Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2287

Benchmarking IBM Process Server

Bo Cao

Studiengang:	Informatik
Prüfer:	Prof.Dr.Frank Leymann
Betreuer:	Dipl.-Phys.Dieter H.Roller
Begonnen am:	01.07.2010
Beendet am:	31.12.2010
CR-Klassifikation:	C.4, H.3.4, H.4.1, J.1

Inhaltsverzeichnis

1 Einführung

Das erste Kapitel stellt die Struktur der Studienarbeit dar und es wird ein Überblick über diese Arbeit gegeben. Das Hauptziel ist, die Installierung der Textumgebung und die Implementierung der Textprogramm.

1.1Motivation

WebSphere ist eine Produktlinie der Firma IBM, die unterschiedliche Software für Anwendungsintegration, Infrastruktur (z. B. Transaktionen und Warteschlangen) und eine integrierte Entwicklungsumgebung umfasst.[1] Die IBM-Websphere Produktfamilie bietet für die verschiedenen Anforderungen im „e-business“ eine Reihe von skalierbaren Produkten an. WebSphere-Produkte sind seit vielen Jahren auf dem Markt, in den meisten Bereichen mittlerweile Marktführer

Als Grundlagentechnologie setzen diese Produkte auf Industriestandards wie Java (J2EE), XML und WebServices sowie auf Plattformunabhängigkeit. Alle Produkte sind grundsätzlich „offen“ konstruiert, so dass eine Integration mit bestehenden Produkten anderer Hersteller (auch anderer Technologien) möglich ist. Über die standardisierten Schnittstellen wie „WebServices“ können verschiedene Anwendungen (auch verschiedener Implementierungsformen, z.B. .NET) integriert werden, durch die verschiedenen Datenbankschnittstellen (Oracle, MS SQL-Server etc.) werden auch andere Datenbanken als die IBM DB/2 unterstützt. Die Betriebssystemplattform ist grundsätzlich egal, von Windows-Servern über Linux, AIX bis zur AS/400 ist alles einsetzbar. Durch den modularen Aufbau können die einzelnen Komponenten leicht miteinander - und mit Drittkomponenten - kombiniert werden. [2]

Die IBM-WebSphere-Produktfamilie gliedert sich in drei Bereiche:

Abbildung 1-1: Gliederung von WebSphere-Produkt

Mit den anderen Produktlinien DB2, Lotus, Tivoli und Rational bietet IBM eine

umfassende und ausgereifte Produktlinie für das gesamte Middleware-Spektrum an. So wird z.B. der DB2 Content-Manager problemlos gegen FileNet, IXOS und andere Archivierungssoftware positioniert, die aktuelle Version der Datenbank DB2 8.2 wird gegen Oracle positioniert, und Rational ist DER Hersteller von Entwicklungstools, die alle Prozesse in der Entwicklung (von UML über Team-Entwicklungsfunktionen und Teststools mit Dokumentation und Config-Management bis zum Change-Management) abbilden.

Auch im WebSphere-Bereich setzt die WebSphere-Produktfamilie durchgängig auf offene Plattformen: Angefangen beim Betriebssystem (Linux), der Laufzeitumgebung (Java-Applicationserver und Webserver) über die Integrations-Techniken (Web-Services) bis zur Entwicklungsumgebung (Eclipse) sind im Prinzip ALLE notwendigen Tools und Techniken frei verfügbar:

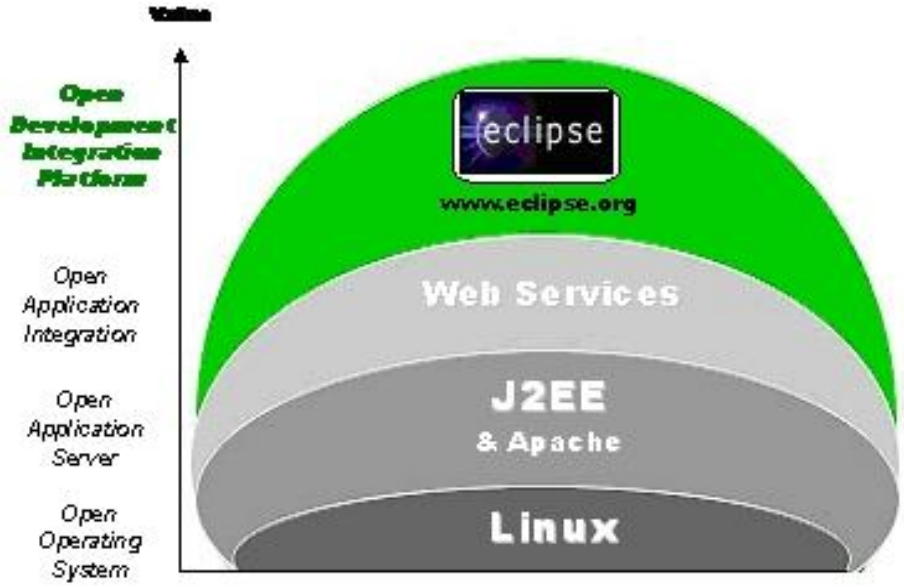


Abbildung 1-2: Die offene Plattformen der WebSphere-Produktfamilie

1.2 Ziel und Ablauf der Studienarbeit

Das Ziel dieser Studienarbeit ist es, einige Bpel-Programme zu implementieren. Es besteht aus zwei Teilen:

- Die Initialisierung des IBM® Process Server
- Die Ausführung von Programme auf IBM® Process Server
- Aufzeichung des Ergebnisse

2 Laufzeitumgebung

In diesem Kapital werden die erforderliche Laufzeitumgebung und das eingesetzte Betriebssystem vorgestellt.

2.1 Eingesetzte Software und Hardware

Um das Text- Programm zum Laufen zu bringen, wurde IBM® WebSphere® Application Server V7.0.0 (1.0.0.20080911_1339) als Laufzeitumgebung installiert. Die Installation wurde durch IBM® Installation Manager Version 1.4.1 (1.4.1000.20190810_1125) erfolgreich ausgeführt. Als Datenbank wurde IBM® DB2 Express-C Version 9.7.1 verwendet .

Das Testsystem wird wie folgt eingesetzt:

Hardwareumgebung:

Prozessor: Pentium® Dual-Core T4500 2.30GHz

Arbeitsspeicher: 3GB RAM

Festplatte: 250GB

Softwareumgebung:

Betriebssystem: Microsoft Windows XP Home Edition mit Service Pack 2

2.2 IBM® WebSphere® Application Server

WebSphere Application Server(WAS) ist ein bekanntes WebSphere-Produkt. WAS wird sehr oft auch einfach mit WebSphere bezeichnet. Der funktioniert als Middleware und bietet ein Programmiermodell, ein Infrastrukturrahmen und viele Stardards für eine konsenquent entworfene Verbindung zwischen Back-End Systemen und Clients.

WebSphere Application Server V 7.0 bietet folgende Funktionalitäten[3]:

- Application Server Zweck
- Entwicklung der Java Application Development Standard
- Verbesserte Verwaltung
- Leichtere Integration
- Extra Werkzeuge und Erweiterungen

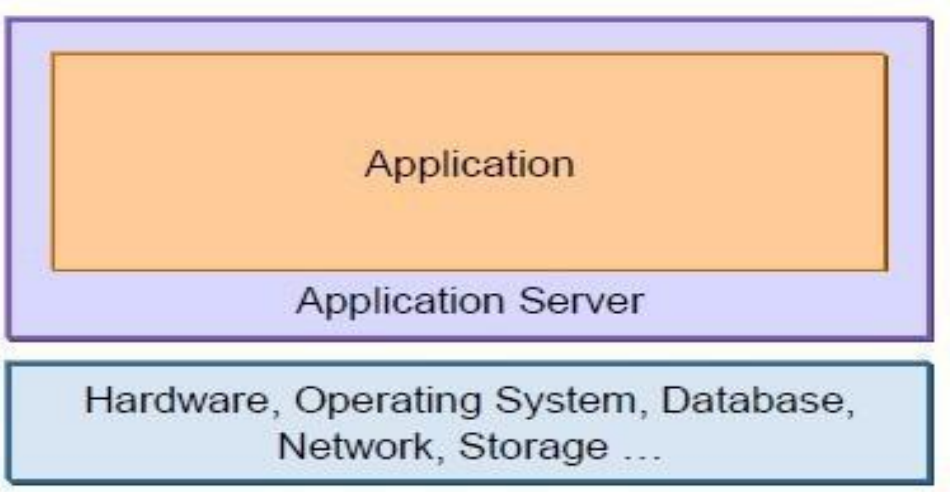


Abbildung 2-1: Darstellung eines Applicationsservers und seiner Umgebung[3]

WebSphere Application Server bietet die Umgebung für die Ausführung der Solution und integriert die Solution mit jeder Plattform und System als die Services der Geschäftsanwendungen nach SOA-Struktur.

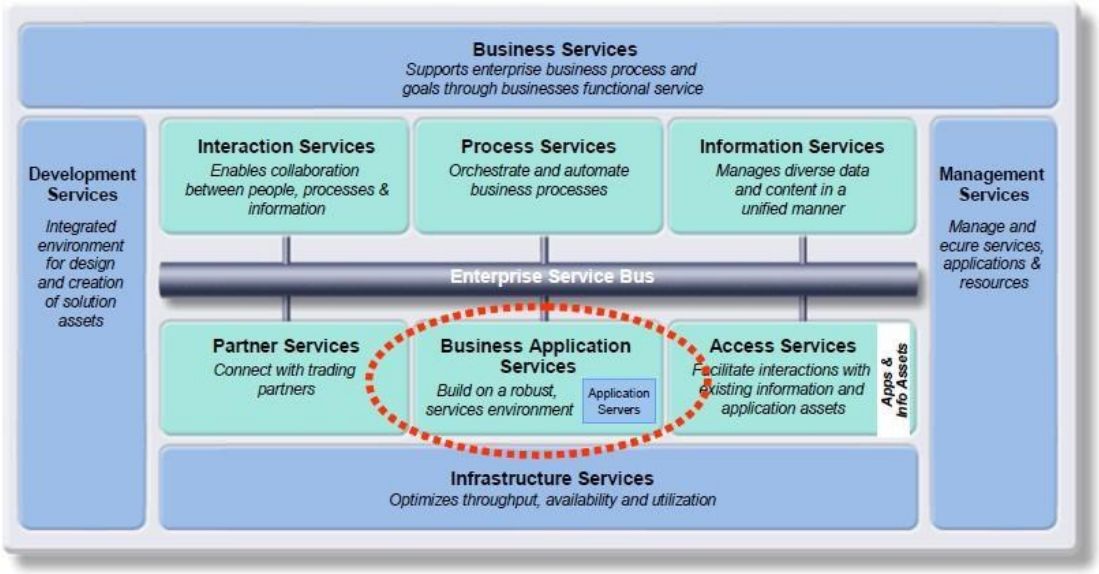


Abbildung 2-2: Die Position des Business Application Services in SOA-Struktur

WebSphere Application Server ist ein zentraler SOA-Baustein. Von einer SOA-Aussicht kann man mit WAS die folgenden Funktionen ausführen:

• Schnell und einfach die wiederverwendbare Anwendungsservices erstellen und bereitstellen.

• Ausführung der Services in einer sicheren, skalierbaren, hochverfügbaren Umgebung

• Verbindet Softwarewert und erweitert dessen Reichweit

• Einfache Anwendungsverwaltung

• Erfüllung hoher Anforderungen und steigerung der Kernwiederverwendbarkeit

Mit zunehmenden geschäftlichen Anforderungen, stehen immer mehr neue Technologie- Standards zur Verfügung. Seit 1998 ist WebSphere stark gewachsen und passt sich mit neuen Technologien und neuen Standards für eine innovative spitze Umgebungen, um den Entwurf der voll integrierten Solution zu ermöglichen und die Geschäftsanwendungen zu realisieren.[3]

2.3 IBM DB2

DB2 ist ein kommerzielles relationales Datenbankmanagementsystem (RDBMS) der Firma IBM, dessen Ursprünge auf das System R und die Grundlagen von E. F. Codd vom IBM Research aus dem Jahr 1970 zurückgehen. DB2 verwaltet Daten in Tables und speichert sie in Tablespaces. DB2 unterstützt neben den Standard-SQL-Datentypen auch binäre Datentypen (Text, Töne, Bilder, Videos, XML-Daten). Zusammen mit Oracle Database und MS SQL-Server gehört DB2 zu den Datenbanksystemen mit den größten Marktanteilen. [4]

Innerhalb einer DB2 Installation existieren die folgenden wichtigen Datenbanksystem Objekte:

- Instanzen
- Datenbanken
- Schemata
- Tabellen
- Views

Instanz

Eine Instanz (auch: Datenbankmanager) ist ein Objekt, das Daten verwaltet. Es kontrolliert, was mit den Daten gemacht werden kann, und verwaltet die ihm zugewiesenen Systemressourcen.

Datenbank

Eine relationale Datenbank ist eine Sammlung von Tabellen. Eine Tabelle besteht aus einer festen Zahl von Spalten, die den Attribute des Relationenschemas entsprechen, sowie einer beliebigen Zahl an Reihen.

Tabellen

Eine relationale Datenbank speichert Daten als eine Menge von zweidimensionalen Tabellen. Jede Spalte der Tabelle repräsentiert ein Attribut des Relationenschemas, jede Zeile entspricht einer spezi_schen Instanz dieses Schemas.

Views

Ein View ist eine e_ziente Art, um Daten wiederzugeben, ohne sie in einer Tabelle zu speichern. Ein View ist keine echte Tabelle, sondern eine Sicht auf einen Ausschnitt der Datenbank, und braucht keinen permanenten Platz. [5]

DB2 kennt die in der nachstehenden Grafik abgebildeten Datentypen.Weitere Datentypen können vom Benutzer definiert werden.

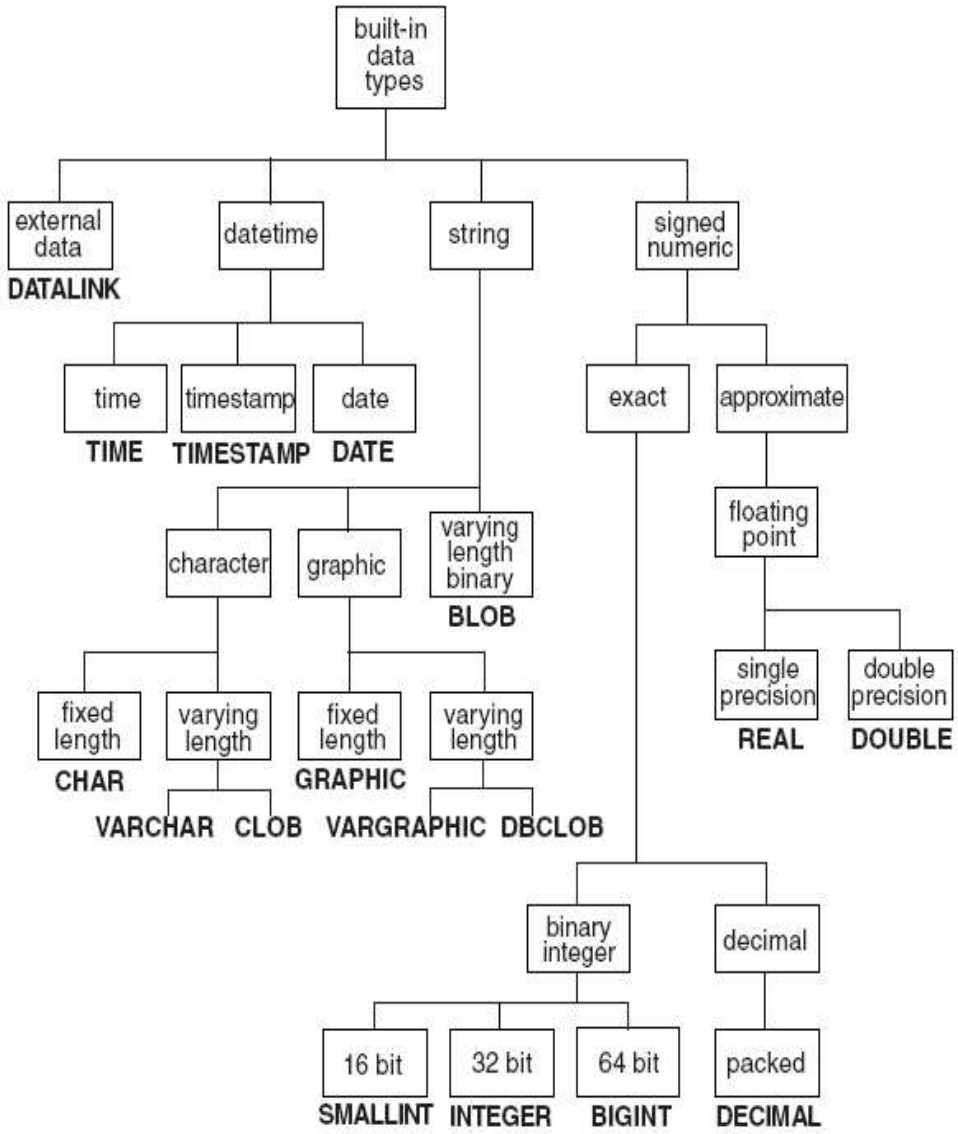


Abbildung 2-3: Datentypen in DB 2

2.4 WebSphere Integration Developer

Der WebSphere Integration Developer basiert auf der Eclipse Plattform und enthält fast alle Features aus dem Rational Application Developer. Zusätzlich sind neue Perspektiven enthalten, die über weitgehend grafisch orientierte Editoren den einfachen Aufbau der dynamischen Prozesse in BPEL ermöglichen. Die Entwicklung

wird größtenteils über XML-Standards realisiert, wobei ausschließlich mit BPEL, WSDL, Service Component Architecture (SCA) und Service Data Objects (SDOs) gearbeitet wird. Der WebSphere Integration Developer verfügt über ein integriertes Testsystem des WebSphere Process Servers, das Debugging von Prozessen und Mediation Flows ermöglicht. Seit v6.2 sind auch automatisierte Tests von SCA-Komponenten möglich.

Im WebSphere Integration Developer entwickeln Sie vornehmlich Prozessmodelle mit der Business Process Execution Language (BPEL), die Sie entweder aus einem WebSphere Business Modeler importiert haben oder selbst erstellen und auf einem WebSphere Process Server deployen. Dabei liegt der Schwerpunkt der Entwicklung von Integrationsanwendungen nicht auf der reinen Programmierung sondern vielmehr auf der Assemblierung der Anwendungen. Der WebSphere Integration Developer ist daher für die BPEL-Welt weitgehend grafisch orientiert. Die BPEL-Modelle kommunizieren über WebService-Schnittstellen mit IT-Systemen. [6]

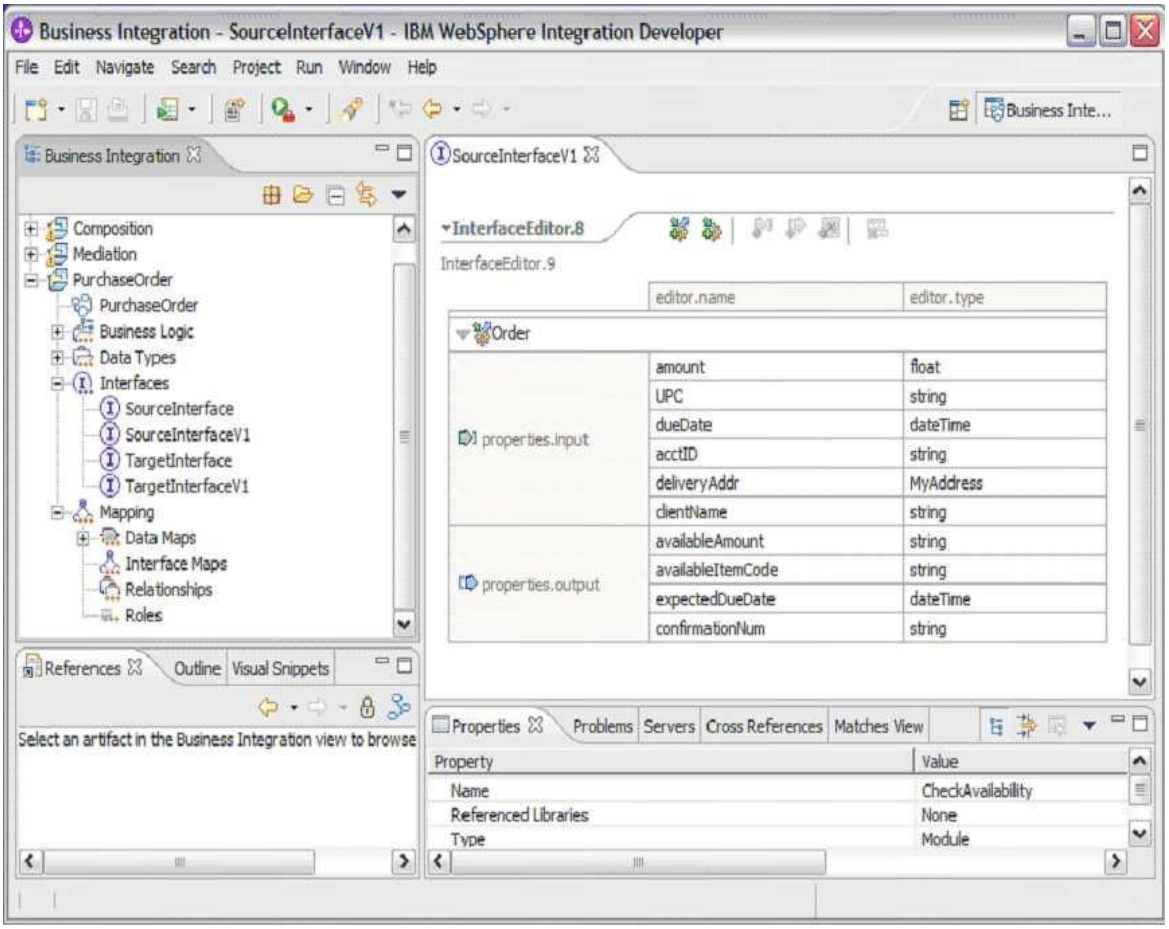


Abbildung 2-4: WebSphere Integration Developer

2.4 Business Process Execution Language (BPEL)

Die WS-Business Process Execution Language (BPEL) ist eine XML-basierte Sprache zur Beschreibung von Geschäftsprozessen, deren einzelne Aktivitäten durch Webservices implementiert sind. Durch die Abstraktion mittels BPEL kann die Schnittstelle eines Webservice, der die an einem Prozess beteiligten Webservices steuert, beschrieben werden – beispielsweise in welcher Reihenfolge Nachrichten eintreffen müssen. [7]

BPEL baut auf der Web Services Architektur auf. Die Web Services Description Language (WSDL) dient dazu einen anzubietenden WebService zu beschreiben, damit ein ServiceRequestor weiss, wie der Service genutzt werden kann. Wenn ein System einen WebService aufruft, wird eine XML-Datei mit WSDL-Deklarationen via SOAP, dem Übertragungsprotokoll, übertragen. In dieser Datei werden generell Operationen beschrieben die der Service zur Verfügung stellt, Protokolle und Datenformate über die er kommuniziert definiert und Adressangaben (URLs) festgehalten. Zudem können auch transaktionsspezifische Bestandteile enthalten sein. Auf die Transaktionssprache von WebServices WS-Transaction wird in dieser Arbeit nicht näher eingegangen, da dies den Rahmen dieser Arbeit sprengen würde. Speziell für BPEL werden in einer WSDL-Deklaration Message.

Das <message>-Tag definiert Nachrichten die zwischen der Prozessinstanz und den Partnern ausgetauscht werden. Eine Nachricht besitzt einen Namen und beinhaltet <part>-Tags. Mittels <part>-Tags und den darin definierten Attribute „name“ und „type“ wird die Datenstruktur eines Bestandteils der Nachricht definiert. Es gibt Eingabe- (Input-), Ausgabe- (Output-), und Fehler- (Fault-) Nachrichten. Letztere wird im Fehlerfalle ausgetauscht. Eine mögliche Eingabe- und eine Fehler-Nachricht unseres Beispielprozesses würde wie folgt aussehen:

```
<message name=“auftragsDaten“>
<part name=“kundenInformationen“ type=“sns:kundenInformationen“/>
<part name=“auftragsPositionen“ type=“sns:auftragsPositionen“/>
</message>
<message name=“auftragsDatenFehler“>
<part name=“problemInformation“ type=“xsd:string“/>
</message>
```

Unter einem PortType versteht man ein Element, das eine Reihe von abstrakten Operationen umfasst, die sich jeweils auf die definierten Ein- und Ausgabenachrichten beziehen. Für unseren Beispielprozess und mit Einbezug der obiger definierten Nachrichten (messages) würde eine PortType-Definition in einem WSDL-Dokument wie folgt aussehen:

```
<portType name="auftragsDatenPT">
<operation name="auftragsDatenSenden">
<input message ="auftragsDaten">
<output message ="rechnung">
<fault name ="auftragNichtKompletierbar"message="auftragsDatenFehler"/>
</operation>
</portType>
```

Das letzte der drei wesentlichen Elemente, die für einen BPEL-Process in WSDL definiert werden müssen, sind die PartnerLinkTypes. Ein PartnerLinkType representiert die Interaktion welche zwischen dem eigentlichen Prozess und den darauf zugreifenden Services (Partner; beteiligte Drittsysteme) stattfindet. Ein PartnerLinkType ist kein Standardelement von WSDL und wird durch Angabe einer Referenz im WSDL-Definitions-Header durch <definitions …xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"> benutzt. Die PartnerLinkTypes könnten auch in einem separaten Dokument definiert werden. Dazu ein Beispielcode:

```
<plnk:partnerLinkType name="einkaufenLT">
<plnk:role name="KäuferService">
<plnk:portType name="auftragsDatenPT"/>
</plnk:role>
<plnk:role name="VerkäuferService">
<plnk:portType name="auftragsDatenPT"/>
</plnk:role>
</plnk:partnerLinkType>
```

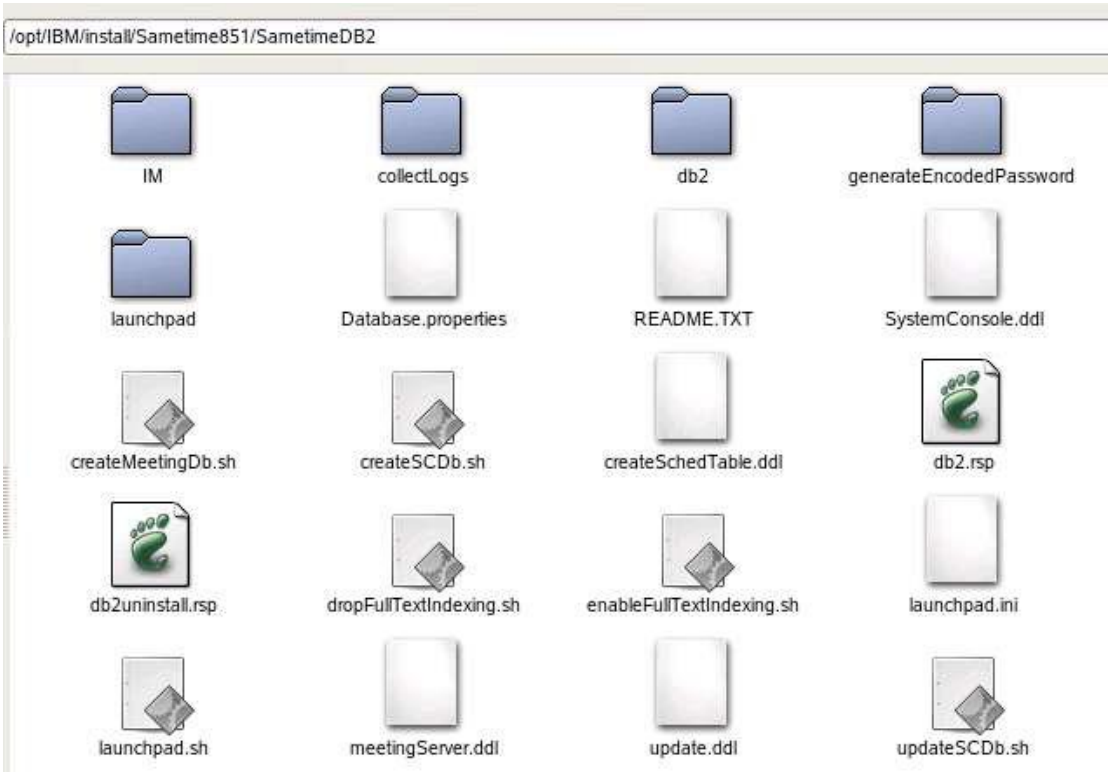
3 Instantisierung

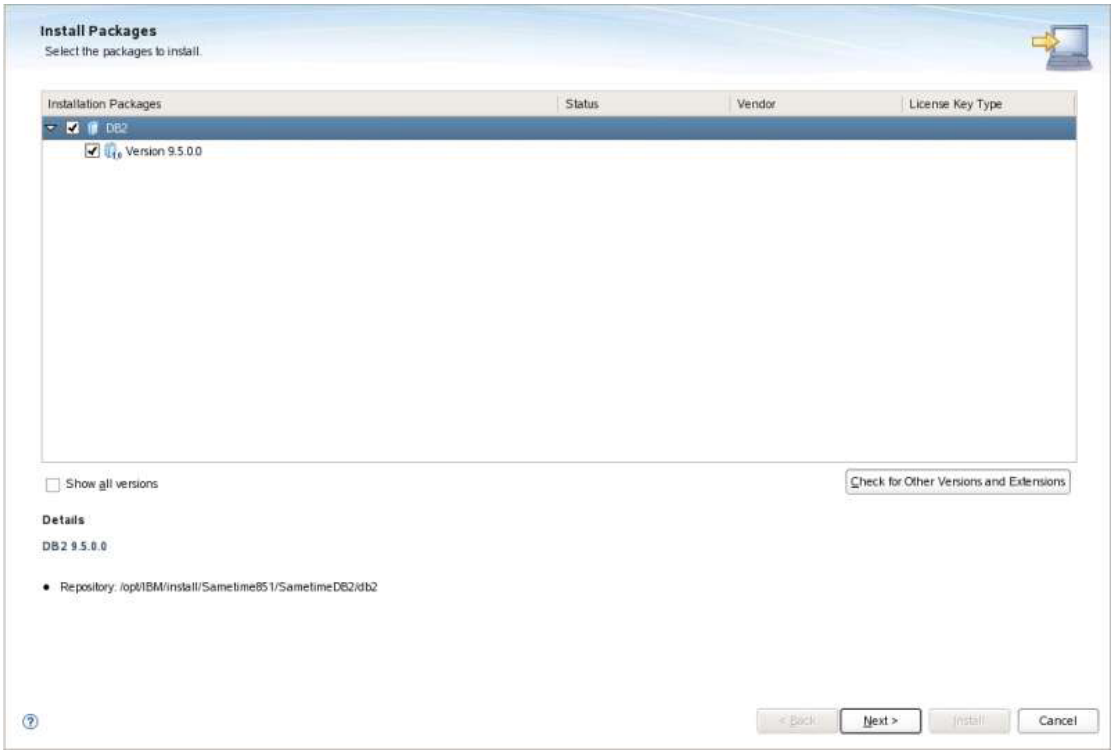
3.1 Instantisierung IBM DB2

Der DB2 Datenbankserver ist eine zentrale Komponente in der Sametime Installation. Er speichert alle Konfigurationen und Verbindungsinformationen der einzelnen Komponenten. Der DB2 Datenbankserver kann für High Availability und Load Balancing ausgelegt werden.

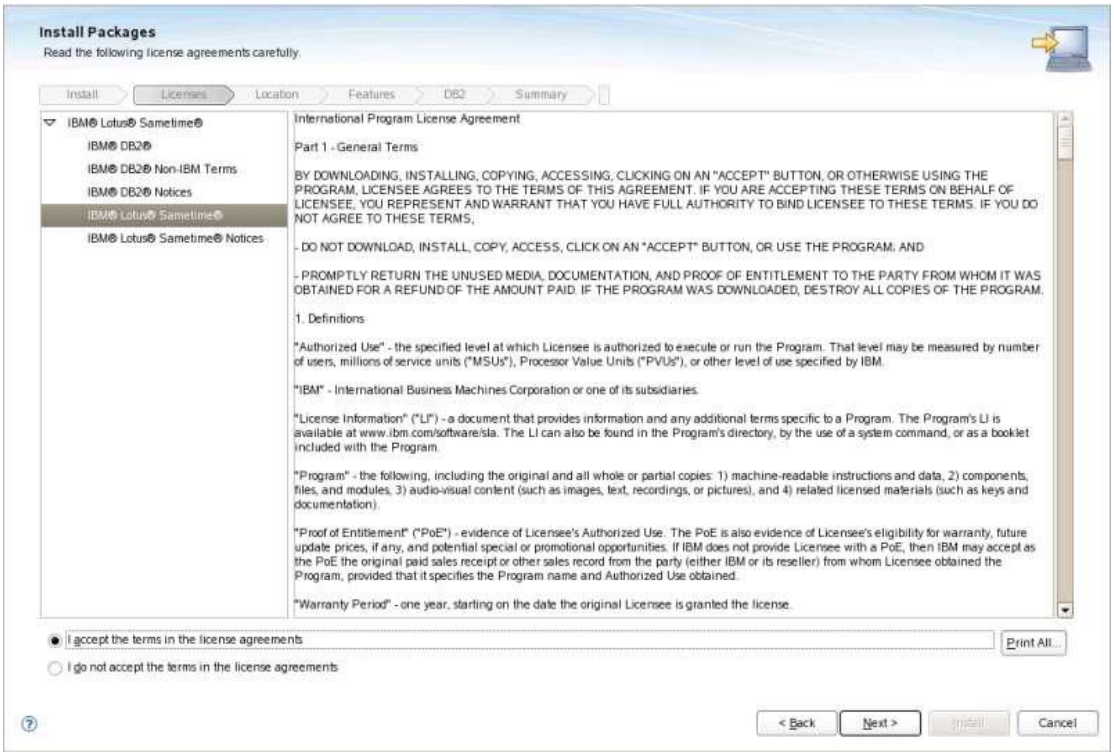
•Zum Starten der Installation ins Installationsverzeichnis wechseln

- „launchpad.sh“ mit Doppelklick starten
- launchpad.sh über die Shell starten

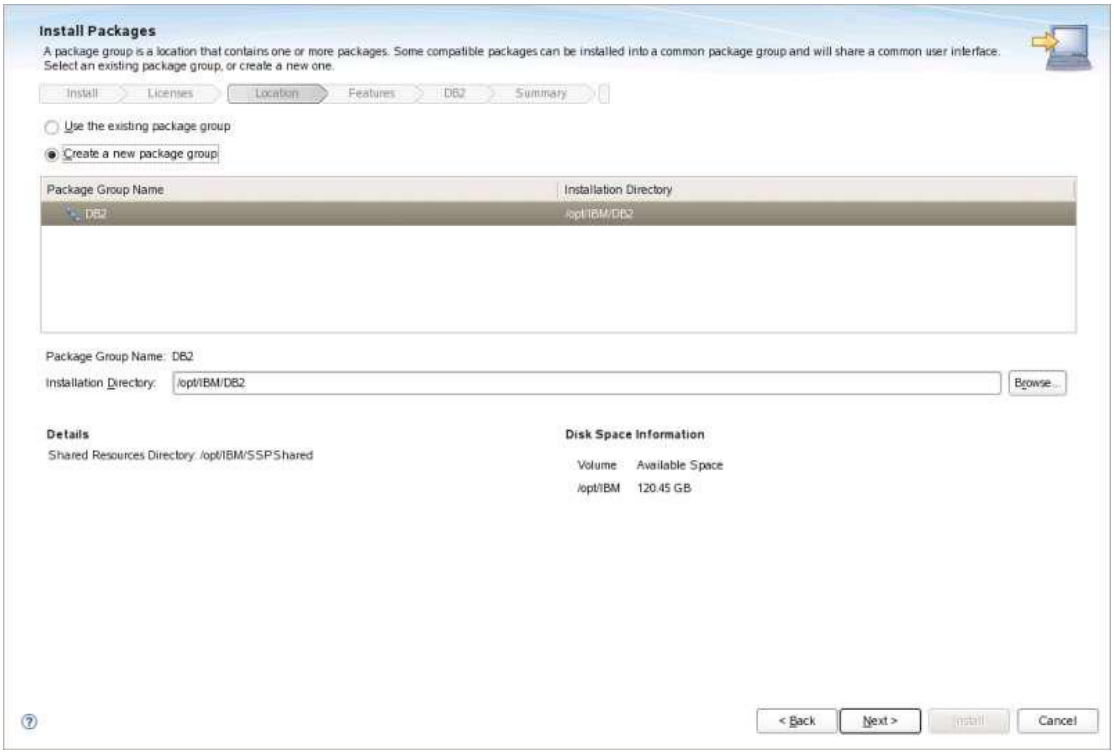




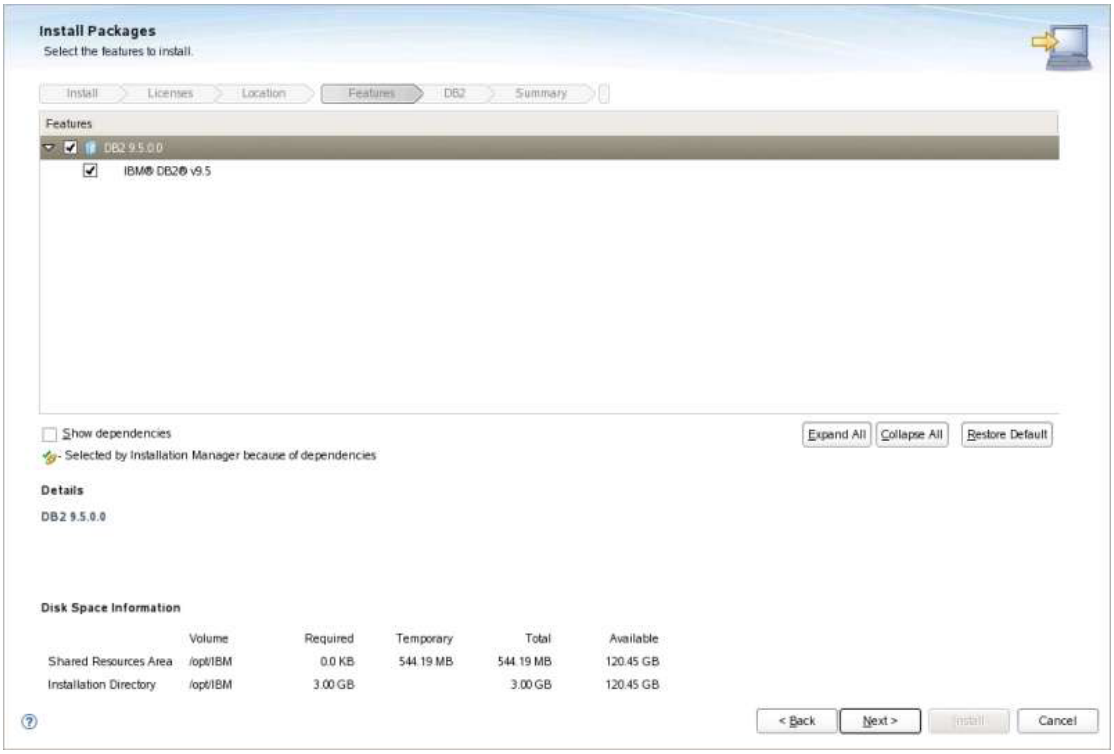
•Die Lizenzvereinbarungen akzeptieren und weiter mit „Next“



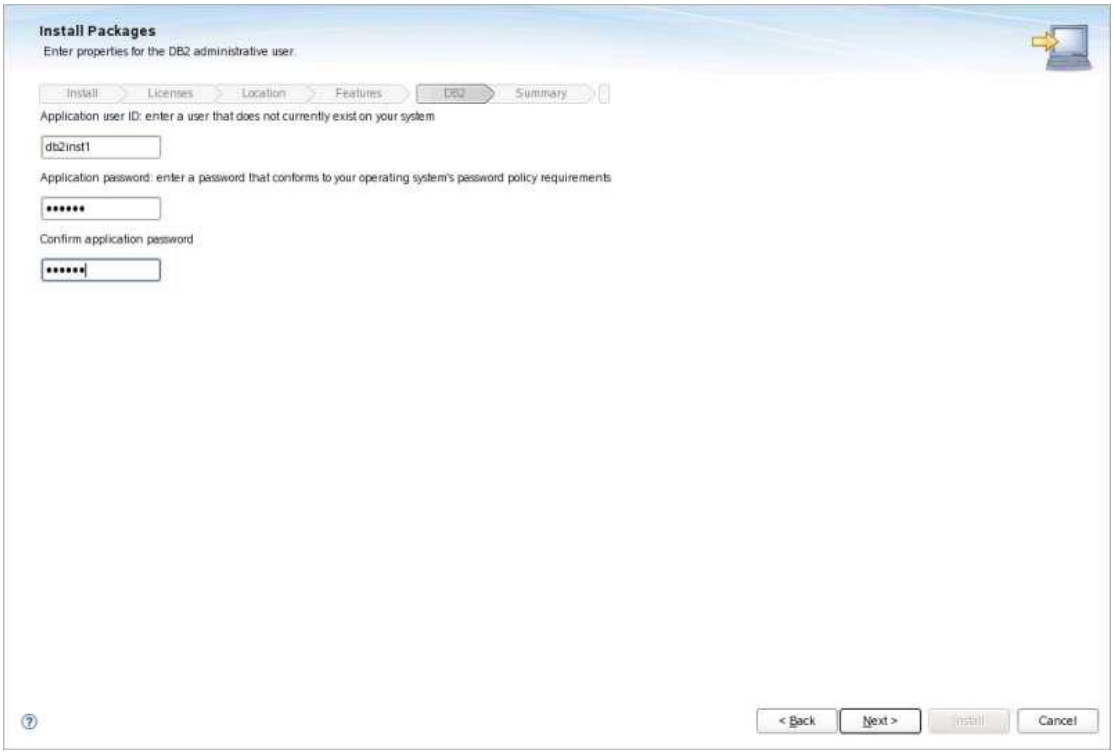
•Neue Paketgruppe erstellen und Installationspfad festlegen



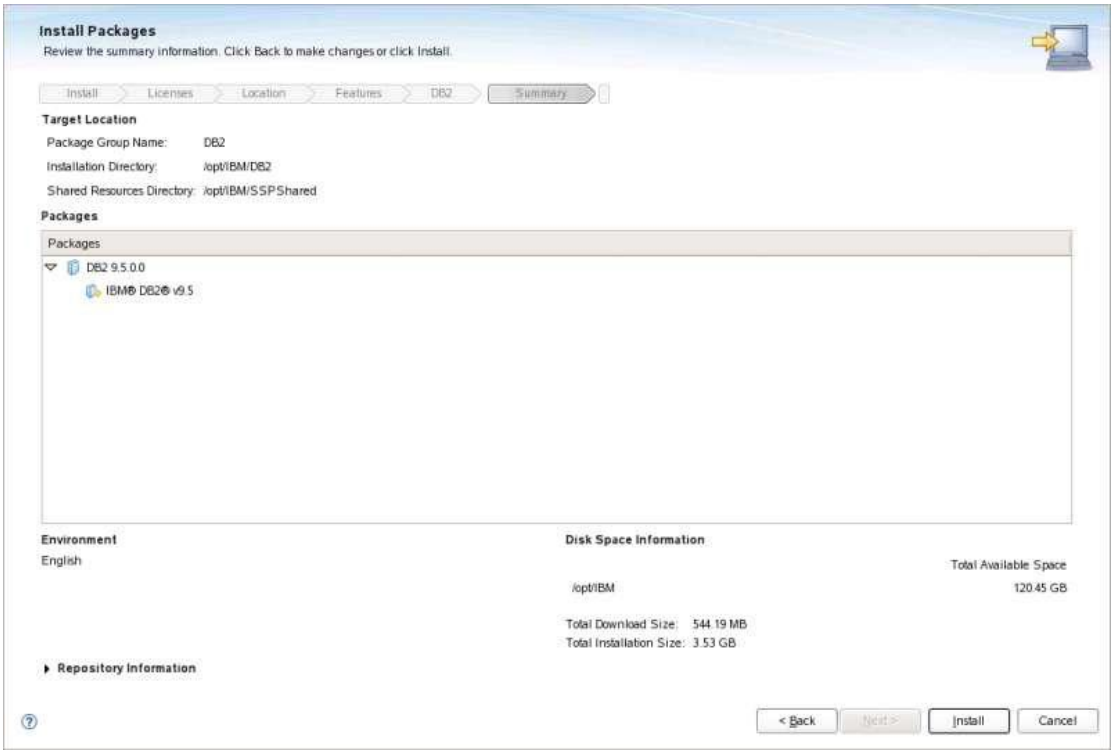
•Welche Pakete sollen installiert werden?



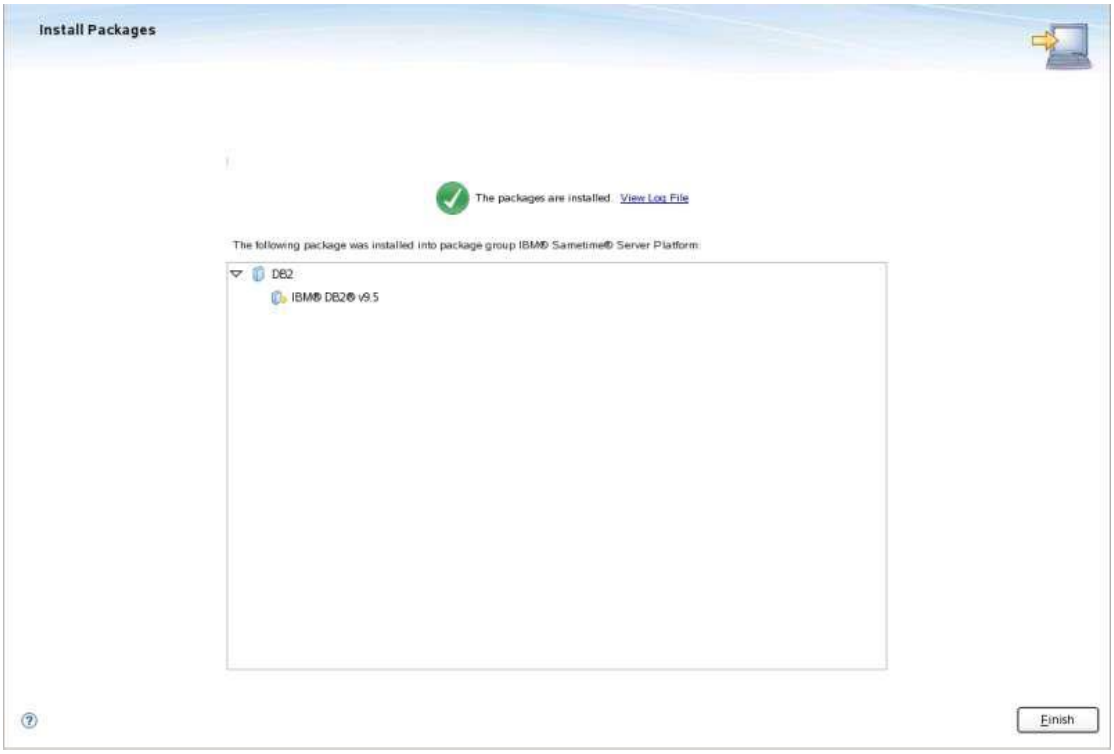
•Administrativen Account erstellen



•Installation starten



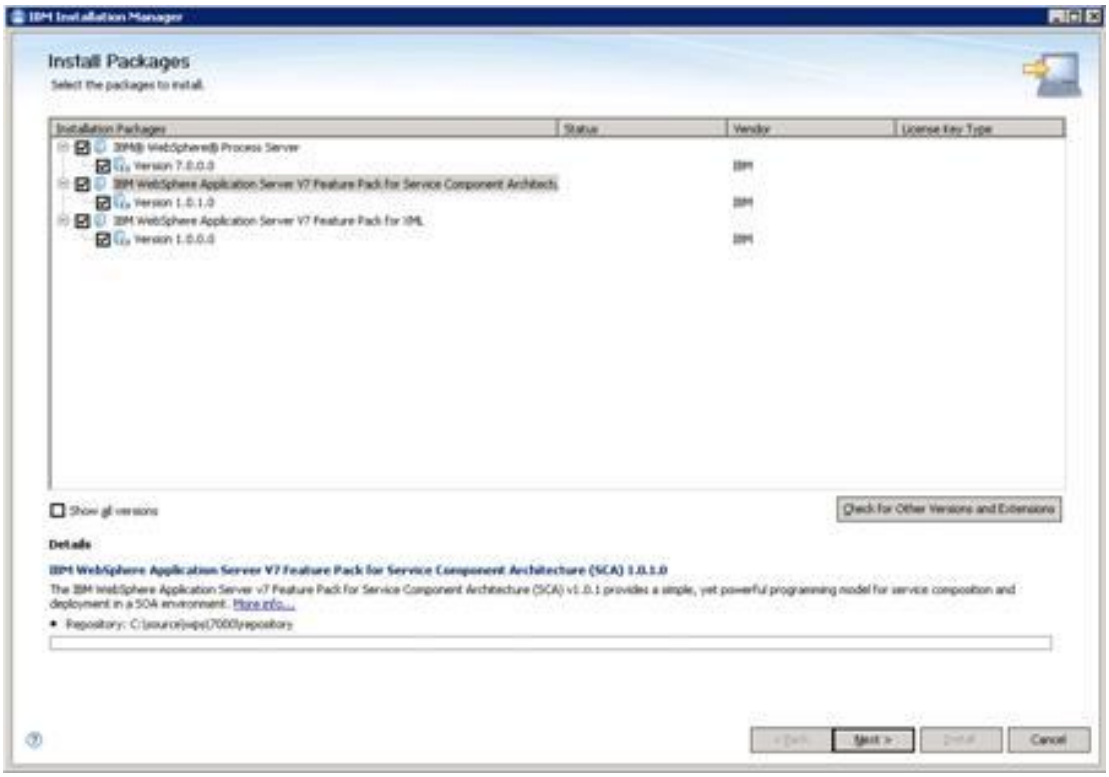
•Erfolgreicher Abschluss der Installation



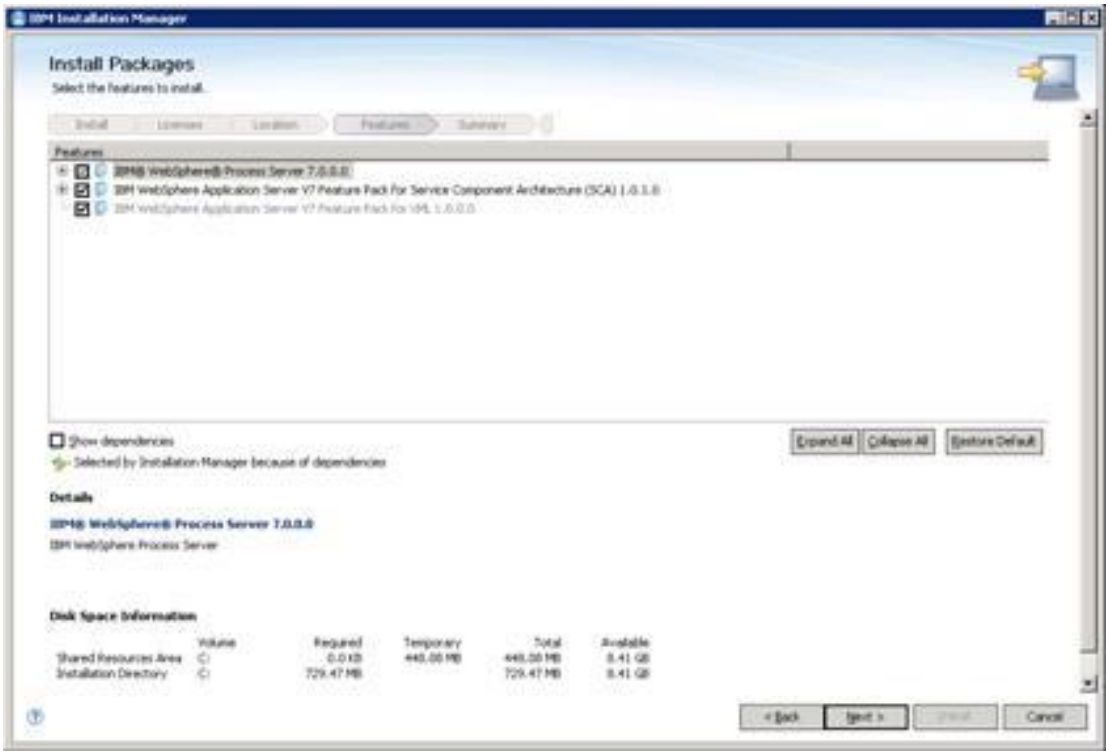
3.2 Instantisierung Websphere Process Server

•Entpacken Sie die .zip-Datei, starten Sie das Installationsprogramm 'launchpad.exe'

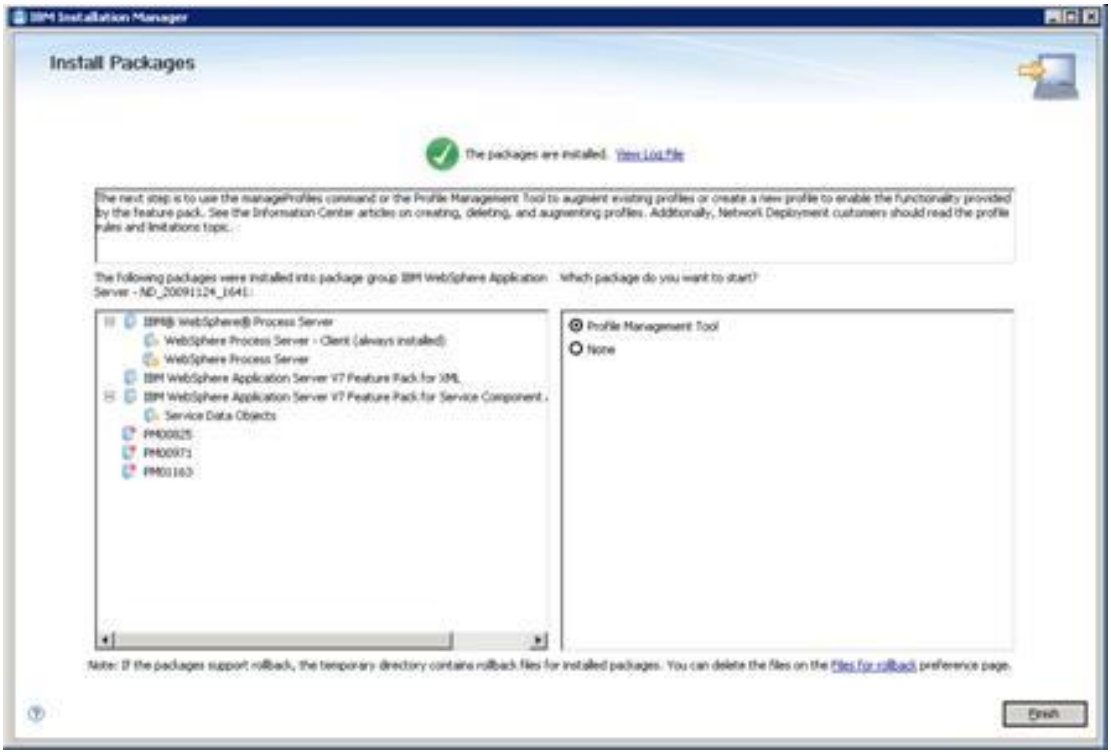
•Zur instalisierende Software auswählen



•Funktionen auswählen

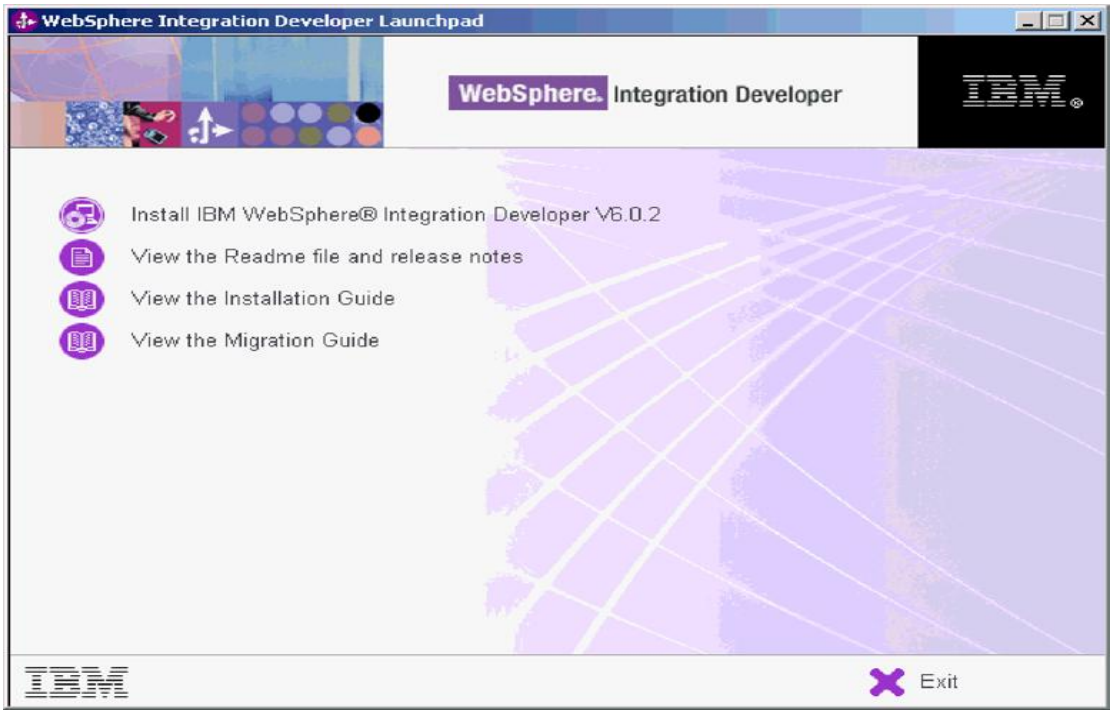


•Erfolgreicher Abschluss der Installation

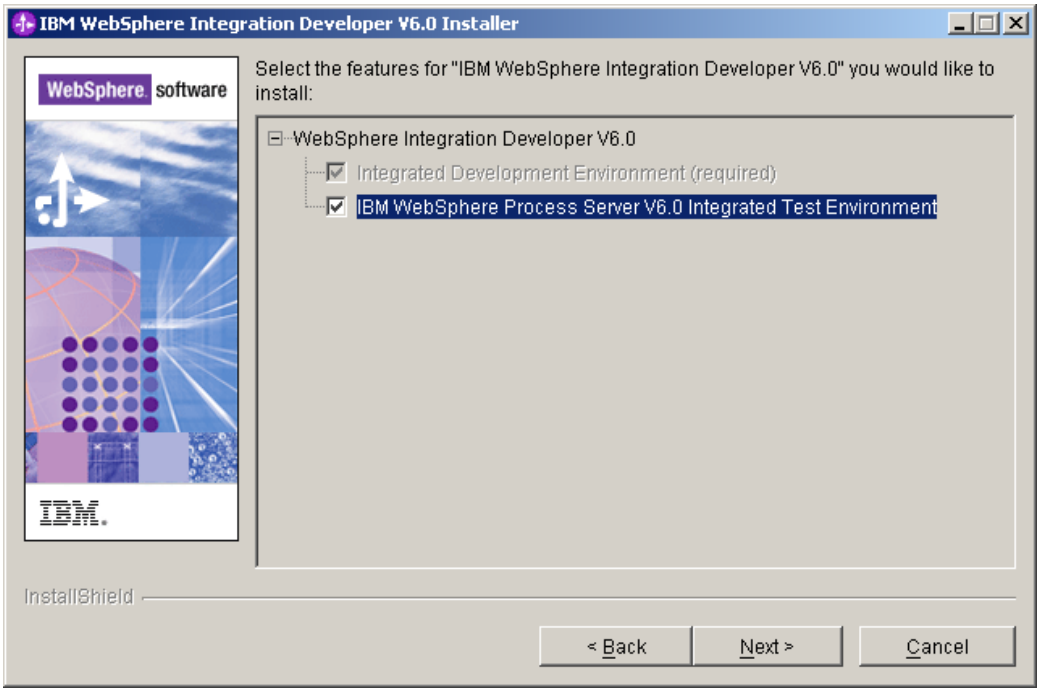


3.3 Instantisierung WebSphere Integration Developer

• Das Launchpad von WebSphere Integration Developer starten und “Install IBM-WebSphere Integration Developer” auswählen

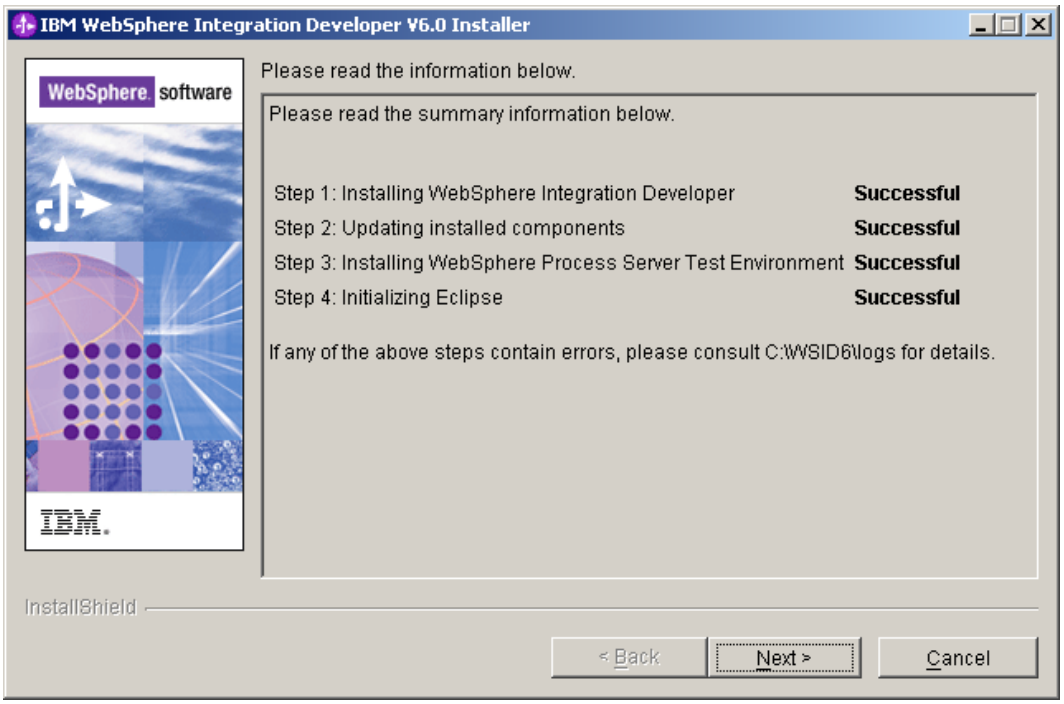


• “IBM WebSphere Process Server Integrated Test Environment” auswählen und “Next”



.

• Die Anweisungen im Installation befolgen und erfolgreicher Abschluss der Installation



4 Ausführung

In diesem Kapital werden die BPEL-Programme ausgeführt. Und das Process Model wird auch angezeiget. Zum Schluss werden die Ergebnisse analysiert.

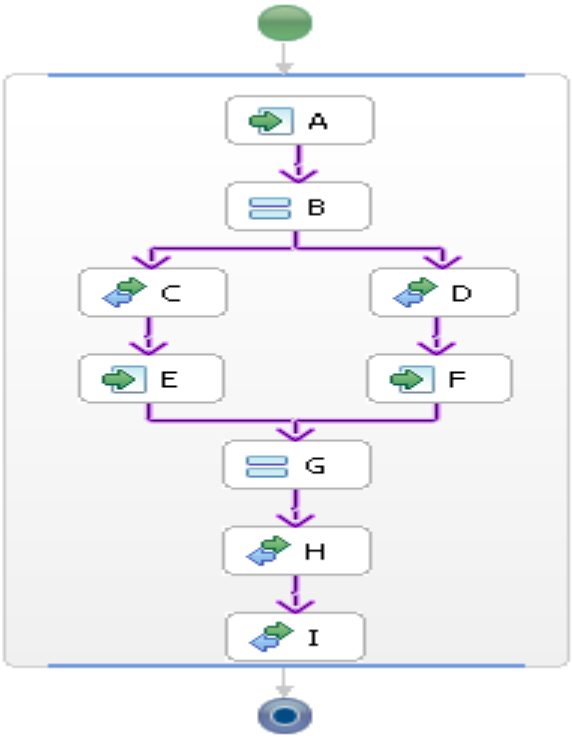
4.1 Umgebung bei der Implementierung

Das Testsystem wird wie folgt eingesetzt:

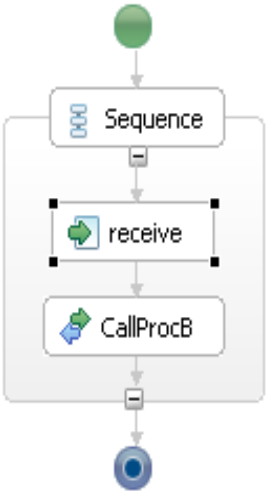
Hardwareumgebung:
Prozessor: Pentium® Dual-Core T4500 2.30GHz
Arbeitsspeicher: 3GB RAM
Festplatte: 250GB
Softwareumgebung:
Betriebssystem: Microsoft Windows XP Home Edition mit Service Pack 2

4.2 Process Model

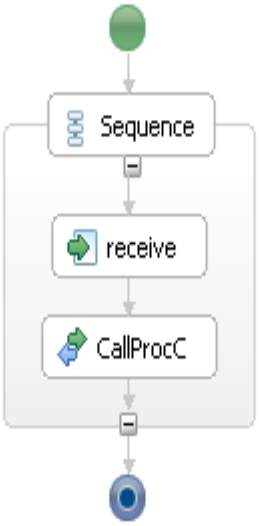
•Diagramm von BPEL Process TTProcA



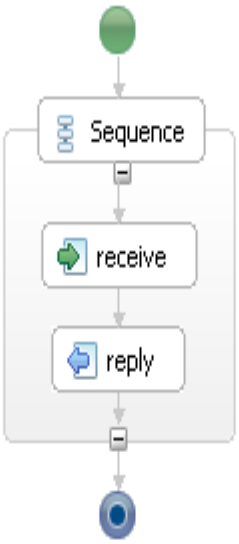
•Diagramm von BPEL Process TTProcB



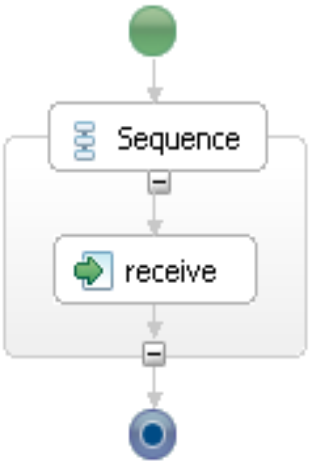
•Diagramm von BPEL Process TTProcC



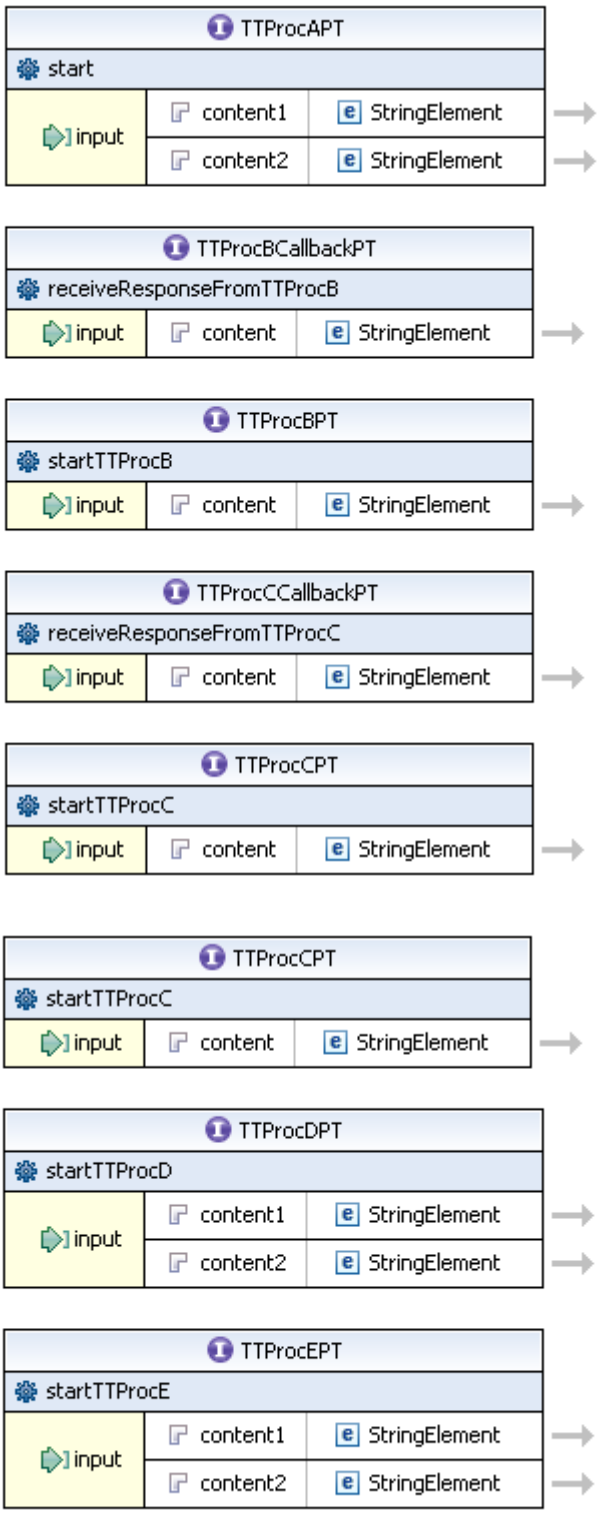
•Diagramm von BPEL Process TTProcD



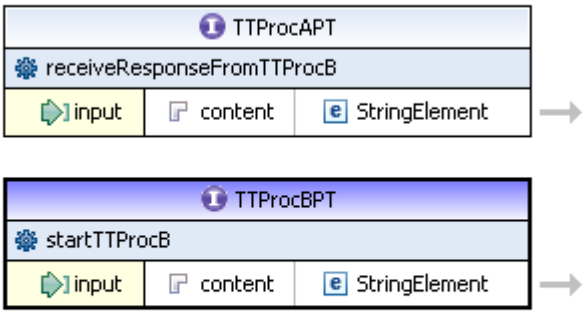
•Diagramm von BPEL Process TTProcE



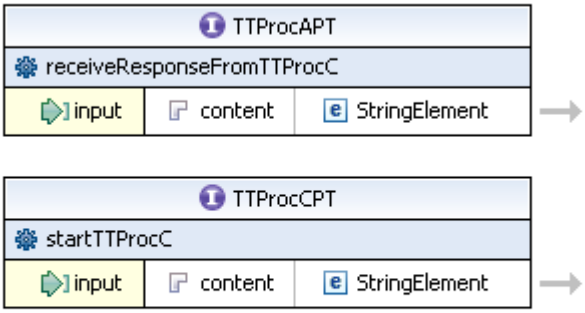
•Diagramm von WSDL TTProcA



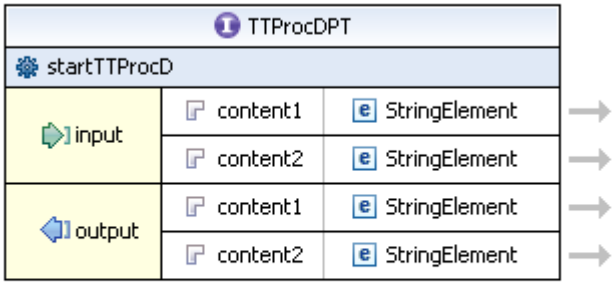
•Diagramm von WSDL TTProcB



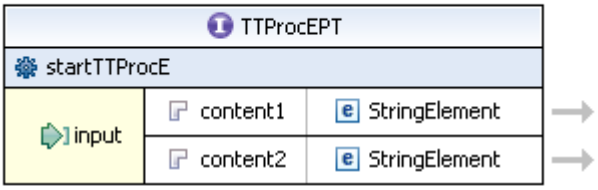
•Diagramm von WSDL TTProcC



•Diagramm von WSDL TTProcD



•Diagramm von WSDL TTProcE



4.3 Ergebnis-Analyse

Die erste Schritt zur Ausführung ist Datenbank zu starten. Danach braucht Server ca 4 mins zu starten. Wenn das Process Server bereit zu laufen ist, muss man Server Manger aktivieren. Ab jetzt kann das BPEL-Programm ausgeführt werden. Die Ausführungszeit für jeweilige Programm sieht folgende aus:

TTProcA: 12 sec TTProcB: 15 sec TTProcC: 15 sec
TTProcD: 22 sec TTProcE: 30 sec

5 BPEL-Processes

• BPEL Process TTProcA

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="TTProcA"
        expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
        suppressJoinFailure="yes"
        targetNamespace="http://rol.swomTest.org/TTProcA"
        xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
        xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
        xmlns:tns="http://rol.swomTest.org/wsdl/TTProcA/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <partnerLinks>
    <partnerLink name="TTProcAPL"
                 myRole="TTProcA"
                 partnerLinkType="tns:TTProcALT"/>
    <partnerLink name="TTProcBPL"
                 myRole="CallbackForTTProcB"
                 partnerRole="TTProcB"
                 partnerLinkType="tns:TTProcBLT"/>
    <partnerLink name="TTProcCPL"
                 myRole="CallbackForTTProcC"
                 partnerRole="TTProcC"
                 partnerLinkType="tns:TTProcCLT"/>
    <partnerLink name="TTProcDPL"
                 partnerRole="TTProcD"
                 partnerLinkType="tns:TTProcDLT"/>
    <partnerLink name="TTProcEPL"
                 partnerRole="TTProcE"
                 partnerLinkType="tns:TTProcELT"/>

  <variables>
    <variable messageType="tns:Message1" name="InRequest"/>
    <variable messageType="tns:Message" name="OutRequest1"/>
    <variable messageType="tns:Message" name="OutRequest2"/>
    <variable messageType="tns:Message" name="InResponse1"/>
    <variable messageType="tns:Message" name="InResponse2"/>
    <variable messageType="tns:Message1" name="InInvoke"/>
    <variable messageType="tns:Message1" name="OutInvoke"/>
  </variables>
```

```
<correlationSets>
   <correlationSet name="correlation1" properties="tns:correlationProperty"/>
   <correlationSet name="correlation2" properties="tns:correlationProperty"/>
</correlationSets>

<flow>

   <links>
      <link name="LinkAB"/>
      <link name="LinkBC"/>
      <link name="LinkBD"/>
      <link name="LinkCE"/>
      <link name="LinkDF"/>
      <link name="LinkEG"/>
      <link name="LinkFG"/>
      <link name="LinkGH"/>
      <link name="LinkHI"/>
   </links>

   <receive createInstance="yes"
               name="A"
               operation="start"
               partnerLink="TTProcAPL"
               portType="tns:TTProcAPT"
               variable="InRequest">
      <sources>
         <source linkName="LinkAB"/>
      </sources>
   </receive>

   <assign name="B">
      <targets>
         <target linkName="LinkAB"/>
      </targets>
      <sources>
         <source linkName="LinkBC"/>
         <source linkName="LinkBD"/>
      </sources>
      <copy>
         <from variable="InRequest" part="content1" />
         <to variable="OutRequest1" />
      </copy>
       <copy>
```

```
        <from variable="InRequest" part="content2" />
        <to variable="OutRequest2" />
    </copy>
</assign>

<invoke name="C"
            operation="startTTProcB"
            partnerLink="TTProcBPL"
            portType="tns:TTProcBPT"
            inputVariable="OutRequest1">
    <targets>
        <target linkName="LinkBC"/>
    </targets>
    <sources>
        <source linkName="LinkCE"/>
    </sources>
    <correlations>
        <correlation initiate ="yes" set="correlation1"/>
    </correlations>
</invoke>

<invoke name="D"
            operation="startTTProcC"
            partnerLink="TTProcCPL"
            portType="tns:TTProcCPT"
            inputVariable="OutRequest2">
    <targets>
        <target linkName="LinkBD"/>
    </targets>
    <sources>
        <source linkName="LinkDF"/>
    </sources>
    <correlations>
        <correlation initiate ="yes" set="correlation2"/>
    </correlations>
</invoke>

<receive name="E"
              operation="receiveResponseFromTTProcB"
              partnerLink="TTProcBPL"
              portType="tns:TTProcBCallbackPT"
              variable="InResponse1">
    <targets>
        <target linkName="LinkCE"/>
```

```
    </targets>
    <sources>
        <source linkName="LinkEG"/>
    </sources>
     <correlations>
          <correlation set="correlation1"/>
     </correlations>
</receive>

<receive name="F"
              operation="receiveResponseFromTTProcC"
              partnerLink="TTProcCPL"
              portType="tns:TTProcCCallbackPT"
              variable="InResponse2">
    <targets>
        <target linkName="LinkDF"/>
    </targets>
    <sources>
        <source linkName="LinkFG"/>
    </sources>
     <correlations>
          <correlation set="correlation2"/>
     </correlations>
</receive>

<assign name="G">
    <targets>
        <target linkName="LinkEG"/>
        <target linkName="LinkFG"/>
    </targets>
    <sources>
        <source linkName="LinkGH"/>
    </sources>
    <copy>
        <from variable="InResponse1" />
        <to variable="InInvoke" part="content1" />
    </copy>
    <copy>
        <from variable="InResponse2" />
        <to variable="InInvoke" part="content2" />
    </copy>
</assign>

 <invoke name="H"
```

```
                operation="startTTProcD"
                partnerLink="TTProcDPL"
                portType="tns:TTProcDPT"
                inputVariable="InInvoke"
                outputVariable="OutInvoke">
            <targets>
                <target linkName="LinkGH"/>
            </targets>
            <sources>
                <source linkName="LinkHI"/>
            </sources>
        </invoke>


        <invoke name="I"
                operation="startTTProcE"
                partnerLink="TTProcEPL"
                portType="tns:TTProcEPT"
                inputVariable="OutInvoke">
            <targets>
                <target linkName="LinkHI"/>
            </targets>
        </invoke>



    </flow>
</process>
</partnerLinks>
```

• BPEL Process TTProcB

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="TTProcB"
            expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
            suppressJoinFailure="yes"
            targetNamespace="http://rol.swomTest.org/TTProcB"
            xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
            xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
            xmlns:tns="http://rol.swomTest.org/wsdl/TTProcB/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <partnerLinks>
        <partnerLink name="TTProcBPL"
```

```
                    myRole="TTProcB"
                    partnerRole="CallbackForTTProcB"
                    partnerLinkType="tns:TTProcBLT"/>
  </partnerLinks>

  <variables>
    <variable messageType="tns:Message" name="Request"/>
  </variables>


  <sequence>

    <receive createInstance="yes"
                name="receive"
                operation="startTTProcB"
                partnerLink="TTProcBPL"
                portType="tns:TTProcBPT"
                variable="Request">

    </receive>

    <invoke name="CallProcB"
                operation="receiveResponseFromTTProcB"
                partnerLink="TTProcBPL"
                portType="tns:TTProcAPT"
                inputVariable="Request">

    </invoke>
  </sequence>
</process>
```

• BPEL Process TTProcC

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="TTProcC"
          expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
          suppressJoinFailure="yes"
          targetNamespace="http://rol.swomTest.org/ProcC"
          xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
          xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
          xmlns:tns="http://rol.swomTest.org/wsdl/TTProcC/"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
  <partnerLinks>
    <partnerLink name="TTProcCPL"
                 myRole="TTProcC"
                 partnerRole="CallbackForTTProcC"
                 partnerLinkType="tns:TTProcCLT"/>
  </partnerLinks>

  <variables>
    <variable messageType="tns:Message" name="Request"/>
  </variables>


  <sequence>

    <receive createInstance="yes"
             name="receive"
             operation="startTTProcC"
             partnerLink="TTProcCPL"
             portType="tns:TTProcCPT"
             variable="Request">

    </receive>

    <invoke name="CallProcC"
             operation="receiveResponseFromTTProcC"
             partnerLink="TTProcCPL"
             portType="tns:TTProcAPT"
             inputVariable="Request">

    </invoke>
  </sequence>
</process>
```

• BPEL Process TTProcD

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="TTProcD"
          expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
          suppressJoinFailure="yes"
          targetNamespace="http://rol.swomTest.org/TTProcD"
          xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
          xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
          xmlns:tns="http://rol.swomTest.org/wsdl/TTProcD/"
```

```
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <partnerLinks>
      <partnerLink name="TTProcDPL"
                        myRole="TTProcD"
                        partnerLinkType="tns:TTProcDLT"/>
   </partnerLinks>

   <variables>
      <variable messageType="tns:Message" name="Request"/>
   </variables>


   <sequence>

      <receive createInstance="yes"
                  name="receive"
                  operation="startTTProcD"
                  partnerLink="TTProcDPL"
                  portType="tns:TTProcDPT"
                  variable="Request">
      </receive>

      <reply name="reply"
                operation="startTTProcD"
                partnerLink="TTProcDPL"
                portType="tns:TTProcDPT"
                variable="Request">
      </reply>

   </sequence>
</process>
```

• BPEL Process TTProcE

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="TTProcE"
            expressionLanguage="http://www.w3.org/TR/1999/REC-xpath-19991116"
            suppressJoinFailure="yes"
            targetNamespace="http://rol.swomTest.org/TTProcE"
            xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
            xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
            xmlns:tns="http://rol.swomTest.org/wsdl/TTProcE/"
```

```
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <partnerLinks>
      <partnerLink name="TTProcEPL"
                        myRole="TTProcE"
                        partnerLinkType="tns:TTProcELT"/>
   </partnerLinks>

   <variables>
      <variable messageType="tns:Message" name="Request"/>
   </variables>


   <sequence>

      <receive createInstance="yes"
                  name="receive"
                  operation="startTTProcE"
                  partnerLink="TTProcEPL"
                  portType="tns:TTProcEPT"
                  variable="Request">
      </receive>

   </sequence>
</process>
```

Literaturverzeichnis:

[1] IBM WebSphere Process Server
http://en.wikipedia.org/wiki/IBM_WebSphere_Process_Server

[2] Gartner Group, “IBM Has Top Share in All Application,
Middleware Markets,”Joanne Correia, Yefim Natis, Massimo Pezzini
Roy Schulte, May 7, 2003.

[3] WebSphere V7.0 Redbook

[4] Wikipedia DB2 Wikipedia DB2
http://de.wikipedia.org/wiki/DB2

[5] IBM DB2
http://salz.is.informatik.uni-duisburg.de/db2doc/de_DE/index.html

[6] WebSphere Integration Developer
http://sauron.inf.mit.bme.hu/Edu/REMo/remo08.nsf/4c94cd99b663b3e7c12574c5005213c8/0094a29698053ef6c125751600357932/$FILE/WID_WPS_WBM_Introduction.pdf

[7] Business Process Execution Language (BPEL)
http://de.wikipedia.org/wiki/WS-Business_Process_Execution_Language

Abbildungsverzeichnis

Erklärung

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

Unterschrift: (Bo Cao)

Stuttgart, 31.12.2010