programmierschnittstelle bzw. JAXP (Java API for XML Processing) [Gla] und dom4j (DOM for Java) [OSP] in Java-Plattform [Micb] angewendet, um die XML-basierte BPEL-Prozessen zu verarbeiten. Die Weiterentwicklung und Qualitätssicherung von »Process View« aus [Stro9] werden in Eclipse-Entwicklungsumgebung [Fou] durchgeführt, es wird für den Entwurf und Validierung von BPEL-Prozess den Extension »Eclipse BPEL Designer« [Tea] angewendet.

3. Process View

In diesem Kapitel werden die erweiterte Konzeption und Fortsetzung von »Process View« auf Basisarbeit von [Str09] erklärt. In dieser Arbeit bietet einen zielorientierte Mechanismus an, die den BPEL-Prozess nach konkrete Verarbeitungsschritten auf den Prozesselementen modifizieren können. In Abschnitt 3.1 wird ein generell Muster für den »Process View« vorgestellt, es sollen einen allgemeine Konzeptdarstellung von Verarbeitungsmechanismus für den Lesern anschaffen. Die praktische Implementierung bzw. Methode wird in Abschnitt 3.2 auf Seite 27 ins Details erzählt. Es wird viel vordefinierte Operationen in den Applikation spezifiziert, mit den viel individuelle Funktionen von »Process View« durch Kombination realisieren. In Abschnitt 3.2.3 auf Seite 31 werden die jeweils grundlegend Operation aus Arbeit von [Str09] in dieser Arbeit vervollständigt und mit Codebeispiel erklärt.

3.1. Konzept

Die Generation von einen abstrakte Sicht von Basisprozess wird als einen Transformator-Programm von BPEL-Prozess entwickelt, der einen XML-basierte Rules-Dokument bzw. Verarbeitungsmechanismus und einen Basisprozess einlest und eine neue oder abstrakte Teilprozess ausgeben. Die Abbildungen 3.1 stellt das Verarbeitungsprinzip von Transformation dar, die den View Generation Pattern von [Str09] basiert. In den weitere Abschnitten wird das jeweils relevant Bauelement in den Konzeptskizze ausführlich erläutert.

Es wird in den Implementierung das Rules-Dokument als einen Kernelement betrachtet. Der in Java geschriebenen Programm führt die Operationen in Rules-Dokument durch. Ein Basisprozess können einen bereits kommentierte Prozess oder einen ursprüngliche Geschäftsprozess sein. Ein Rules-Dokument sollen einen sinnvolle Ergebnis für den Basisprozess realisieren, deswegen es werden die technische Verständnissen auf den WS-BPEL und betriebliche Wissen von Geschäftsprozess in Anwendungsszenario für den Erstellung eines Rules-Dokuments benötigt. In der Arbeit wird »Process View« als einen Werkzeug für den geschäftlichen Prozessspezialisten entworfen. Die Spezialisten müssen sich die Aktivitäten in den Geschäftsprozess und die betriebliche Kommunikationen zwischen Partnern auskennen, um einen anwendungsspezifische Sicht von Basisprozess zu bekommen.

Es wäre aufwendig für den Personen, die trotz die Geschäftsprozess sich gut auskennen aber nicht vertraut mit BPEL-Prozess sind, einen Rules-Dokument manuelle zu generieren. Es fordert bei Erstellung von Rules-Dokument außer die Kenntnisse von WS-BPEL noch die richtige Verständnissen von Operationen, mit den mehr Verarbeitungsmöglichkeiten

durch sinnvolle Schachtelung realisieren können. Es wird in dieser Arbeit ein paar konkrete Funktionen von »Process View« angeboten, die einen entsprechende Rules-Dokument automatisch generieren können. Solche vordefinierte Funktionen realisieren den bestimmte Zielen bzw. Anforderungen von »Process View«. Es wird in Kapitel 4 die jeweils Funktion diskutiert und mit den Anwendungsfälle veranschaulicht.

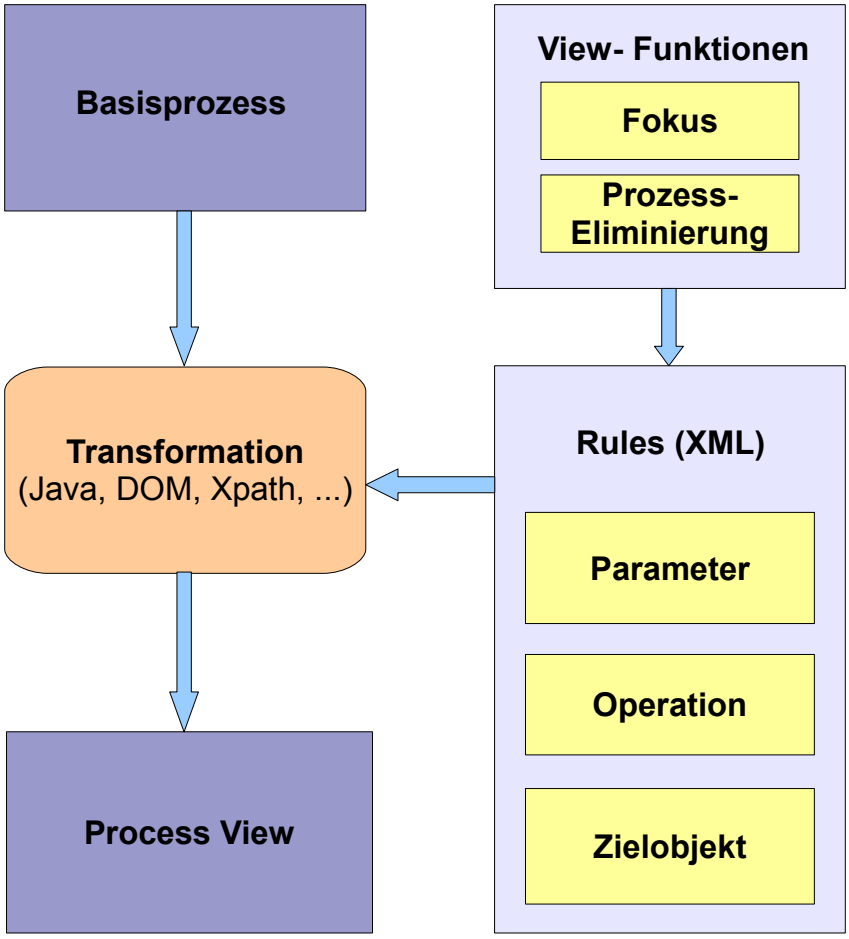


Abbildung 3.1.: Process View Generation

Um oben genannte Konzept zu realisieren, es wird den mehr detaillierte Arbeiten gebraucht. Der ausführbar Prototyp in Eclipse-Umgebung stellt bereits die Implementierungsmethode dar, der in der Arbeit von [Stro9] entwickelt wird. Es wird notwendig, die Basismethode und grundlegende Operationen nochmal in dieser Arbeit kurz zu erzählen, weil sie mit den in Kapitel 4 aufklärte View-Operationen einen wichtigen Zusammenhang haben und dabei eine elementare Rolle für den Vervollständigung von »Process View« spielen.

3.2. Methode

In dieser Abschnitt werden die Konstruktion von Rules-Sprache und die Transformation in den Programm in Details erklärt. Das manuelle erstellte Rules-Dokument kontrolliert die Transformationen in den Programm, sie spezifizieren die Zielobjekten und entsprechende Operationen für die Transformation. Die Transformation in den Programm wird durch Algorithmen implementiert. Es wird in Abschnitt 3.2.3 die wichtige Verarbeitungsalgorithmen in Pseudocode erklärt.

3.2.1. Erweiterte Rules-Sprache

In dieser Arbeit wird die Rules-Sprache aus [Str09] erweitert, die Basisstruktur von Rules-Sprache bleibt unverändert, es wird dabei manche neue Elementen hinzugefügt. Für einen ausführliche Erklärung von Basisstruktur können in der Arbeit von [Str09, Seit 30 bis 33] gefunden werden. Einen entsprechende XML Schema Dokument von erweitert Rules-Sprache wird in A.1 auf Seite 60 angehängt.

Es wird in den neu Rules-Sprache für einen Rules-Dokument so definiert, dass jeweils Rules-Dokument aus einen Regelgruppe bzw. <rules> besteht. Für den <rules> wird einen Benennung als einen Attribut sowie »name« hinzugefügt. In einen <rules> beinhaltet es einen Parameterspezifikation bzw. <parameter> und einen Sequenz von mehrere Regeln bzw. <rule> Elementen.

Parameter

Die Parameterspezifikation besteht aus zwei Operationen sowie Aktivitätsgruppierung <aggregateOpaque> und Prozessreinigung <cleaning>. Es wird nach die Transformation von den Regeln abgefragt. ob die weitere Verschönerung auf den resultierend BPEL-Prozess durchführen dürfen. Wenn den <value> auf »false« gesetzt wird, wird die Operation nicht durchgeführt. Es wird in Anschnitt 3.3 einen ausführliche Erklärung von diese zwei Operationen mit Anwendungsfälle und Pseudocode gegeben.

<aggregateOpaque> Es wird nach den Ausführung von <aggregateOpaque> alle nach einander liegende und undurchsichtige Aktivität in einen sinnvolle Struktur gruppiert.

<cleaning> Die Operation »cleaning« bedeutet einen Eliminierung von alle unnötigen Elementen bzw. Artefakten in den resultierend BPEL-Prozess. Die strukturierte Aktivitäten werden gelöscht, die keine Aktivität darin beinhaltet. Die alle Markierung sowie »@tag« werden gelöscht, die nicht genutzte Deklarationen von BPEL-Prozess sowie <import>, <extension>, <partnerLink>, <variable>, <link> und <correlationSet> werden reinigt, um die übrige Prozessinformationen weiter zu verborgen. Eine genaue Spezifikation sieht man in [Str09, Seit 62 bis 63].

Rule

Außer Element <parameter> beinhaltet es in den <rules> noch einen Sequenz von mehrere Regeln <rule>, jeweils <rule> besteht aus einen Operationsspezifikation bzw. <actions> und die Zielspezifikation bzw. <targets>. Ein <rule> besetzt zwei Attributen sowie »name« und »apply«:

name Jeder <rule> Element wird durch eigne Benennung eindeutig identifiziert.

apply Ein boolesch Attribut von <rule>, es wird voreingestellt als »true«, es ist ein Schalter für den <rule>.

Operationen

Das <actions> Element in den <rule> können mehrere Operationen in einen sinnvolle Ordnung beinhaltet. In den Arbeit von [Str09] wird drei grundlegende Operationen für den <actions> Element entwickelt, die wird in Abschnitt jeweils mit entsprechende Erweiterungen ins Details erzählt. In dieser Arbeit wird weitere Operationen und Vervollständigungen entwickelt, die in Abschnitt 3.3 ausführlich erklärt werden. Die Aktionsspezifikationen stehen wie in folgende Auflistung für den <actions> Element zu Verfügung:

<actionOmit> Die Aktion <actionOmit> löscht die ausgewählte Aktivitätselement in den BPEL-Prozess. Das Attribut <preserveChildren> erhaltet die durch »Perserve« kommentierte Kinderaktivitäten in einen strukturierte Aktivität, wenn diese strukturierte Aktivität gelöscht werden sollen und dieser Attribut als »true« gesetzt wird. Das Attribut <preserveTransitionConditions> wird dabei benötigt, um die »TransitionConditions« in einen ausgewählte Aktivität zu beschützen. Es wird in dieser Arbeit so vorschlägt, dass Löschung von Aktivität bei Bedarf von <preserveTransitionConditions> die Aktion <actionOpaque> benutzen.

<actionOpaque> Die Aktion <actionOpaque> verbergt die Aktivität durch Löschung von entsprechende Informationen. Mit dem Attribut <preserveAttribute> wird es den Attribute von Zielaktivität nicht gelöscht.

<actionSetAttributeTo> Durch diese Aktion wird die Attribute in den ausgewählte Aktivität modifiziert. Das Attribut <attributName> weist den Name von Attribut in den Zielaktivität hin, das Attribut <value> setzt den Wert von entsprechend Attribut auf. Wenn der <value> null ist, wird das Attribut gelöscht.

<addPreseve> Die Aktion <addPreserve> fügt die Information »Preserve« für den Aktivitäten hinzu, um die entsprechende Aktivitäten zu beschützen.

<addTag> Die Aktion <addTag> fügt die Information »@tag« für den Aktivitäten hinzu, um die entsprechende Aktivitäten zu dokumentieren. Die Attribut »value« speichert die kommentierte Informationen.

Zielobjekt

Das <targets> Element in den <rule> spezifiziert den Auswahlkriterium und navigiert den Zielobjekten in den BPEL-Prozess, die durch den Operationen in <actions> modifiziert werden sollen. Es wird in dieser Arbeit die Lokalisierung und Selektion von Zielobjekten durch Xpath-Ausdrücke implementiert. Ein Element in BPEL-Prozess wird entweder durch die Anpassung von <tag>, <attribute> und <type> oder durch beliebige logische Kombinationen sowie <or>, <and> und <not> selektiert. Durch der Kombinationsalgorithmus in Abschnitt 4.1 wird die <targets> Elemente zum Xpath-Ausdrücke mit logische Prädikaten umgewandelt. Es wird in dieser Arbeit die mehrfache verschachtelte Kombinationen mit logische Symbolen unterstützt. Die Anwendungsprinzip von <tag>, <attribute> und <typ> wird in folgende Aufzählung kurz erklärt:

<tag> Ein <tag> Element wird bei Auswahl einer durch »@tag« kommentierte Objektelement in BPEL-Prozess angewendet. Das Attribut »tagName« spezifiziert den Inhalt von »@tag«, es wird die alle Konstrukten (auch die nicht durch »@tag« kommentierte Elementen) in den BPEL-Prozess ausgewählt, wenn den »tagName« durch Sternsymbol wie »*« zugewiesen wird.

<attribute> Ein <attribute> Element wird bei Auswahl der Elementen durch den Wert in entsprechende Attributen angewendet. Das Attribut »attributeName« in <attribute> spezifiziert den Name eines Attributes. Das Attribut »value« in <attribute> spezifiziert den Wert des entsprechendes Attributes in Zielobjekten.

<type> Ein <type> Element wird bei Auswahl der Element durch die Typisierung von Aktivitäten in BPEL-Prozess angewendet. Das Attribut »typeName« in <type> spezifiziert die Type des Zielobjektes, z.B »invoke«, »flow«. Es wird alle Aktivitäten (auch die strukturierte Aktivitäten) ausgewählt, wenn der »typeName« durch Sternsymbol wie »*« spezifiziert wird.

3.2.2. Beispiel von einem Rules-Dokument

Listing 3.1 zeigt, wie einen Rules-Dokument für einen konkret Anwendungsbeispiel wie Reisebüroprozess aus [IBM] konstruiert werden können. In dieser Beispielfall sollen einen Fokussieren auf den Aktivität wie »Confirmation« generiert werden. Dieser Rules-Dokument wird von den in Abschnitt 4.3.1 vorgestellte Fokussieren-Funktion automatische generiert.

Listing 3.1 Beispielfall von einem Rules-Dokument

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:rules xmlns:tns="http://www.eclipse.org/bpel/views/rules"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.eclipse.org/bpel/views/rules Rules.xsd" name="5">

        <parameter>
            <aggregateOpaque value="true"/>
            <cleaning value="true"/>
        </parameter>

        <rule apply="true" name="addPreserve">
                <actions>
                        <addPreserve />
                </actions>
                <targets>
                  <or>
                        <attribute attributeName="name" value="Confirmation" />
                        <attribute attributeName="name" value="CheckFlightReservation" />
                        <attribute attributeName="name" value="CheckHotelReservation" />
                        <attribute attributeName="name" value="While" />
                        <attribute attributeName="name" value="HiddenSequence" />
                        <attribute attributeName="name" value="CheckCarReservation" />
                        <attribute attributeName="name" value="EvaluateReservationResult" />
                        <attribute attributeName="name" value="Reply" />
                  </or>
                </targets>
        </rule>

        <rule apply="true" name="Omission">
                <actions>
                        <actionOmit />
                </actions>
                <targets>
                        <tag tagName="*" />
                </targets>
        </rule>

</tns:rules>
```

Die Abbildungen 3.2 stellt die Ausführung von Listing 3.1 graphisch dar, wie die Aktionen in den Rules-Dokument durchgeführt werden. Die Aktivitäten in den abgerundetes Rechtecke werden durch die Aktion <addPreserve> beschütztet, die übrige Aktivitäten werden wegen keine Erhaltung durch die Aktion <actionOmit> gelöscht.

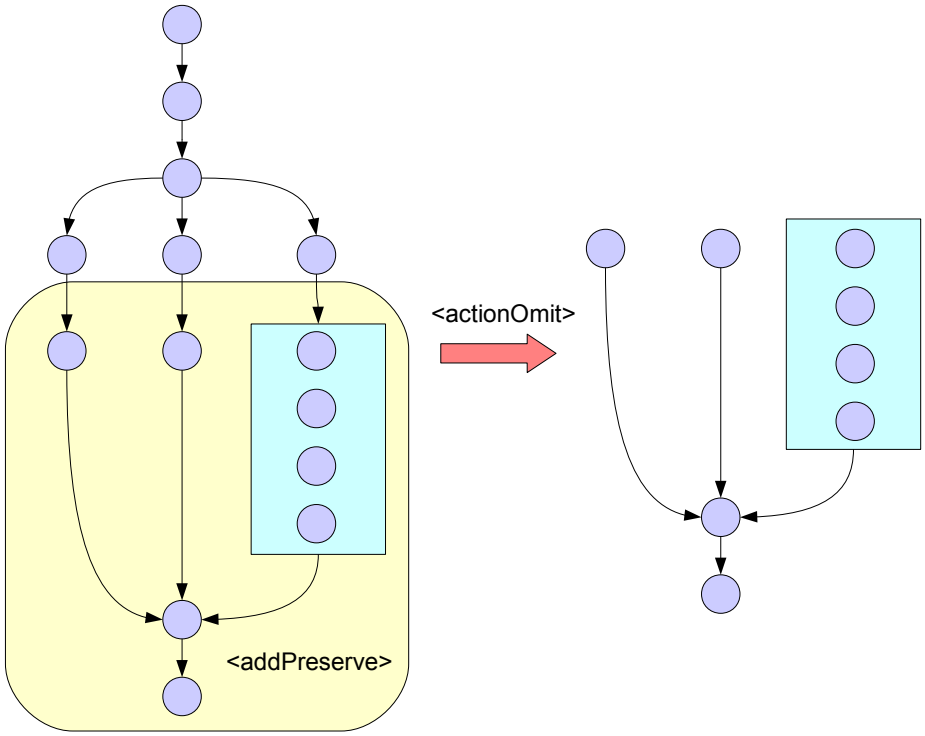


Abbildung 3.2.: Die graphische Darstellung von der Ausführung des Rules-Dokuments

3.2.3. Transformationsmechanismus

Die Transformation in der praktische Implementierung wird durch den Algorithmen realisiert. Ein gleichwertig Algorithmus wird die Verarbeitungsprinzip von Transformation durch Pseudocode und natürliche Sprache deklarativ erklärt. Listing 3.2 zeigt, wie der Algorithmus aus den Arbeitsschritten konstruiert wird, um die Transformation zu realisieren.

Der Algorithmus lest das Basisprozess und das Rules-Dokument ein, dann es wird zwei entsprechenden DOM-Bäume in den Speicher generiert. Nach den Lokalisierung von jeweils Zielobjekt durch die Methode »searchTargetsUsingXpath()« werden die definierte Operationen durchgeführt. Diese Methode benutzt die Xpath-Technologie unter Java, jeder <targets> Element können als einen ausführbare Ausdruck für den Xpath-Prozessor durch einen rekursiv Algorithmus umgewendet werden. In Abschnitt 4.1 wird der Algorithmus mit Xpath-Anwendung ausführlich erklärt. Nach den erfolgreiche Modifizierungen wird die Verschönerung auf den resultierend DOM-Baum durchgeführt, es wird dann einen XML-basierte Sicht von Basisprozess ausgegeben.

Listing 3.2 Algorithmus für die Transformation

```
function processTransformation (basisProcess, rulesDocument) return processView

 variable : processDocument //buffer of process in memory
          targets        //constructs in processDocument


     processDocument <--- basisProcess

     forEach (rule in rulesDocument)
     {
        targets <-- serachTargetsUsingXpath(<targets> in <rule>, processDocument)
        forEach (targets)
        {
           forEach (operation in <actions>)
           {
              switch (operation)
              {
              case (<addPreserve>): execute addPreserve (processDocument, targets)
              case (<actionSetAttributeTo>): execute actionSetAttribueTo (processDocument,
                  targets)
              case (<actionOpaque>): execute actionOpaque (processDocument, targets)
              case (<actionOmit>): execute actionOmit (processDocument, targets)
              }
           }
        }
     }

    forEach (operation in <parameter> from rulesDocument)
    {
      if (value is true)
      {
          switch (operation)
          {
             case (<aggregateOpaque>): execute aggregateOpaque (processDocument)
             case (<cleaning>):     execute cleaning (processDocument)
          }
       }
    }

    processView <-- Document
```

3.3. Grundlegende Operationen

In dieser Abschnitt werden die drei grundlegende Operationen und die praktische Implementierung durch die entsprechende Sourcecode vorgestellt. Das Prinzip von den Operationen wurde schon in Arbeit von [Stro9] ins Details erklärt. In dieser Arbeit wird die leichte Veränderungen und Erweiterungen von den Operation erläutert.

3.3.1. Konfiguration der Attribute

Es wird die Operation »SetAttributeTo« nach den Spezifizierte Funktionalitäten vervollständigt. Durch diese Operation können die Attribute von Aktivität beliebige verarbeitet werden. In dieser Arbeit wird der Sonderfall von diese Operation implementiert. Bei Löschung eines Attributes wird die entsprechenden Attributen von andere Aktivitäten durch Zuweisung »opaque« verborgen, die den Wertzuweisung von dieser gelöscht Attribut aufführen und einen unmittelbar Zusammenhang haben. D.h, die gelöschte Attribute und entsprechende Informationen werden nicht mehr für andere Aktivitäten sichtbar und aufrufbar.

Listing 3.3 zeigt einen leicht modifiziert Prozessblock aus [Str09]. In dieser Testbeispiel wird die Attribut »ID« von diese Aktivität »receive« durch Rules-Dokument wie Listing 3.4 gelöscht, die entsprechenden Information wie Attribut »default« in strukturierte Aktivität wird durch Ersetzung »opaque« verborgen. In Listing 3.5 wird der resultierend Prozessblock dargestellt.

Listing 3.3 Testbeispiel für »SetAttributeTo«

```
<bpel:receive name="receiveBookingConfirmation_Airline#2" ID="gold_one"
    operation="confirmBooking" partnerLink="airline#2" inputVariable="flightConf">
       <bpel:targets>
         <bpel:target
             linkName="flightBookingRequest_Airline#1_to_receiveBookingConfirmation_Airline#2">
         </bpel:target>
       </bpel:targets>
       <bpel:sources>
         <bpel:source linkName="receiveBookingConfirmation_Airline#2_to_checkData">
         </bpel:source>
       </bpel:sources>
</bpel:receive>

<bpel:sequence name="airlineContact" default="gold_one">
<!--
hier mehrere Kinderaktivitäten
-->
</bpel:sequence>
```

Listing 3.4 Rules-Dokument für das Testbeispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:rules xmlns:tns="http://www.eclipse.org/bpel/views/rules"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.eclipse.org/bpel/views/rules Rules.xsd" name="1">

        <parameter>
            <aggregateOpaque value="true"/>
            <cleaning value="true"/>
        </parameter>

        <rule name="remove_receiveAirline#2_ID" apply="true">
               <actions>

                      <actionSetAttributeTo attributeName="ID"
                             value="null" />
               </actions>
               <targets>
                      <attribute attributeName="name"
                             value="receiveBookingConfirmation_Airline#2" />
               </targets>
        </rule>
</tns:rules>
```

Listing 3.5 Ergebnis von dem Rules-Dokument

```
<bpel:receive name="receiveBookingConfirmation_Airline#2" operation="confirmBooking"
    partnerLink="airline#2" inputVariable="flightConf">
       <bpel:targets>
         <bpel:target
             linkName="flightBookingRequest_Airline#1_to_receiveBookingConfirmation_Airline#2">
         </bpel:target>
       </bpel:targets>
       <bpel:sources>
         <bpel:source linkName="receiveBookingConfirmation_Airline#2_to_checkData">
         </bpel:source>
       </bpel:sources>
</bpel:receive>

<bpel:sequence name="airlineContact" default="##opaque">
<!--
hier mehrere Kinderaktivitäten
-->
</bpel:sequence>
```

Listing 4.4 zeigt den kommentierte Sourcecode von Operation »SetAttributeTo«, die in der praktisch Arbeit vervollständigt wird.

Listing 3.6 Implementierung der Konfiguration der Attribute unter Java

```
//Es wird der Wert von dem Attribut entweder überschrieben oder gelöscht. Durch die
    Anpassung von Attribut in den DOM-Baum wird jeder Knote durchgesucht.
public void SetAttributeTo(Document docBPEL, Node node, Element action) {
       if (node.hasAttributes()) {
              if (((Element) node).hasAttribute(action
                     .getAttribute("attributeName"))) {
              System.out.println(" SetAttributeTo(" + node.getNodeName()+ ")");

                     if (action.getAttributeNode("value").getNodeValue().equals(
                            "null")) {
                     // Dieser Attribut wird gelöscht
                     // Die Informationen von dieser Attribut wird für alle andere
                         Aktivitäten verborgen
                            String str = ((Element) node).getAttribute(action
                                          .getAttribute("attributeName"));
                            ((Element) node).removeAttribute(action
                                          .getAttribute("attributeName"));
                            IsAttributeBReferenceToAttribueA(docBPEL
                                          .getDocumentElement(), str);
                     }
                     else {
                        // Der Wert wird überschrieben.
                            ((Element) node).setAttribute(action
                                          .getAttribute("attributeName"), action
                                          .getAttribute("value"));
                            }
              }
       }
}
//Die Inhalte aus dem gelöscht Attribut wird in andere Aktivitäten gesucht, die
    entsprechende Informationen sollen verborgen werden.
public void IsAttributeBReferenceToAttribueA(Node node2, String str) {
       if (node2.hasAttributes()) {
              NamedNodeMap n2 = node2.getAttributes();
              // durchsuchen die Attribute in dieser Knoten
              for (int y = 0; y < n2.getLength(); y++) {
              if ((n2.item(y)).getNodeValue().equals(str)) {
                     (n2.item(y)).setNodeValue("##opaque");
                     }
              }
       }
              if (node2.hasChildNodes()) {
              NodeList nl = node2.getChildNodes();
              for (int x = 0; x < nl.getLength(); x++) {
              // durchsuchen alle Kinderaktivitäten von dieser Knoten
              IsAttributeBReferenceToAttribueA((nl.item(x)), str);
              }
       }
}
```

3.3.2. Undurchsichtige Aktivität

Diese Operation »actionOpaque« bleibt unverändert und es wird keine neue Funktionen erweitert. Eine detaillierte Erklärung mit Testbeispiel wird in Arbeit von [Str09, Seit 43 bis 45] gefunden.

3.3.3. Aktivitätseliminierung

In dieser Arbeit wird einen generelle Eliminierungsverfahren von Aktivitätselementen entworfen, es bietet einen vereinfachte Methode für den Aktivitätseliminierung »actionOmit« in BPEL-Prozess an. Listing 3.7 zeigt den deklarativ Verfahren von Operation »actionOmit«, die in der praktisch Arbeit vervollständigt und erweitert wird.

In den BPEL-Prozess wird prinzipielle nur die Aktivitätselement ausgewählt, um die Modifizierung und Löschung von Aktivitäten zu realisieren. Es können die einfache Aktivitäten und die strukturierte Aktivitäten (inklusive »scope«). Bevor die Eliminierung durchgeführt wird, die ausgewählte Aktivitäten wird auf den signierte Marke »Preserve« rekursive durchgesucht.

Es wird in der Arbeit so vorgeschrieben, dass in jede Aktivitäten darin einen Marke »Preserve« gibt, könnt es nicht gelöscht werden. Die Methode »IsStructuredActivityCompletelyOmittable()« implementiert dieser Durchsuchung. Es wird in der Arbeit keine auf konkrete Aktivitätstype spezifizierte Methode zum Eliminierung entwickelt. Für jede Aktivitätstype wird es einen generelle Lösung spezifiziert. Es entscheidet entweder Eliminierung oder Erhaltung wegen »Preserve«, die meisten Schwierigkeiten bei spezielle Löschung wird nach die Erstellung eines effizient Rules-Dokumentes verschoben.

Nach die Löschung von Aktivität müssen die Prozesssemantik gewährleistet, deswegen wird es bei mehrere mögliche Situationen betrachtet. Die Aktivitäten in BPEL-Prozess können sequenziell verknüpft oder mit gerichtete Pfeile zusammen verbindet. Eine Aktivität in einen Sequenz können unmittelbar gelöscht wird, aber in einen Flow wird die Aktivität mit andere Aktivitäten durch Verknüpfung sequenzielle oder nebenläufig verbindet. Es wird nach den Löschung die vorherig Semantik nicht beschädigt.

Listing 3.7 Algorithmus für die Aktivitätseliminierung

```
function omit (targetActivity)

     if (isActivity(targetActivity))
     {
        omitActivity(targetActivity)
     }
     else
     if (isStructuredActivity(targetActvity))
     {
        //Die strukturierte Aktivität wird auf die markierte Anmerkung >Preserve<
            durchgesucht, wenn es keine gibt, dann wird ganz gelöscht
        if (IsStructuredActivityCompletelyOmittable(targetActivity))
        {
          omitStrukturedActivty(targetAcitivty)
        }
     }

-------------------------------------------------------------------------------

function omitActivity (targetActivity)

  variable : Predecessors //list of nodes
             Successors   //list of nodes

     if (the targetActivity has not preserve mark)
     {
       //Die Aktivität hat nur Eingangspfeile oder nur Ausgangspfeile, wird einfach gelöscht
        if (the targetActivity has only incoming links or only outgoing links)
        {
           remove(targetAcitivity)
        }
        else if (the targetActivity has incoming links and outgoing links)
        {
            Predecessor <--- getAllPredecessor(targetActivity)
            Successors <--- gteAllSuccessor(targetActivity)
           //Die Verknüpfung wird erzeugt, bevor die Zwischenaktivität gelöscht wird
            forEach (Predecessors)
            {
               forEach (Successors)
               {
                  addLinkBetweenPredecessorAndSuccessor();
               }
             }
            remove(targetActivity)
         }
         else
         {
          //Die Aktivität besitzt keine Verknüpfung oder liegt in Sequenz, wird einfach
              gelöscht
           remove(targetActivity)
         }
     }
```

4. Realisierung der View-Operationen

In diesem Kapitel wird die in dieser Studienarbeit fertiggestellte Kernimplementierung von View-Operationen ins Details erklärt. Die View-Operationen sind die drei Grundoperationen in Abschnitt 3.3 und die erweiterte Operationen in diesem Kapitel. Die Implementierung von View-Operationen und die entsprechende Konzeptionen sind die Kernaufgaben in den praktisch Arbeit.

Die Navigation und Auswahlspezifikation von Verarbeitungsobjekt auf BPEL-Prozess wird durch Anwendung von Xpath-Sprache realisiert, es wird in Abschnitt 4.1 die wichtige Prinzip und Algorithmus von Xpath-Anwendung detailliert vorgestellt.

In Abschnitt 4.2 werden die Vervollständigung und Erweiterung von den View-Operationen erläutert, die in der Basisarbeit von [Str09] für den letzte Verschönerung spezifiziert wurden.

Die entwickelte Funktionen schaffen die zielbasierte Sichten auf BPEL-Prozess, solche Funktionen in Abschnitt 4.3 werden als den vordefinierte Kombinationsmustern von den View-Operationen spezifiziert.

In Abschnitte 4.4 wird die Dokumentation für Installation und Anwendung von Jar-Paket vorgestellt.

4.1. Xpath-Anwendung

In der Arbeit wird die <targets> Elemente in Rules-Dokument als Xpath-Ausdrücke durch Algorithmus umgewandt. Die Navigation von Zielobjekt in BPEL-Prozess werden durch die Programmierschnittstelle java.xml.xpath [Suna] unabhängig vom Objektmodell realisiert. In dieser Abschnitt wird die Prinzip von Generierung von Xpath-Ausdrücke vorgestellt. Eine erweiterte Generierung von Ausdrücke für den logische Kombination von Auswahlkriterien wird durch einen rekursive Algorithmus entwickelt.

4.1.1. Ausdrucksgenerierung

Bei Aktivitätsauswahl durch entsprechende »@tag« kommentierte Dokumentation wird eine Ausdrucksformel wie Listing 4.2 generiert. Die »tagName« in den Ausdruck wird von die eingegebene Zeichenkette des Attribute »tagName« in <tag> Element beispielsweise

wie Listing 4.1 ersetzt. Die Xpath-API compiliert den Xpath-Ausdruck und wertet es aus, der Programm sucht den W3C Dokument [Sunb] in DOM-Baum durch, die Aktivitäten werden lokalisiert und ausgewählt, die dieser durch entsprechende »@tag« kommentierte Dokumentation beinhalten. In BPEl-Prozess wird die angehörige Dokumentation von »@tag« unter den Aktivität angehängt, deswegen werden die Xpath-Prozessor nur die Elemente in erste Unterschicht zusammenpassen werden.

Listing 4.1 <tag> in <targets> Element

```
<targets>
      <tag tagName="gold"/>
</targets>
```

Listing 4.2 <tag> Xpath-Ausdruck

```
Expression = "//*[(contains(./*/text(), '@tag gold'))]"
```

Bei Elementauswahl durch entsprechende Wertzuweisung eines Attributes wird einen <attribute> Element in <targets> beispielsweise aus Listing 4.3 als eine Ausdruck wie Listing 4.4 umgewandt. Die »attributeName« und »value« werden von entsprechenden Zeichenkette in <attribut> Element ersetzt. In Xpath bietet es so eine Methode an, dass die Element mit spezifizierte Wert von entsprechende Attribute navigiert werden können. Die Xpath-Prozessor wählt alle angepasste Element in BPEL-Prozess aus, die diese Bedingung erfüllt.

Listing 4.3 <attribute> in <targets> Element

```
<targets>
      <attribute attributeName="name" value="CheckFlightReservation" />
</targets>
```

Listing 4.4 <attribute> Xpath-Ausdruck

```
Expression = "//*[(@name = 'CheckFlightReservation')]"
```

Bei Auswahl von Aktivitätstype von BPEL-Prozess wird eine Ausdruck wie Listing 4.6 erzeugt. In Xpath wird die Auswahl durch Elementname schon durch »name()« spezifiziert. Die »typeName« in den Ausdruck wird durch die eingegebene Typenname von BPEL-Prozess beispielsweise wie Listing 4.5 ersetzt. Die alle angegebene Aktivitätstype werden ausgewählt.

Listing 4.5 <type> in <targets> Element

```
<targets>
      <type typeName="invoke" />
</targets>
```

Listing 4.6 <type> Xpath-Ausdruck

```
Expression = "//*[(name() = 'invoke')]"
```

4.1.2. Logische Kombination von Auswahlkriterien

Die Xpath unterstützt auch die logische Operatoren »and«, »or« und »not« in den Prädikaten, um die Element in einen bestimmte Bereich zu beschränken. In dieser Arbeit wird ein Algorithmus entwickelt, um die Ausdrucksgenerierung aus den logische kombinierte Auswahlkriterien zu unterstützen.

In das rekursiv Algorithmus sowie A.3.1 in Anhang wird die logische Kombinationen in <targets> Element von Rules-Dokument als einen gleichwertige Xpath-Ausdruck umgewandt. Die mehrfach verschachtele durch logische Operatoren kombinierte Auswahlkriterien werden durch mehrfach verschachtele Klammern in das Prädikat realisiert.

Listing 4.7 zeigt beispielsweise, wie die beide Auswahlkriterien durch »and« in das Prädikat von Xpath-Ausdruck kombiniert.

Listing 4.7 <and> Kombination von Prädikaten

```
Expression = "//*[((contains(./*/text(), tagName)) and (@attributeName = value))]"
```

Listing 4.8 zeigt beispielsweise, wie die beide Auswahlkriterien durch »or« in das Prädikat von Xpath-Ausdruck kombiniert.

Listing 4.8 <or> Kombination von Prädikaten

```
Expression = "//*[((contains(./*/text(), tagName)) or (@attributeName = value))]"
```

Listing 4.9 zeigt beispielsweise, wie die beide Auswahlkriterien durch »and« und »not« in das Prädikat von Xpath-Ausdruck kombiniert.

Listing 4.9 <not> Kombination von Prädikaten

```
Expression = "//*[(not ((name() = typeName) and (@attributeName = value)))]"
```

Listing 4.10 zeigt beispielsweise die generierte Xpath-Ausdrücke von Rules-Dokument wie Listing 3.1 nach die Implementierung von Algorithmus A.3.1 in Anhang.

Listing 4.10 Die Xpath-Ausdrücke für die Lokalisierung von den Zielobjekte

```
//Der Xpath-Ausdruck für die erste Aktion <addPreserve>
Expression = "//*[(@name='Confirmation' or @name='CheckFlightReservation' or
    @name='CheckHotelReservation' or @name='While' or @name='HiddenSequence' or
    @name='CheckCarReservation' or @name='EvaluateReservationResult' or @name='Reply')]"


//Der Xpath-Ausdruck für die zweite Aktion <actionOmit>
Expression = "//*[(//*)]"
```

4.1.3. Performance und Zeitmessung

Die Performance ist einen kritischen Faktor von den Softwaresystemen, es wird in der Arbeit eine Messung von den wichtige Faktoren für Xpath-Anwendungen und die Durchsuchung von Zielobjekte in den DOM-Baum analysiert. Die XML-Verarbeitung mit DOM-Baum besitzen eine gute Modifizierbarkeit von XML-Dokument und eine hoch Speicherbedarf, wenn die Basisgeschäftsprozesse bereits ziemlich groß sind, werden die gebrauchte Ressourcen die wichtigen Elementen. Die Abbildungen 4.1 und 4.2 zeigen gegenseitig den wichtigen Faktoren sowie Speicherbedarf, Leistungsverbrauch, Threads und generierte Klassenanzahl.

Die beide Implementierungsmethode basieren auf DOM-Prinzip, aber benutzten unterschiedliche Verfahren bei Zielnavigation. Deswegen war es bei dem Speicherbedarf keine deutliche Differenz dazwischen erscheint, die Speicherbedarf wird nur von den Größe von den eingegebene Prozessdaten abhängig. Bei andere kritische Faktoren sieht man auch einen deutliche Bewertung dazwischen nicht. Die Ausführungszeit wird dann wichtiger, wenn es durch die oben genannte Ressourcen keine deutliche Entscheidung gemacht wird.

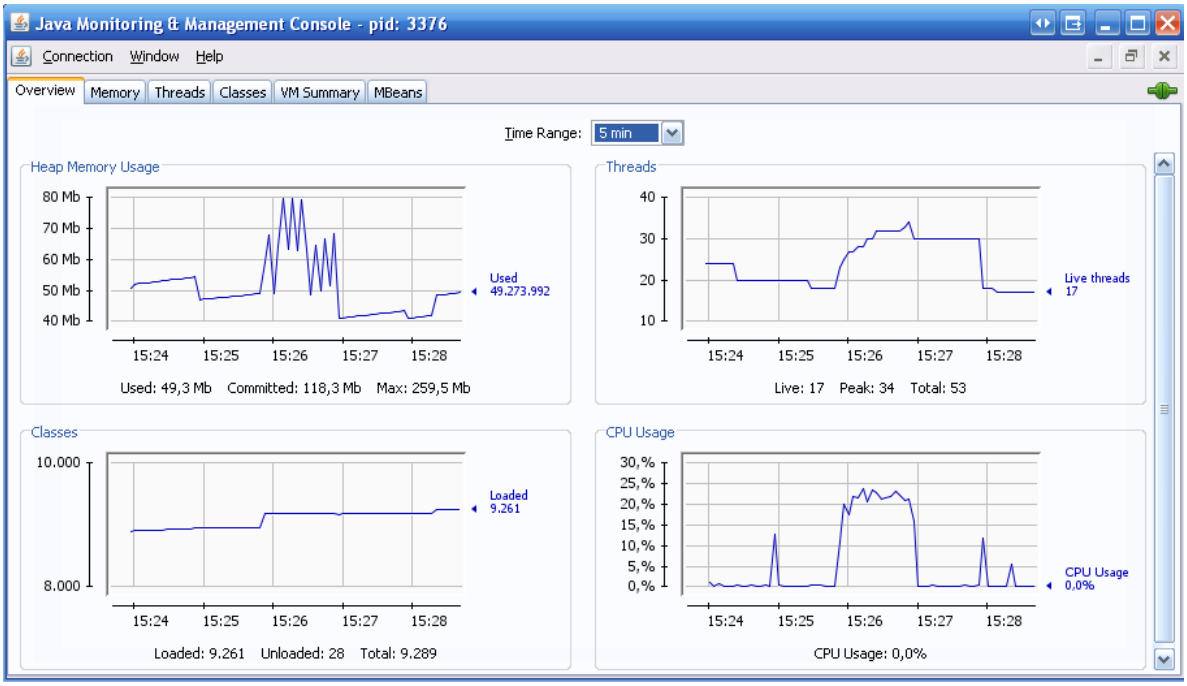


Abbildung 4.1.: Performance von der Xpath-Anwendung

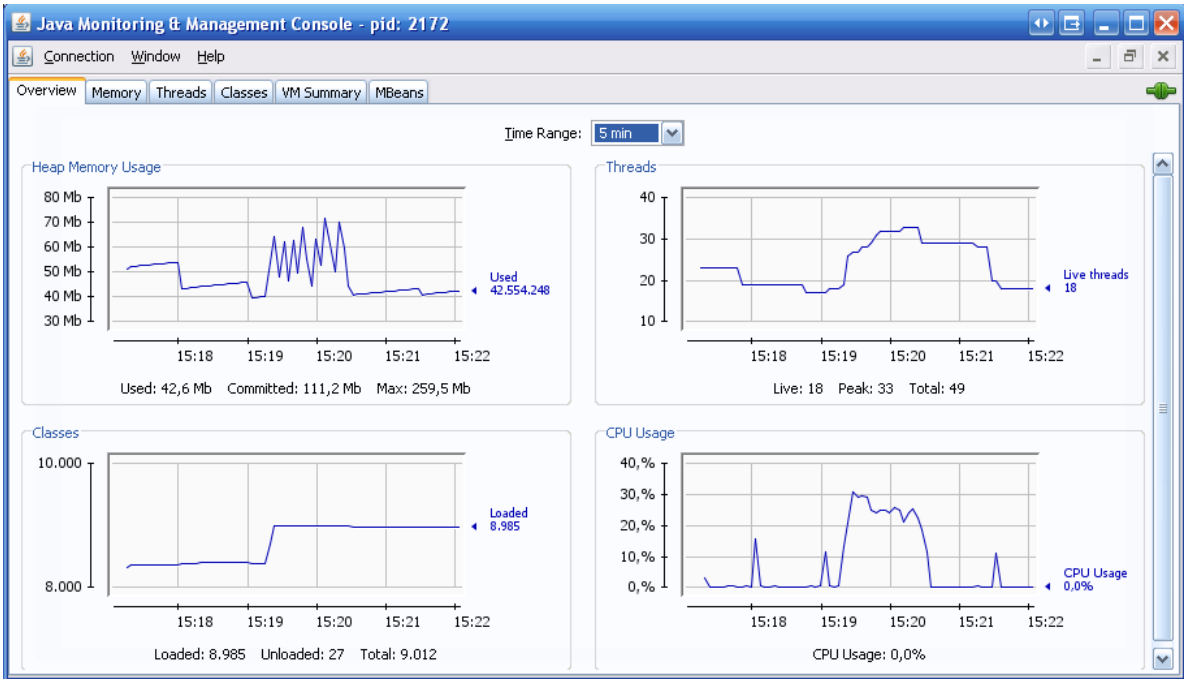


Abbildung 4.2.: Performance von dem Durchsuchen in dem DOM-Baum

Die Tabelle 4.1 stellt die Zeitmessung von den zwei verschiedene Implementierungsmethode für die Navigation von Zielobjekte durch mehrfache Experimenten dar. Es wird bei dem Zeitmessung mehrere Testfälle sowie Prozessdaten mehrfach in Schleife durchgeführt. Die Verbrauchzeit von Xpath-Anwendungen sind offensichtlich mehr als die Durchsuchen von Zielobjekte in den DOM-Baum. Durch die weitere Zeitmessungen für den Ausführungstest von Xpath-Anwendung werden die Zeitaufwände von den Methoden untersucht, es zeigt in den Testwerkzeug von Eclipse so, dass liegt die Zeitaufwand von den Evaluierung von Xpath-Ausdrücke durchschnittlich ca. 15 bis 20 Prozent, aber die Ausführungszeit von der Durchsuchung in den DOM-Baum liegt durchschnittlich nur bis ca. 5 Prozent. D.h, es wird wirklich die mehrere Zeiten bei Auswertung von Xpath-Ausdrücke als die Durchsuchung in den DOM-Baum benötigt. Eine verbesserte Lösung für die Xpath-Anwendung wäre eine potenzielle Optimierung von den Xpath-Ausdrücke.

	5 Fälle	50 Fälle	500 Fälle	1000 Fälle	2000 Fälle
Xpath-Anwendung	2524	5058	32186	57773	108877
DOM-Baum Durchsuchen	1582	5187	28030	53657	101425

Tabelle 4.1.: Zeitmessung und Vergleich in Millisekunden

4.2. Erweiterte View-Operationen

In dieser Abschnitt werden die vervollständigte und erweiterte Operationen in der Arbeit erklärt. Zusammen mit vorn genannte Grundoperationen sollen diese Operationen einen verbesserte Anwendungsfähigkeit für »Process View« anbieten. Die View-Operationen in der Arbeit sind die drei Grundoperationen in Abschnitt 3.3 und die erweiterte Operationen in dieser Abschnitt. Die Funktionen in Abschnitt 4.3 werden als den gültige Kombinationen von den View-Operationen gesehen.

4.2.1. Automatisches Hinzufügen von Tags

Es wird in der Arbeit von [Str09] einen »Tag Plugin« für den Kontextmenü von »BPEL Designer« entwickelt, mit den die Aktivitäten in BPEL-Prozess durch »Preserve« oder die neue Informationen sowie »@tag« kommentiert werden können. Weil es in dieser Arbeit kein graphisch Verarbeitungswerkzeug von BPEL-Prozess mit »BPEL Designer« entwickelt wird, es wird nur durch Kommandos in den Quelltext die Methoden aus Jar-Paket aufgerufen. Die Rules-Dokument sollen die alle mögliche Verarbeitung allein unterstützen, deswegen werden die Hinzufügung von neue Tags für den Aktivitäten einen sehr hilfreiche View-Operation.

Listing 4.11 zeigt beispielsweise einen Rules-Dokument, die die Aktivität »CheckFlightReservation« aus den Beispielprozess A.2.1 in Anhang durch neue Informationen kommentiert. Weil die Dokumentationen von den Aktivitäten bei Reinigungsoperation aus Abschnitt 4.2.3 beseitigt werden, es wird bei dieser Beispiel keine Reinigung durchgeführt. Listing 4.12 stellt die entsprechende resultierende Aktivität dar.

Listing 4.11 Rules-Dokument für die Hinzufügung von den Tags

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:rules xmlns:tns="http://www.eclipse.org/bpel/views/rules"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.eclipse.org/bpel/views/rules Rules.xsd"
        name="addHinweise">

        <parameter>
                <aggregateOpaque value="true" />
                <cleaning value="false" />
        </parameter>

        <rule apply="true" name="addTag">
                <actions>
                        <addTag value="department" />
                        <addTag value="stuttgart" />
                        <addPreserve />
                </actions>
                <targets>
                        <attribute attributeName="name" value="CheckFlightReservation" />
                </targets>
        </rule>
</tns:rules>
```

Listing 4.12 Die durch neue Tags annotierte Aktivität

```
<bpel:invoke inputVariable="FlightReservationInput" name="CheckFlightReservation"
    operation="reserve" outputVariable="FlightReservationOutput"
    partnerLink="FlightReservation" portType="ns4:FlightReservation"
    wpc:displayName="CheckFlightReservation" wpc:id="19">
      <bpel:targets>
        <bpel:target linkName="Link6"/>
      </bpel:targets>
      <bpel:sources>
        <bpel:source linkName="Link9"/>
      </bpel:sources>
      <bpel:documentation>@tag department @tag stuttgart @mark preserve</bpel:documentation>
</bpel:invoke>
```

4.2.2. Aktivitätsgruppierung

Es wird nach mehrere Durchführungen von <actionOpaque> mehrere undurchsichtige Aktivitäten erzeugt, eine sinnvolle Gruppierung von solche undurchsichtige Aktivitäten bietet einen besser anschauliche Darstellung von resultierend BPEL-Prozess. Die Erhaltung von den gültige Semantik von resultierend Ergebnisprozess wird einen kritische Punkt, es sollen keine Schleife oder unnötige Strukturen generiert werden. In der Arbeit von [Stro9, Seit 67 bis 68] wird die Definition von Gruppierung von undurchsichtige Aktivitäten erklärt, in dieser Arbeit wird diese Operation wie <aggragateOpaque> weiter aufgebaut und implementiert.

Der rekursiver Algorithmus von Aktivitätsgruppierung sucht die Aktivitäten in den DOM-Baum von BPEL-Prozess durch. Bei dem Fund einer erste undurchsichtige Aktivität werden die alle verknüpfte Nachfolgern von diese Aktivität untersucht, wenn es auch eine undurchsichtige Aktivität ist, wird es gelöscht, die entsprechende Verknüpfungen werden durch Erzeugung von Pfeile erhaltet. Die neue resultierende Nachfolgern von die erste undurchsichtige Aktivität werden nochmal untersucht, wenn es keine undurchsichtige Aktivität gibt, wird der Algorithmus auf den nächste undurchsichtige Aktivität gesucht. In dieser Algorithmus wird nur die nach einander liegende undurchsichtige Aktivitäten schrittweiser gruppiert. Das vermeidet die Erzeugung von Schleife durch ein vereinfacht Prinzip.

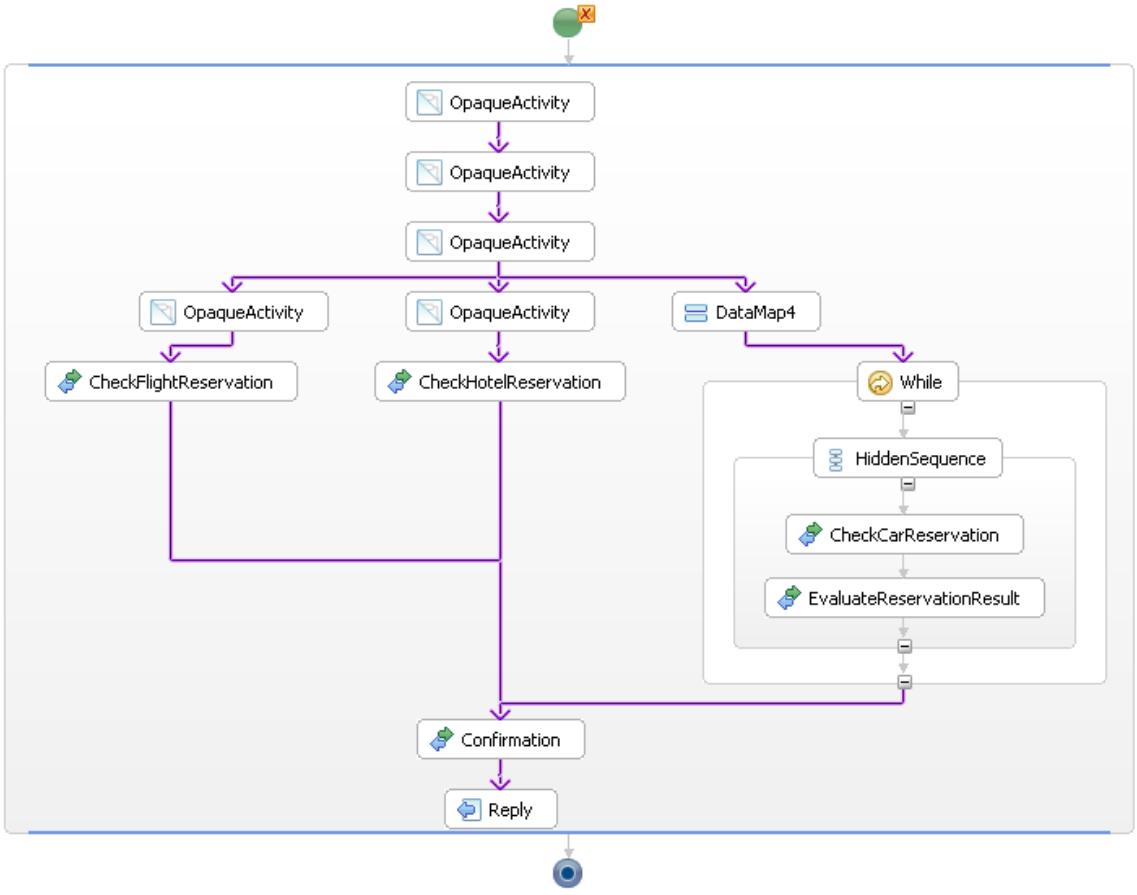


Abbildung 4.3.: Der BPEL-Prozess nach mehrfachem <actionOpaque>

Die Abbildung 4.3 stellt einen Testbeispiel von Aktivitätsgruppierung dar, die resultierend BPEL-Prozess nach den <actionOpaque> spezifiziert noch die Struktur von Basisprozess, es bietet einen niedrige Beschützen von geschäftliche Informationen an. Nach die Aktivitätsgruppierung wird das Ergebnis in die Abbildung 4.4 illustriert. Die überflüssige Pfeile

werden wegen die einseitige Löschung von den Verbindungspunkte zwischen den undurchsichtige Aktivitäten und normale Aktivitäten nach dem Aktivitätsgruppierung resultiert, durch die Operation <cleaning> werden solche überflüssige Elementen beseitigt. Eine hervorragende Darstellung mit Reinigung von unnötigen Links wird in die Abbildung 4.5 gezeigt.

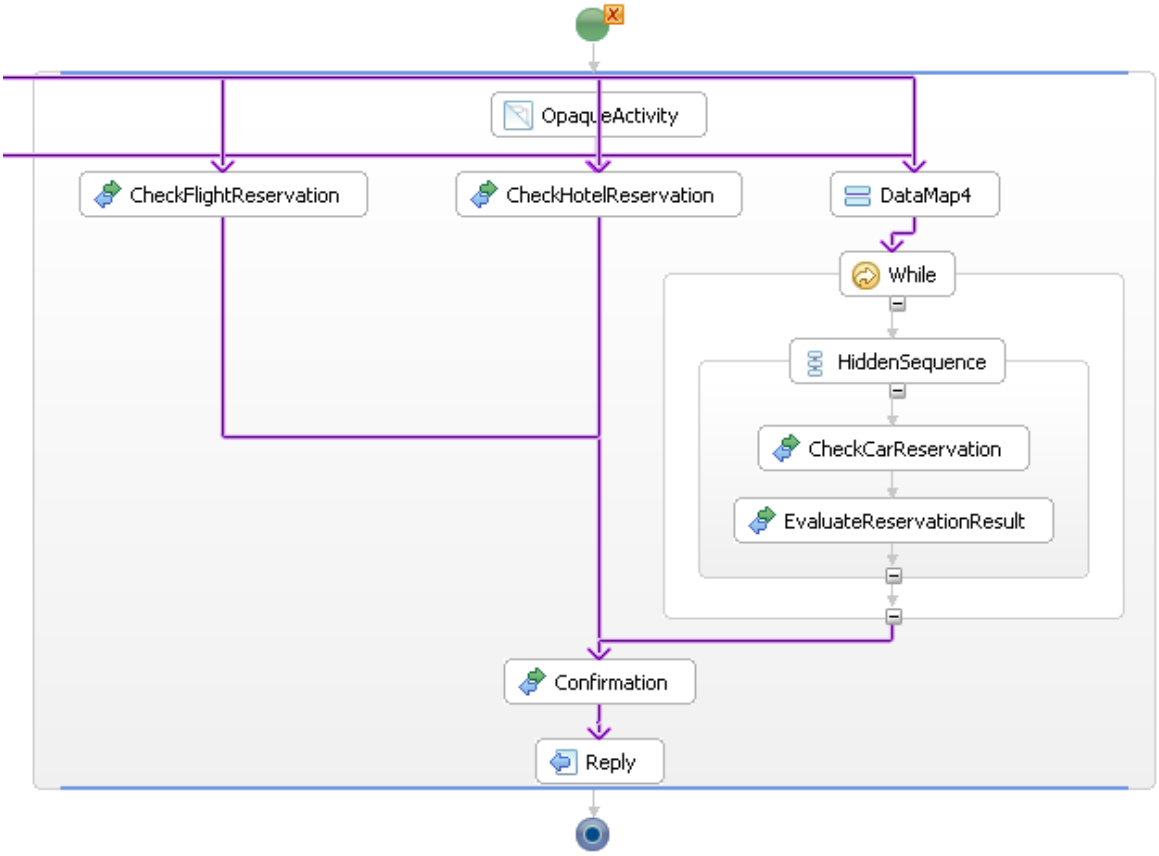


Abbildung 4.4.: Der BPEL-Prozess ohne Reinigung nach Gruppierung von undurchsichtigen Aktivitäten

4.2.3. Prozessreinigung

Die in Arbeit von [Stro9, Seit 62 bis 63] spezifizierte Operation sowie <cleaning> für Prozessreinigung strebt einen gut-formatierter BPEL-Prozess nach die Verarbeitungen an. Die unnötige Elemente von resultierend Prozess werden gelöscht, um die empfindliche Informationen zu beschützen. Eine fortschrittliche Reinigung von Prozessstruktur und Prozesssemantik wird nach die Standard von WS-BPEL 2.0 [AAA+] durchgeführt. Ein Problem von den

überflüssige Verknüpfungen wird ausgelöst, die in der Arbeit von [Stro9, Seit 63] aufgestellt wird.

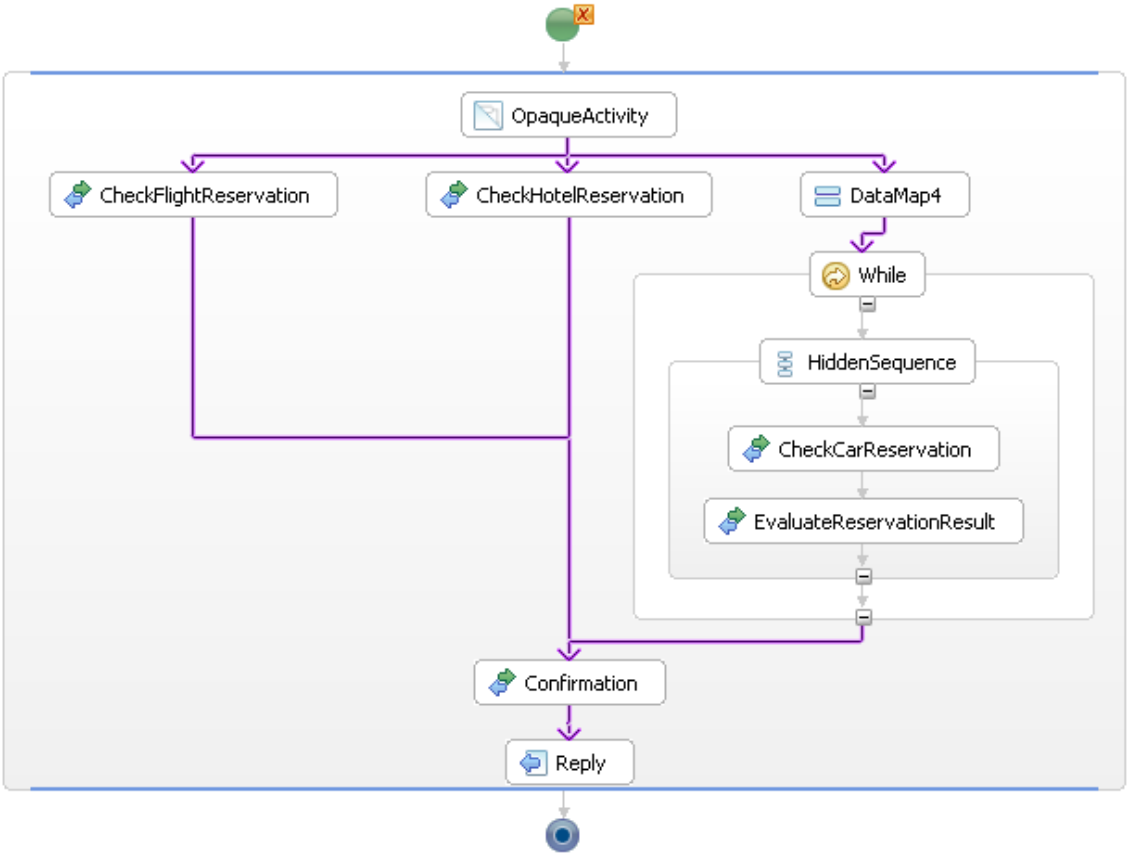


Abbildung 4.5.: Der BPEL-Prozess mit Reinigung nach Gruppierung von undurchsichtigen Aktivitäten

Zusammen mit Aktivitätsgruppierung wird die Prozessreinigung für den Verschönerung von Prozess in <parameter> Element von Rules-Dokument angewendet. Listing 4.13 zeigt ein Beispiel für den Anwendung von <prarameter> Element.

Listing 4.13 Anwendungsbeispiel von <parameter> Element

```
<parameter>
              <aggregateOpaque value="true" />
              <cleaning value="true" />
</parameter>
```

4.3. View-Funktionen

Die vervollständigte View-Operationen schaffen eine Vielfalt von Prozessverarbeitung durch beliebige Kombination. Die Erstellung eines Rules-Dokumentes werden außer die Wissen von den Geschäftsprozess im Unternehmen einen tiefen Verständnis von View-Operationen und die technische Erkenntnissen von WS-BPEL erfordert. Für den Fachleute, die nur die geschäftliche Erkenntnis von Geschäftsprozess besitzen, werden die vordefinierte Funktionen für den »Process View« sehr hilfreich. Es wird in dieser Arbeit zwei vorprogrammierte Funktionen für »Process View« entwickelt. Die Entwicklung von neue View-Funktion ist durch Vorprogrammierung von einen sinnvolle Kombination möglich.

4.3.1. Fokussieren

Ein Geschäftsprozess können sehr komplex sein und Hunderte von Aktivitäten beinhalten. Es ist immer möglich, dass die Prozessexperte sich nur die Teile von ganz Geschäftsprozess oder einen lokale Semantik von eine bestimmte Aktivität interessieren. Eine Fokussieren von einen bestimmte Aktivität oder einen bestimmte Prozessgebiete wird nützlich, um einen partielle und konzentrierte Sicht auf BPEL-Prozess zu bekommen. Die Prozessexperte müssen nur die Name der Aktivität und das numerisch Sichtumfang eingeben, es wird dann einen konzentriert Prozess durch Abschnitt von nicht nützliche Aktivitäten generiert. Dadurch wird ein effizient Prozessstück nach den Anforderung erstellt, die andere geschäftliche Details von ganz Geschäftsprozess, die für den Prozessanalyse nicht aufgerufen werden, wird gleichzeitig geschützt.

Es wird in der Arbeit eine Klasse »ViewsFunction.Transaction« entwickelt, die mehrere Methoden sowie Funktionen für »Process View« beinhaltet. In die Klasse wird die Fokussieren als Methode »SetAFocus()« mit Parametern implementiert. Nach Eingabe von Aktivitätsname und Nummern, die die Schrittanzahl für den rekursive Suchen von Vater- und Kinderknoten in den Graphen deklariert, wird es einen vorprogrammierte Rules-Dokument generiert. Die Ergebnis und erzeugt Rules-Dokument von die Ausführung von Codebeispiel aus Listing 4.14 werden in Anhang A.2.2 gezeigt.

Listing 4.14 Anwendungsbeispiel von der Methode »SetAFocus«

```
Transaction op = new Transaction();
//setAFocus(Document, String, Integer, Integer, Boolean) oder
//setAFocus(String, String , Integer, Integer, Boolean)
op.setAFocus("TravelBooking.bpel", "CheckCreditCard", 1,1,false);
```

Die Methode »SetAFoucs« sucht erst die alle Zielobjekte aus, die verbleiben sollen. Durch Kombination von zwei View-Operationen sowie <addPreserve> und <actionOmit> werden die alle Zielobjekte erst durch »Preserve« kommentiert, dann werden die alle Aktivitätselemente außer die schon kommentierte Aktivitäten gelöscht. Es wird unbedingt so aufgepasst,

dass es bei Anwendung von Fokussieren nur in den BPEL-Prozess funktioniert, deren Verknüpfungen zwischen Aktivitäten in BPEL-Prozess durch Pfeile realisiert werden. Weil es bei Durchsuchen von Vater- und Kinderaktivitäten wegen die Komplexität schwer realisierbar ist, die nicht durch Pfeile verknüpft werden.

4.3.2. Teilprozesseliminierung

In WS-BPEL 2.0 [AAA$^+$] wird einen Extension für Subprozess [KKL$^+$05] entwickelt, um einen höhere Wiederverwendbarkeit und Modularität in Prozessmodellierung zu schaffen. Die genaue Spezifikation und Anwendungen von BPEL-SPE für »Process View« sind nicht in den Aufgabegebiete der Arbeit. Einer vordefinierter Subprozess können wegen die besten Erfolgsrezept als einen Teilprozess in den Basisprozess eingesetzt werden. In dieser Abschnitt wird die Eliminierung von Teilprozess vorgestellt. Einer vorrätiger Teilprozess bzw. Prozessfragment in Basisprozess wird einfach gelöscht, um die wichtige Know-How in den Geschäftsprozess zu schützen.

In die Klasse »ViewsFunction.Transaction« wird eine Methode »removeFragment()« implementiert, die die Teilprozesseliminierung schaffen können. Mit diese Methode wird das Prozessfragment aus den Basisprozess weggelassen, um dieser Prozessfragment zu verbergen. Die Ergebnis und erzeugt Rules-Dokument aus die Ausführen von dem Codebeispiel in Listing 4.15 werden in Anhang A.2.3 gezeigt. In »businessActivities.bpel« kommen die zwei vordefinierte geschäftliche Teilprozesses bzw. »hotel« und »flight« vor, die gelöscht werden müssen.

Listing 4.15 Anwendungsbeispiel von der Methode »removeFragment«

```
Transaction ts = new Transaction();
//removeFragment(Document, Document, Boolean) oder
//removeFragment(String, String, Boolean)
ts.removeFragment("Travelbooking.bpel", "businessActivities.bpel",false);
```

Die Methode »removeFragment« lest die Basisprozess und die Prozessfragment ein. Die Aktivitäten in Basisprozess sollen gelöscht werden, die in den Prozessfragment gefunden werden. Durch Untersuchung auf das Prozessfragment werden die alle Aktivitäten erkannt und als die Zielobjekte speichert. Nach die Generierung von Rules-Dokument aus <actionOmit> und die alle Zielobjekte wird die Basisprozess verarbeitet, die entsprechende Aktivitäten in Basisprozess werden dann gelöscht.

4.3.3. Transformation

In die Klasse »ViewFuncction.Transaction« bietet es eine primitive Operation »transform()« an, die als einen zentraler Arbeitsschritt für den Generierung von »Process View« in Abschnitt 3.1 prinzipielle betrachtet wird. In Listing 4.16 und 4.17 werden die zwei Aufrufmethode von »transform()« für den Eingabe und Ausgabe spezifiziert.

Listing 4.16 W3C Document als Eingabe und Ausgabe

```
public Document transform(Document process, Document rules, boolean logging)
                            throws DocumentException, IOException {

                        return docViewBPEL;
}
```

Listing 4.17 Dateinamen als Eingabe und Ausgabe

```
public String transform(String process, String rules, boolean logging)
                            throws IOException, DocumentException {

                        return "view_" + process;
}
```

Die Abbildung 4.6 stellt das Verbindungsschema zwischen die Klasse »transaction« und die weitere Klassen dar. Für die View-Funktionen werden die Methoden aus die drei Klassen »RuleImport«, »Operation« und »ProzessLesenAusgabe« aufgerufen. Die Klasse »RuleImport« ruft die Methoden aus die Klassen »Operation« und »ProzessLesenAusgabe« weiter auf.

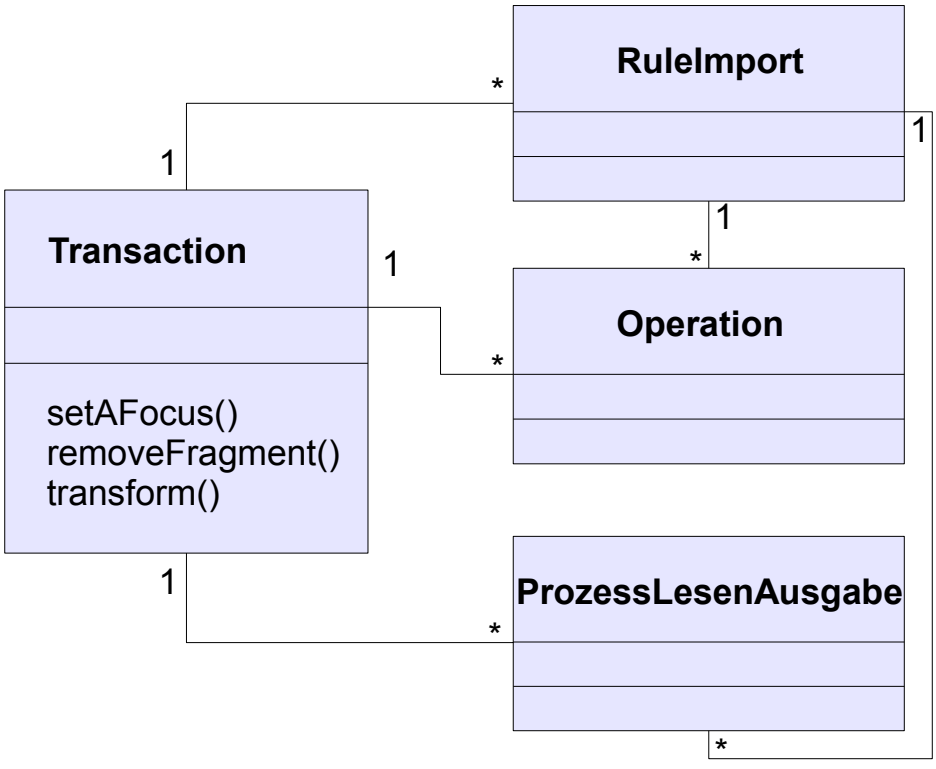


Abbildung 4.6.: Klassendiagramm von »Transaction«

In der Arbeit werden für die drei Methode in Abschnitt 4.3 ein boolesch Schalter »logging« hinzugefügt, es sollen eine zusätzliche Informationen für den Weiterverarbeitung von resultierend BPEL-Prozess anbieten. Durch »logging« werden die Aktivitäten und entsprechende ausgeführte Aktionen protokolliert. Nach den Prozessverarbeitung wird einen XML-basierter Protokolltext erzeugt, um die weitere korrelierte Anwendungen mit BPEL-Prozess einfach anpassen zu können.

Listing 4.18 Die protokollierten Nachrichten von dem Anwendungsbeispiel in Abschnitt 4.3.2

```
<log4j:content>
       <log4j:event logger="ViewProcess.Test" timestamp="1268918514123"
              level="INFO" thread="main">
              <log4j:message>
                   <![CDATA[this is a log message.]]>
              </log4j:message>
       </log4j:event>
       <log4j:event logger="ViewProcess.Test" timestamp="1268918514153"
              level="INFO" thread="main">
              <log4j:message>
                   <![CDATA[<actionOmit>DataMap3</actionOmit>]]>
              </log4j:message>
       </log4j:event>
       <log4j:event logger="ViewProcess.Test" timestamp="1268918514163"
              level="INFO" thread="main">
              <log4j:message>
                   <![CDATA[<actionOmit>CheckHotelReservation</actionOmit>]]>
              </log4j:message>
       </log4j:event>
       <log4j:event logger="ViewProcess.Test" timestamp="1268918514183"
              level="INFO" thread="main">
              <log4j:message>
                   <![CDATA[<actionOmit>DataMap2</actionOmit>]]>
              </log4j:message>
       </log4j:event>
       <log4j:event logger="ViewProcess.Test" timestamp="1268918514213"
              level="INFO" thread="main">
              <log4j:message>
                   <![CDATA[<actionOmit>CheckFlightReservation</actionOmit>]]>
              </log4j:message>
       </log4j:event>
</log4j:content>
```

4.4. Anwendungsszenario

Es wird in dieser Abschnitt die Installationsanleitung von Jar-Paket sowie »view4bpel.jar« und den Hinweis von der Verwendung mit Codebeispiele erklärt.

Das Jar-Paket »view4bpel.jar« können einfach ins Repository von Anwendungsserver importiert, um die View-Applikation in Webanwendungsserver zu integrieren und verwenden.

Für den weitere Anwendungsentwicklung in Eclipse-Plattform können »view4bpel.jar« in den referenzierte Bibliothek konfiguriert werden. Für den Ausführung von »view4bpel.jar« wird die weitere Jar-Pakets »dom4j.jar« und »log4j.jar« benötigt. Die Funktion »prettyPrint« von XMl-Dokument in »dom4j« wird aufgerufen, um das resultierend BPEL-Prozess in einen schönen XML-Format zu generieren. »log4j« wird für den Protokollierung von Prozessverarbeitung angewendet, eine entsprechende Konfigurationsdatei »log4j.properties« sollen dabei helfen, das erzeugt Protokoll zu spezifizieren.

Das XML-basiert Rules-Dokument ist das Kerndokument für den Prozessverarbeitung, ein effizient Rules-Dokument schafft einen anwendungsspezifische Sicht von Basisprozess. Die Tabelle 4.2 zeigt die alle unterstützte View-Operationen für den Erstellung von Rules-Dokument, solche View-Operationen wurden in »view4bpel.jar« als primitive Operationen betrachtet und unterstützt. Jede Kombination von den View-Operationen in einen sinnvolle Sequenz wird unter »view4bpel.jar« erfolgreich implementiert.

View-Operation	Kurzerklärung
<actionOmit>	löscht die Aktivität
<actionOpaque>	verbergt die Aktivität
<actionSetAttributeTo>	konfiguriert das Attribut
<addPreserve>	fügt »Preserve« Hinweis hinzu
<addTag>	fügt «@tag« Information hinzu
<aggregateOpaque>	Gruppierung von undurchsichtigen Aktivitäten
<cleaning>	Prozessreinigung

Tabelle 4.2.: Kurzfassung von allen View-Operationen

Die Tabelle 4.3 zeigt die summarische Zusammenfassung von der aufrufbare Methode in »ViewFunction.Transaction« aus »view4bpel.jar«. Zusammen mit Listing 4.19 sollen eine behilfliche Anleitung für den richtige Anwendung von »view4bpel.jar« gegeben werden.

View-Funktion	Kurzerklärung
transform()	transformiert Basisprozess durch Rules-Dokument
setAFocus()	fokussiert auf die Aktivität
removeFragment()	eliminiert das Teilprozess

Tabelle 4.3.: Kurzfassung von allen View-Funktionen

Listing 4.19 zeigt beispielsweise, wie die alle Funktionen aufgerufen werden. Es ist auch einen ausführbaren Beispielcode, die eine erfolgreiche Installation von »view4bpel.jar« beweist. Die entsprechende Dokumenten sowie Prozessbeispiele und rules-Dokumente für diesen Beispielcode werden in den Installationspaket gefunden.

Listing 4.19 Die Nachprüfung von der Installation

```
import java.io.IOException;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPathExpressionException;
import org.dom4j.DocumentException;
import ViewsFunction.Transaction;


public class test {

	public static void main(String[] args) throws IOException, XPathExpressionException,
	    ParserConfigurationException, DocumentException {

	       Transaction ts = new Transaction();
	       ts.setAFocus("Travelbooking.bpel", "DataMap2", 2, 2,false);
	       ts.setAFocus("Travelbooking.bpel", "DataMap4", 2, 2,false);
	       ts.transform("testCase01.bpel", "Rule01_testCase01.xml",false);
	       ts.removeFragment("Travelbooking.bpel", "businessActivities.bpel",false);
	       ts.transform("Approval-in-standard-code.bpel", "Rule_removeIdInfo.xml",false);


	}
}
```

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

Eine anwendungsspezifische Abstraktion von Basisgeschäftsprozess wird immer mehr notwendig, um die Komplexität von Geschäftsprozess zu reduzieren und die kritische Geschäftsinformationen zu verbergen. Eine generierte Sicht bzw. »Process View« von Basisprozess steht eine abstrakte oder partielle Prozesssemantik für den entsprechenden Anwendungsfällen zur Verfügung. Die Methode und effektive Algorithmen für den »Process View« wurden schon seit ein paare Jahren eingearbeitet, es wird die manche erreichte Ergebnissen von »Process View« in den Arbeiten wie [BRB07] [CDT06] [LS03] [RBRB06] gefunden.

Die vorhanden Workflow-Management-Systemen unterstützen die Funktionalität von »Process View« nicht, es wird in der Arbeit einen Anwendungsprogramm auf den Basisarbeit von [Str09] für den »Process View« entwickelt. In den Programm wird es mehrere einfachen View-Operationen angeboten. Durch eine sinnvolle Kombination von primitive View-Operationen in den Rules-Dokument wird der Basisprozess sequenziell transformiert, eine abstrakte Sicht von Basisprozess wird generiert. Die entwickelte View-Operationen und View-Funktionen stellen eine effiziente Verarbeitungsfähigkeit für den »Process View« dar.

Es verfügt eine hohe Erweiterbarkeit für den View-Funktionen bei den Weiterentwicklung, eine neue anwendungsspezifische Abstraktion von BPEL-Prozess können durch eine neue Kombination von View-Operationen realisiert werden. Nach die vertraute Erkenntnisse von View-Operationen und Rules-Sprache wird jede zielorientierte Transformation von Basisprozess erreichbar.

5.2. Ausblick

Das in der Arbeit entwickelt Anwendungsprogramm wird sich nur für den Prozessexperten oder IT-Spezialisten orientiert. Das Applikationsprogramm wird als einen webbasierte Extension-Funktion für den Prozessverarbeitung in den Webserver integriertet, die Prozessspezialisten erstellen den Rules-Dokument und rufen die Webanwendung auf, die entsprechende Prozessverarbeitung wird durch XML-basierte Datentransformation implementiert.

Die Vergrößerung von dem Anwendungsgebiet ist auch möglich. Einer Ausbau in den Prozesswerkzeug können in den weiter Arbeit realisiert werden. Es wird zahlreiche Werkzeugen

für den Modellierung von Geschäftsprozess entwickelt, die die von W3C standardisierte WS-BPEL Spezifikation befolgen. Eine installierbare Erweiterung in den Prozesswerkzeugen z.B Eclipse-Extension »BPEL Designer« können solche Anwendungen schaffen, sodass es außer die Modellierung von BPEL-Prozess auch die Funktionen »Process View« unterstützt.

Literaturverzeichnis

[AAA+] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guızar, N. Kartha, et al. Web services business process execution language version 2.0. (Zitiert auf den Seiten 5, 7, 9, 16, 17, 18, 46 und 49)

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, et al. Business process execution language for web services, version 1.1. *Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation*, 2003.

[BBC+06] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, et al. Web Services policy 1.2-framework (WS-policy). *W3C Member Submission*, 25, 2006. (Zitiert auf Seite 12)

[BHLT06] T. Bray, D. Hollander, A. Layman, R. Tobin. Namespaces in XML 1.0. *World Wide Web Consortium, Recommendation REC-xml-names-20060816*, 2006. (Zitiert auf Seite 21)

[BHM+04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. Web Services Architecture, W3C Working Group Note 11 February 2004. *World Wide Web Consortium, article available from: http://www. w3. org/TR/ws-arch*, 2004. (Zitiert auf Seite 13)

[BLFM98] T. Berners-Lee, R. Fielding, L. Masinter. *Uniform resource identifiers (URI): generic syntax*. RFC 2396, August 1998, 1998. (Zitiert auf Seite 21)

[BRB07] R. Bobrik, M. Reichert, T. Bauer. *View-based process visualization*. Springer, 2007. (Zitiert auf Seite 54)

[CD+99] J. Clark, S. DeRose, et al. XML path language (XPath) version 1.0, 1999. (Zitiert auf Seite 22)

[CDT06] I. Chebbi, S. Dustdar, S. Tata. *The view-based approach to dynamic inter-organizational workflow cooperation*, volume 56. Elsevier, 2006. (Zitiert auf Seite 54)

[CFNO02] M. Champion, C. Ferris, E. Newcomer, D. Orchard. Web Services Architecture, W3C Working Draft 14 November 2002, 2002. (Zitiert auf den Seiten 11 und 12)

[CHR+04] L. Clement, S. Hately, I. von Riegen, S. AG, T. Bellwood, I. Capell, S. Colgrave, I. Dovey, P. Macias, L. Novotny, et al. UDDI Version 3.0. 2 UDDI Spec Technical Committee Draft, Dated 20041019. *Organization for the Advancement of Structured Information Standards (OASIS), October*, 19:0–2, 2004. (Zitiert auf Seite 12)

[CMRW07] R. Chinnici, J. Moreau, A. Ryman, S. Weerawarana. Web services description language (WSDL) version 2.0 part 1: Core language. Technical report, Technical report, World Wide Web Consortium, 2007. (Zitiert auf den Seiten 10, 11 und 12)

[CT99] J. Cowan, R. Tobin. XML information set. 1999.

[CW01] F. Curbera, S. Weerawarana. *Web Services Description Language (WSDL) 1.1*. Citeseer, 2001. (Zitiert auf Seite 13)

[DJMZ05] W. Dostal, M. Jeckle, I. Melzer, B. Zengler. Service-orientierte Architekturen mit Web Services–Konzepte, Standards. *Praxis*, 1, 2005. (Zitiert auf den Seiten 10, 12 und 16)

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. *Hypertext transfer protocol–HTTP/1.1*. RFC 2616, June 1999, 1999. (Zitiert auf Seite 12)

[Fou] E. Foundation. Eclipse IDE. `http://www.eclipse.org/`. (Zitiert auf Seite 24)

[FW04] D. Fallside, P. Walmsley. XML Schema Part 0: Primer Second Edition, W3C Recommendation 28 October 2004. *World Wide Web Consortium*, pp. 0–20041028, 2004. (Zitiert auf Seite 21)

[Gla] S. GlassFish. Java API for XML Processing. `https://jaxp.dev.java.net/`. (Zitiert auf Seite 24)

[H+94] D. Hollingsworth, et al. *Workflow management coalition: The workflow reference model*. Citeseer, 1994. (Zitiert auf Seite 15)

[IBM] IBM. WebSphere Business Process Management Samples und Tutorials. `http://publib.boulder.ibm.com/bpcsamp/`. (Zitiert auf den Seiten 3, 17, 29 und 62)

[KKL+05] M. Kloppmann, D. König, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, I. Trickovic. WS-BPEL Extension for Sub-processes–BPEL-SPE. *Joint white paper, IBM and SAP*, 2005. (Zitiert auf Seite 49)

[KLM+08] D. König, N. Lohmann, S. Moser, C. Stahl, K. Wolf. Extending the compatibility notion for abstract WS-BPEL processes. In *Proceeding of the 17th international conference on World Wide Web*, pp. 785–794. ACM, 2008.

[L+01] F. Leymann, et al. *Web services flow language (WSFL 1.0)*. May, 2001. (Zitiert auf Seite 16)

[LR00] F. Leymann, D. Roller. *Production workflow: concepts and techniques*. Prentice Hall PTR, 2000. (Zitiert auf den Seiten 15 und 20)

[LS03] D. Liu, M. Shen. *Workflow modeling for virtual processes: an order-preserving process-view approach* 1*, volume 28. Elsevier, 2003. (Zitiert auf Seite 54)

[Meg] D. Megginson. Simple API for XML 2.0. `http://www.saxproject.org/about.html`. (Zitiert auf Seite 24)

[Mica] S. Microsystems. JAR File Specification. http://java.sun.com/javase/6/docs/technotes/guides/jar/jar.html. (Zitiert auf Seite 8)

[Micb] S. Microsystems. Java. http://java.sun.com/. (Zitiert auf Seite 24)

[ML03] N. Mitra, Y. Lafon. Soap version 1.2 part 0: Primer. *W3C Recommendation*, 24, 2003. (Zitiert auf Seite 12)

[OAS] OASIS. OASIS Web Services Business Process Execution Language (WSBPEL) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. (Zitiert auf den Seiten 14 und 16)

[OSP] Open-Source-Projekt. Dom4J. http://dom4j.sourceforge.net/. (Zitiert auf Seite 24)

[RBRB06] S. Rinderle, R. Bobrik, M. Reichert, T. Bauer. Business process visualization - use cases, challenges, solutions. In *Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS'06)*, pp. 204–211. Citeseer, 2006. (Zitiert auf Seite 54)

[Ros06] F. Rosenkranz. *Geschäftsprozesse*. Springer-Verlag Berlin Heidelberg, 2006. (Zitiert auf Seite 15)

[Sch08a] M. Schäfer. Business Process Outsourcing. 2008. (Zitiert auf den Seiten 7 und 15)

[Sch08b] D. Schumm. *Graphische Modellierung von BPEL Prozessen unter Verwendung der BPMN Notation*. Universität Stuttgart, Fakultät Informatik, Diplomarbeit, 2008.

[SG07] H. Sieck, A. Goldmann. *Erfolgreich verkaufen im B2B*. Gabler, 2007. (Zitiert auf Seite 7)

[Str09] A. Streule. Abstract Views on BPEL Processes. 2009. (Zitiert auf den Seiten 7, 8, 15, 24, 25, 26, 27, 28, 32, 33, 36, 38, 43, 44, 46, 47 und 54)

[Suna] Sun. Package javax.xml.xpath. http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/package-summary.html. (Zitiert auf Seite 38)

[Sunb] Sun. Package org.w3c.dom. http://java.sun.com/j2se/1.4.2/docs/api/org/w3c/dom/package-summary.html. (Zitiert auf Seite 39)

[Tea] B. D. Team. BPEL Project. http://www.eclipse.org/bpel/index.php. (Zitiert auf Seite 24)

[Tha01] S. Thatte. Xlang. *Specification, Microsoft Corp*, 2001. (Zitiert auf Seite 16)

[W3Ca] W3C. Document Object Model (DOM). http://www.w3.org/DOM/. (Zitiert auf Seite 23)

[W3Cb] W3C. Web of Services. http://www.w3.org/standards/webofservices/. (Zitiert auf Seite 13)

[W3Cc] W3C. World Wide Web Consortium. http://www.w3.org/. (Zitiert auf Seite 12)

[W3C99] W3C. XSL Transformations, Version 1.0, W3C Recommendation 16 November 1999. `http://www.w3.org/TR/xslt`, 1999. (Zitiert auf Seite 22)

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D. F. Ferguson. *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, 2005. (Zitiert auf den Seiten 7, 10, 12 und 16)

[WFM] WFMC. Workflow Reference Model Diagram. `http://www.wfmc.org/reference-model.html`. (Zitiert auf Seite 15)

[YBP+04] F. Yergeau, T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler. *Extensible markup language (xml) 1.0*. Citeseer, 2004. (Zitiert auf Seite 20)

Alle URLs wurden zuletzt am 20.04.2009 geprüft.

A. Anhang

A.1. Rules-Dokument Schema

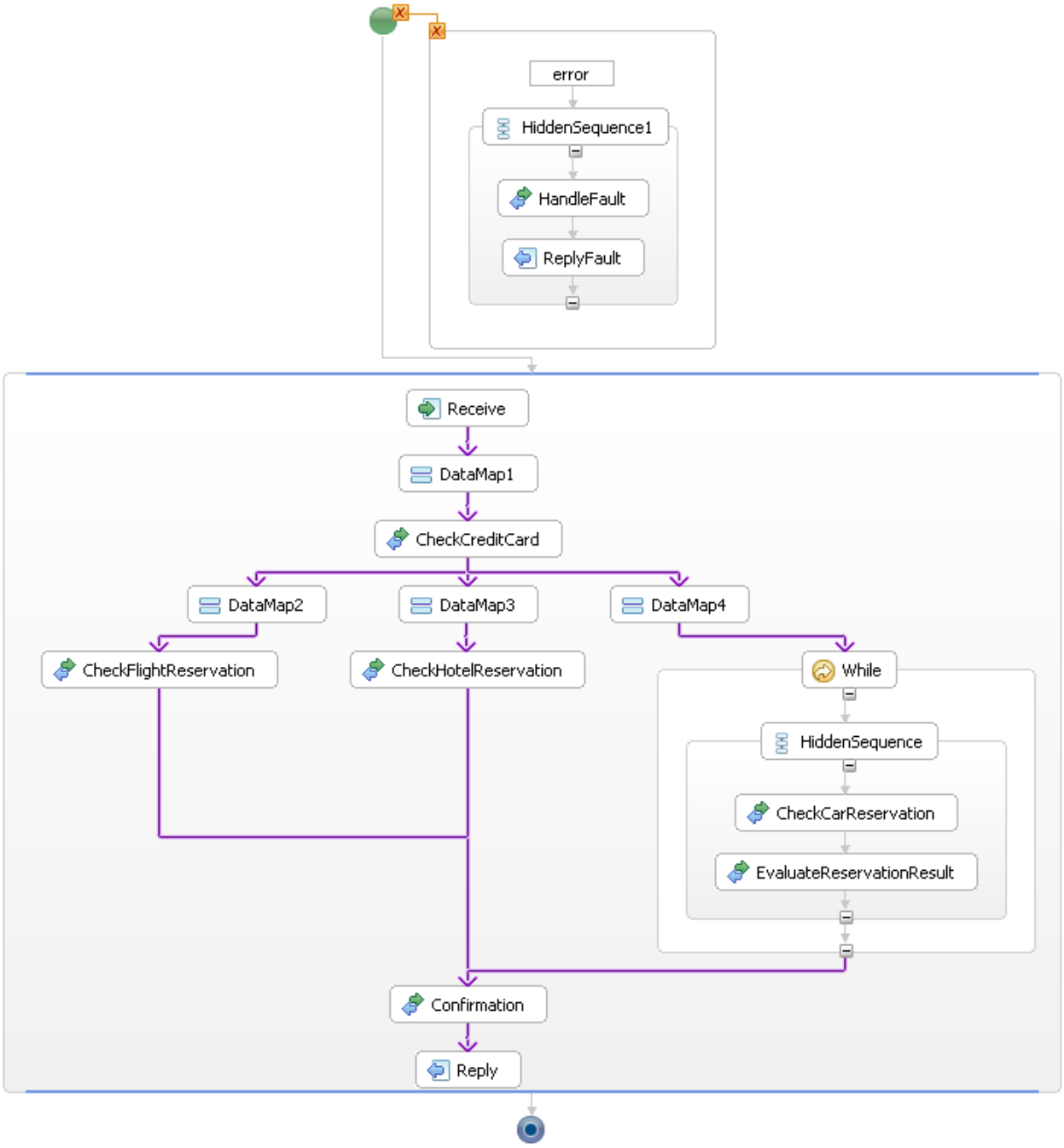
```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://www.eclipse.org/bpel/views/rules"
    xmlns:tns="http://www.eclipse.org/bpel/views/rules">

    <!-- here rules schema -->

    <element name="rules">
        <complexType>
            <sequence maxOccurs="1">
                <element name="parameter" type="tns:parameterType"></element>
                <sequence maxOccurs="unbounded">
                    <element name="rule" type="tns:ruleType"></element>
                </sequence>
            </sequence>
            <attribute name="name" type="string"></attribute>
        </complexType>
    </element>

    <complexType name="ruleType">
        <sequence>
            <element name="actions" type="tns:actionsType"></element>
            <element name="targets" type="tns:targetsType"></element>
        </sequence>
        <attribute name="name" type="string" use="optional"></attribute>
        <attribute name="apply" type="boolean" use="optional"></attribute>
    </complexType>

    <complexType name="actionSetAttributeToType">
        <attribute name="attributeName" type="string"></attribute>
        <attribute name="value" type="string"></attribute>
    </complexType>

    <complexType name="actionsType">
        <sequence>
            <choice maxOccurs="unbounded" minOccurs="1">
                <element name="actionOpaque" type="tns:actionOpaqueType">
                </element>
                <element name="actionOmit" type="tns:actionOmitType">
                </element>
                <element name="actionSetAttributeTo" type="tns:actionSetAttributeToType">
                </element>
```

```
            <element name="addPreserve" type="tns:addPreserveType">
            </element>
            <element name="addTagged" type="tns:addTaggedType">
            </element>
        </choice>
    </sequence>
</complexType>

<complexType name="targetsType">
    <sequence>
        <choice maxOccurs="unbounded" minOccurs="1">
            <element name="and" type="tns:targetsType">
            </element>
            <element name="or" type="tns:targetsType">
            </element>
            <element name="not" type="tns:targetsType">
            </element>
            <element name="tag" type="tns:tagType">
            </element>
            <element name="attribute" type="tns:attributeType">
            </element>
            <element name="type" type="tns:typeType">
            </element>
        </choice>
    </sequence>
</complexType>

<complexType name="attributeType">
    <attribute name="attributeName" type="string"></attribute>
    <attribute name="value" type="string"></attribute>
</complexType>

<complexType name="actionOpaqueType">
    <attribute name="preserveAttributes" type="boolean" use="optional">
    </attribute>
</complexType>

<complexType name="actionOmitType">
    <attribute name="preserveChildren" type="boolean" use="optional">
    </attribute>
    <attribute name="preserveTransitionConditions" type="boolean"
        use="optional">
    </attribute>
</complexType>

<complexType name="actionOmitAttributeType">
    <attribute name="attribute" type="string"></attribute>
</complexType>

<complexType name="typeType">
    <attribute name="typeName" type="string"></attribute>
</complexType>

<complexType name="tagType">
    <attribute name="tagName" type="string"></attribute>
```

```
    </complexType>

    <complexType name="addPreserveType">
        <attribute name="value" type="boolean" use="optional">
        </attribute>
    </complexType>

    <complexType name="addTaggedType">
        <attribute name="value" type="string" use="required">
        </attribute>
    </complexType>

    <!-- here parameter schema -->

    <complexType name="parameterType">
        <sequence maxOccurs="unbounded">
            <choice maxOccurs="unbounded" minOccurs="1">
                <element name="cleaning" type="tns:cleaningType">
                </element>
                <element name="aggregateOpaque" type="tns:aggregateOpaqueType">
                </element>
            </choice>
        </sequence>
    </complexType>

    <complexType name="cleaningType">
        <attribute name="value" type="boolean">
        </attribute>
    </complexType>

    <complexType name="aggregateOpaqueType">
        <attribute name="value" type="boolean">
        </attribute>
    </complexType>

</schema>
```

A.2. Anwendungsfälle

A.2.1. TravelBooking Prozessdiagramm [IBM] in BPEL-Designer


error
HiddenSequence1
HandleFault
ReplyFault
Receive
DataMap1
CheckCreditCard
DataMap2
DataMap3
DataMap4
CheckFlightReservation
CheckHotelReservation
While
HiddenSequence
CheckCarReservation
EvaluateReservationResult
Confirmation
Reply

A.2.2. Process View I: Fokussieren auf »CheckCreditCard«

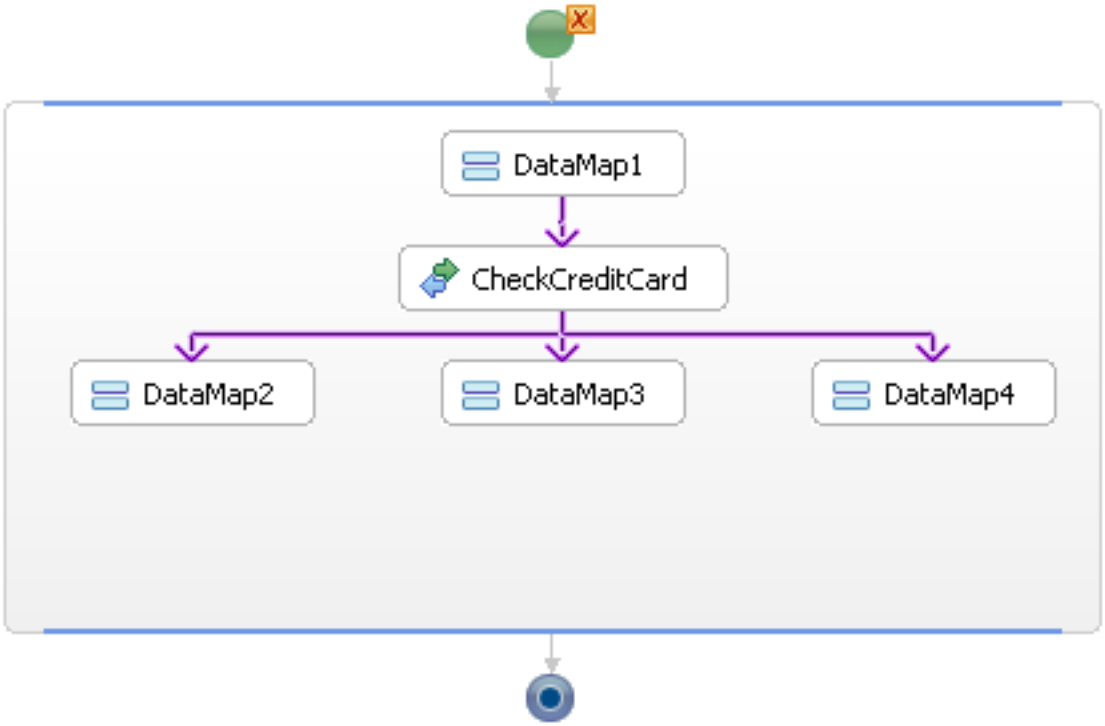


Abbildung A.1.: Fokussieren auf Aktivität »CheckCreditCard«

Listing A.1 Das von »SetAFocus()« generiert Rules-Dokument für das Fokussieren

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:rules xmlns:tns="http://www.eclipse.org/bpel/views/rules"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="setAFocus_CheckCreditCard"
    xsi:schemaLocation="http://www.eclipse.org/bpel/views/rules Rules.xsd">
    <parameter>
        <aggregateOpaque value="true" />
        <cleaning value="true" />
    </parameter>
    <rule apply="true" name="addPreserve">
        <actions>
            <addPreserve />
        </actions>
        <targets>
            <or>
                <attribute attributeName="name" value="CheckCreditCard" />
                <attribute attributeName="name" value="DataMap1" />
                <attribute attributeName="name" value="DataMap2" />
                <attribute attributeName="name" value="DataMap3" />
                <attribute attributeName="name" value="DataMap4" />
            </or>
        </targets>
    </rule>
    <rule apply="true" name="Omission">
        <actions>
            <actionOmit />
        </actions>
        <targets>
            <tag tagName="*" />
        </targets>
    </rule>
</tns:rules>
```

A.2.3. Process View II: Teilprozesseliminierung

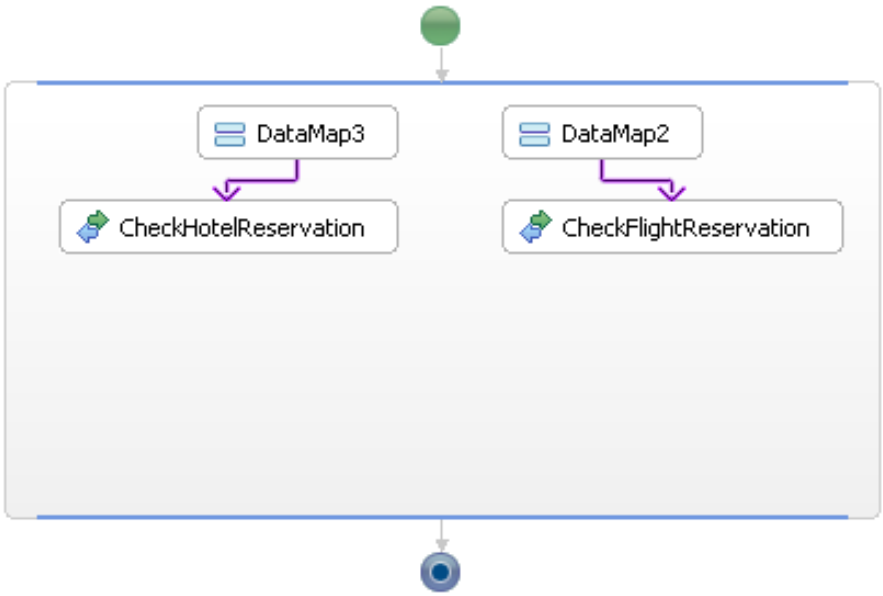


Abbildung A.2.: Teilprozesse, die eliminiert werden sollen

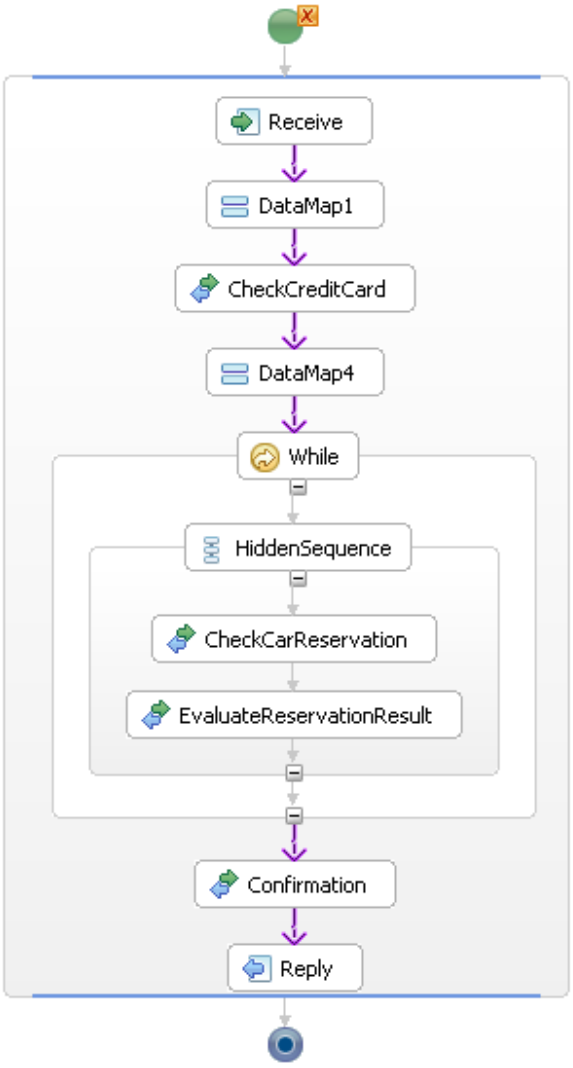


Abbildung A.3.: Das resultierende Prozessfragment nach der Eliminierung

Listing A.2 Das von »removeFragment()« generiert Rules-Dokument für die Teilprozesseliminierung

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tns:rules xmlns:tns="http://www.eclipse.org/bpel/views/rules"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="removeFragmentTravelBooking"
    xsi:schemaLocation="http://www.eclipse.org/bpel/views/rules Rules.xsd">
    <parameter>
        <aggregateOpaque value="true" />
        <cleaning value="true" />
    </parameter>
    <rule apply="true" name="OmitProcessFragment">
        <actions>
            <actionOmit />
        </actions>
        <targets>
            <attribute attributeName="name" value="DataMap3" />
            <attribute attributeName="name" value="CheckHotelReservation" />
            <attribute attributeName="name" value="DataMap2" />
            <attribute attributeName="name" value="CheckFlightReservation" />
        </targets>
    </rule>
</tns:rules>
```

A.3. Quelltext (Auszugsweise)

A.3.1. Algorithmus für die Generierung von Xpath-Ausdrucksformeln

```
    private String ExpressionXpath(Node node, String Expression) {
        // read each target in rule document and turn it to a Xpath-Expression
        // in string

        if (!checkLogikTerm(node)) {
            if (node.getNodeName().equals("tag"))
            // check for the operation on all constructs
            {
                if (((Element) node).getAttributeNode("tagName").getValue()
                        .equals("*")) {
                    System.out.println("Operation: * : for all constructs ");
                    Expression = "//*";
                } // get the tag condition in the predicate of xpath-expression
                else
                    return "contains(./*/text(), " + "'" + "@tag "
                            + ((Element) node).getAttribute("tagName") + "'"
                            + ")";

            }
            if (node.getNodeName().equals("attribute")) { // get the attribute
                                                          // condition in the
                                                          // predicate of
```

```
                                                    // xpath-expression
            return "@" + ((Element) node).getAttribute("attributeName")
                    + "=" + "'" + ((Element) node).getAttribute("value")
                    + "'";
            // System.out.println(Expression);
        }
        if (node.getNodeName().equals("type")) { // check for the operation
                                                 // on all activities
            if (((Element) node).getAttributeNode("typeName").getValue()
                    .equals("*")) {
                System.out.println("Operation: * : for all activities ");
                Expression = "name()='" + namespace + "invoke' or name()='"
                        + namespace + "receive' or name()='" + namespace
                        + "reply' " + "or name()='" + namespace
                        + "assign' or name()='" + namespace
                        + "throw' or name()='" + namespace + "wait'"
                        + "or name()='" + namespace + "exit' or name()='"
                        + namespace + "empty' or name()='" + namespace
                        + "rethrow'" + "or name()='" + namespace
                        + "extensionActivity' or name()='" + namespace
                        + "link' or name()='" + namespace + "sequence'"
                        + "or name()='" + namespace + "if' or name()='"
                        + namespace + "while' or name()='" + namespace
                        + "repeatuntil'" + "or name()='" + namespace
                        + "pick' or name()='" + namespace
                        + "flow' or name()='" + namespace + "foreach'"
                        + "or name()='" + namespace + "scope'";
                // System.out.println(Expression);
            } else
                // get the type condition in the predicate of
                // xpath-expression
                return "name()" + "=" + "'" + "" + namespace + ""
                        + ((Element) node).getAttribute("typeName") + "'";
            // System.out.println(Expression);
        }
    } else { // bind the no logic notation of inside Terms in the predicate
                // of xpath-expression
        if (node.getNodeName().equals("not")) {
            NodeList nl = node.getChildNodes();
            for (int y = 0; y < nl.getLength(); y++) {
                Node childNode = nl.item(y);
                if (childNode.getNodeType() == Node.ELEMENT_NODE) {
                    Expression = "not(" + ExpressionXpath(childNode, "")
                            + ")";
                    // System.out.println(childNode.getNodeName());
                }
            }
        }
        // bind the and logic notation of all inside Terms in the predicate
        // of xpath-expression
        if (node.getNodeName().equals("and")) {
            NodeList nl = node.getChildNodes();
            for (int y = 0; y < nl.getLength(); y++) {
                Node childNode = nl.item(y);
                if (childNode.getNodeType() == Node.ELEMENT_NODE) {
```

```
                    String Expression1 = ExpressionXpath(childNode, "");
                    String Expression2 = "";
                    if (Expression != "")
                        Expression2 = Expression + " and " + Expression1;
                    else
                        Expression2 = Expression1;
                    Expression = Expression2;
                }
            }
            // System.out.println(Expression);
        }
        // bind the or logic notation of all inside Terms in the predicate
        // of xpath-expression
        if (node.getNodeName().equals("or")) {
            NodeList nl = node.getChildNodes();
            for (int y = 0; y < nl.getLength(); y++) {
                Node childNode = nl.item(y);
                if (childNode.getNodeType() == Node.ELEMENT_NODE) {
                    String Expression1 = ExpressionXpath(childNode, "");

                    String Expression2 = "";
                    if (Expression != "")
                        Expression2 = Expression + " or " + Expression1;
                    else
                        Expression2 = Expression1;
                    Expression = Expression2;

                }
            }
            // System.out.println(Expression);
        }

    }
    return "(" + Expression + ")";
}
```

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Jiayang Cai , am 17.05.2010)