Entwicklung und Optimierung von Display-Schnittstellen fuer embedded Linux Boards

-

Masterarbeit

im Fachgebiet Hard- und Softwareentwicklung

vorgelegt von:	Armin Schlegel
Studienbereich:	Fakultaet EFI
Matrikelnummer:	2020863
Erstgutachter:	Prof. Dr. Joerg Arndt
Zweitgutachter:	Peter Meier

Zusammenfassung

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Quisque Natural-Programmierung blandit sed, hendrerit at, pharetra eget, dui. Sed lacus. Pellentesque malesuada. Cras gravida mi id sapien. Ut risus justo, fermentum non, scelerisque sit amet, lacinia in, erat. Proin nec lorem. Quisque porta, nisl at porta aliquam, felis libero consequat ipsum, vitae scelerisque dolor mi a odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis sollicitudin. Proin sollicitudin varius arcu. Morbi eleifend, metus sit amet placerat pharetra, dolor dui lobortis pede, vel imperdiet tellus eros imperdiet lorem. In hac habitasse platea dictumst. Curabitur elit mi, facilisis nec, ultricies id, aliquet et, magna. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam ac est. Mauris turpis enim, feugiat non, imperdiet congue, scelerisque non, purus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam dictum aliquet purus. Maecenas faucibus. Maecenas suscipit.

Abstract

Fusce neque est, tincidunt eu, nonummy nec, tempor iaculis, erat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum egestas, velit a rhoncus gravida, metus dolor pulvinar diam, sit amet placerat risus dolor sit amet elit. Maecenas eget purus ut est mattis porta. Suspendisse ut mi et mauris lobortis malesuada. Vestibulum dapibus. Duis hendrerit, elit eu venenatis eleifend, sapien ante volutpat odio, ac condimentum tellus massa ut massa. Etiam dapibus imperdiet metus. Sed sapien arcu, pulvinar quis, laoreet quis, venenatis non, justo. Aliquam est ante, pulvinar nec, accumsan sed, auctor sed, augue.

Ut adipiscing ligula. In mattis. Ut varius. In nec nulla at eros molestie viverra. Duis dolor risus, lobortis vel, dictum a, pellentesque id, lectus. Sed suscipit orci ac ligula venenatis condimentum. Maecenas et sem lacinia tortor cursus tempus. Mauris pellentesque risus at nulla. In arcu. Curabitur mattis mi quis dolor. In leo. Vivamus ut libero.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

1.1. Motivation

In der heutigen Zeit treten eingebettete Systeme (engl. embedded systems) immer stärker in den Vordergrund. Gerade in den Bereichen der Industrie, Telekommunikation oder Multimedia wächst der Bedarf an Lösungen die durch Zuverlässigkeit, Energiesparsamkeit und kompakter Bauform bestechen.
Obwohl eingebettete Systeme meist für den Anwender unsichtbar ihren Dienst verrichten, sind sie doch inzwischen allgegenw"rtig. Im Bereich der Telekomunikation und Unterhaltungselektronik kommt ein solches System im prinzip nicht mehr ohne ein Display aus. Die Moeglichkeit zur Anzeige multimedialer Daten wird zur Kaufentscheidung. Auch hier gilt die Maxime: besser, schneller, groesser.
Im Sektor der eingebetten Systeme mit Betriebssystem spielt Linux neben diversen anderen Systemen wie beispielsweise RTOS, OSEK, QNX oder auch Windows eine sehr grosse Rolle. In Verbindung zeigen eingebettete Linuxsysteme mit Displays ein grosses Potential. Mit der beliebten ARM-Architektur lassen sich so kostenguenstige, leisttungsstarke Systeme aufbauen, die die gestellten Aufgaben gut erfuellen kann.

1.2. Ziel der Arbeit

Das Ziel dieser Arbeit ist zu zeigen, dass die Verwendung von Displays mit eingebetten Linux Systemen je nach Anforderung einfach oder ueber Umwege realisierbar ist.

1.3. Aufbau der Arbeit

Im ersten Teil der Arbeit werden theoretische Grundlagen gebildet, die fuer das Verstaendnis notig sind. Hier werden Standards wie z.B. HDMI bzw. DVI, LVDS und RGB behandelt. Es wird ein Ueberblick ueber ausgewaehlte embedded Linux Boards gegeben und diese Klassifiziert mit welchen Displayschnittstellen diese ausgestattet sind bzw. ausgestattet werden koennen. Der Zweite Teil behandelt das embedded

Linux Board 'Gnublin', welches von Haus aus keine Displayschnittstelle vorgesehen hat. Hier werden zwei Varianten zur Ansteuerung von Displays erarbeitet. Die Ansteuerung wird hierbei vom Prozessor erledigt, da das 'Gnublin' keine dedizierten Grafikcontroller besitzt. Im dritten Teil wird fuer leistungsstaerkere embedded Linux-Systeme mit HDMI-Schnittstelle eine Hardware entwickelt, RGB- oder LVDS-Panels anzuschliessen. Um die Displys ueber die entwickelte Hardware anzusteuern, wird der dedizierte Grafikcontroller der Boards verwendet.

1.4. Typographische Konventionen

2. Zitate und Referenzen

Die **Service-orientierte Architektur** (SOA) ist seit einiger Zeit *das* Schlagwort im Bereich der Informationstechnologie. So haben z. B. Deutschlands größte Softwarehersteller SAP und die Software AG ihre Unternehmensstrategie komplett auf die SOA ausgerichtet. **?** bietet mit *Netweaver* seine marktführende ERP-Software auf Basis von SOA an,[1] und die **?**, die sich selbst als „The XML Company“ bezeichnet, erweiterte kürzlich noch einmal ihr bereits durchgängig an der SOA orientiertes Produktportfolio durch den Kauf des amerikanischen Unternehmens webMethods um Lösungen zur Unterstützung von Geschäftsprozessen.[2] In einem Atemzug mit der SOA werden häufig Webservices genannt, da sie durch ihre hohe Plattformunabhängigkeit und den Einsatz von Internettechnologie oftmals als Referenzimplementierung für die Services in einer SOA angeführt werden. Doch welche Vorteile bietet der Einsatz von Webservices in Unternehmen? Können mit ihnen tatsächlich flexiblere Softwaresysteme entwickelt werden? Und wie einfach ist die Implementierung von Webservices auf unterschiedlichen Plattformen? Diesen Fragen wird sich der Autor in der vorliegenden Arbeit widmen.

Wie bereits in Kapitel 1 auf Seite 1 erwähnt, ist zur Unterstützung von Geschäftsprozessen der Einsatz von Informationstechnologie notwendig. Der Autor verfolgt mit dieser Arbeit das Ziel, einen Geschäftsprozess mit Hilfe von Webservices zu optimieren. Hierzu wird er in diesem Kapitel eine Einführung in das Thema Webservices und die damit in Zusammenhang stehenden Technologien geben, und auch auf mögliche Einsatzbereiche von Webservices im Rahmen der Geschäftsprozessoptimierung eingehen. Tabelle 4.1 auf Seite 11 zeigt ganz tolle Sachen.

Was ist ein Geschäftsprozess?

Ich empfehle allen Softwareentwicklern die Lektüre von [**?**].

2.1. Definitionen

Die Service-orientierte Architektur ist ein Ansatz der Softwareentwicklung, der sich stark am Konzept der Geschäftsprozesse orientiert und mit Hilfe von Webservices implementiert werden kann. In den beiden folgenden Kapiteln werden beide Begriffe

[1] Vgl. [**?**, S. 127]
[2] Vgl. [**?**]

eingehend erläutert, worauf in Kapitel 5 die für die Umsetzung von Webservices benötigten Technologien vorgestellt werden.

2.2. Service-orientierte Architektur

?[3] definiert den Begriff **Service-orientierte Architektur** (SOA) wie folgt:

> „**Service Oriented Architecture** [...] is a paradigm for organizing and utilizing distributed **capabilities** that may be under the control of different ownership domains.“[4]

Diese bewusst allgemein gehaltene Definition stammt aus dem Referenzmodell der SOA aus dem Jahr 2006. Dieses Modell wurde mit dem Ziel entwickelt, ein einheitliches Verständnis des Begriffs SOA und des verwendeten Vokabulars zu schaffen, und sollte die zahlreichen bis dato vorhandenen, teils widersprüchlichen Definitionen ablösen.[5] Dabei wird zunächst noch kein Bezug zur Informationstechnologie hergestellt, sondern allgemein von Fähigkeiten gesprochen, die Personen, Unternehmen, aber eben auch Computer besitzen und evtl. Anderen anbieten, um Probleme zu lösen. Als Beispiel wird ein Energieversorger angeführt, der Haushalten seine Fähigkeit Strom zu erzeugen anbietet.[6]

[3] Die **Organization for the Advancement of Structured Information Standards** ist nach [**?**] ein internationales Konsortium aus über 600 Organisationen, das sich der Entwicklung von E-Business-Standards verschrieben hat. Mitglieder sind z. B. IBM, SAP und Sun.
[4] [**?**, S. 8]
[5] Vgl. [**?**, S. 4]
[6] Vgl. [**?**, S. 8f.]

3. Bilder und Listings

In den folgenden drei Kapiteln wird der Autor eine einfache Webservice-Umgebung aufbauen, um zu zeigen, wie Webservices in der Praxis angeboten, konsumiert und orchestriert werden können. Hierzu verwendet er ausschließlich Open-Source-Software, im Speziellen **Apache Tomcat**[7] als Servlet-Engine, **Apache Axis2**[8] als SOAP-Engine und **ActiveBPEL**[9] als Workflow-System. Die Installation und Konfiguration der benötigten Anwendungen wird in Kapitel 4.1 beschrieben. Die komplette Umgebung inkl. der vom Autor erstellten Webservices befindet sich als virtuelle Maschine auf der dieser Arbeit beigelegten DVD. Im Folgenden wird der DNS-Name `linux-ws` als Bezeichnung für den Webservice-Server verwendet.

3.1. Anbieten eines Webservice

Mit Hilfe von Apache Axis2 können Webservices sehr einfach auf Basis von normalen Java-Klassen angeboten werden. Es ist lediglich eine zusätzliche XML-Datei namens `META-INF/services.xml` nötig, in der die zu veröffentlichenden Klassen und Methoden beschrieben werden. Abbildung 3.1 zeigt die Struktur eines einfachen HelloWorld-Webservice.

Abbildung 3.1.: HelloWorld-Webservice: Dateistruktur

Die Klasse `HelloWorld` besitzt nur die Methode `SayHello`, die den String `Hello World!` zurückgibt. Sie wird in Listing 3.1 gezeigt.

[7] Website: http://tomcat.apache.org/
[8] Website: http://ws.apache.org/axis2/
[9] Website: http://www.activebpel.org/

Listing 3.1: HelloWorld-Webservice: Java-Klasse `HelloWorld`

```
1 public class HelloWorld {
2   public String SayHello() {
3     return "Hello World!";
4   }
5 }
```

3.2. Netzwerkverkehr beim Aufruf von PersonFactory

3.2.1. SOAP-Request

Listing 3.2 zeigt die mitgeschnittene SOAP-Anfrage per HTTP an den Webservice PersonFactory. Wie am Ende von Kapitel 1 beschrieben, wird die eigentliche SOAP-Nachricht mittels des HTTP-`POST`-Befehls (Zeile 1) an den Webservice unter der angegebenen URL (Zeile 1) auf dem Server (Zeile 5) geschickt. In Zeile 3 wird über den Befehl `SOAPAction` übermittelt, welche Funktion des Webservice (in diesem Fall `CreatePerson`) aufgerufen werden soll. Die XML-Nutzlast (Zeilen 8–18) besteht dann aus einer einfachen SOAP-Nachricht aus Envelope, Header und Body, die einen RPC durchführt. Die aufzurufenden Funktion wird noch einmal im SOAP-Body in Zeile 15 definiert.

Listing 3.2: SOAP-Request an PersonFactory per HTTP

```
1 POST /axis2/services/PersonFactory HTTP/1.1
2 Content-Type: text/xml;charset=UTF-8
3 SOAPAction: "urn:PersonFactory#CreatePerson"
4 User-Agent: Jakarta Commons-HttpClient/3.0.1
5 Host: linux-ws:8080
6 Content-Length: 380
7
8 <soapenv:Envelope
9   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
11  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
12  xmlns:urn="urn:PersonFactory">
13  <soapenv:Header/>
14  <soapenv:Body>
15    <urn:CreatePerson soapenv:encodingStyle=
16      "http://schemas.xmlsoap.org/soap/encoding/" />
17  </soapenv:Body>
18 </soapenv:Envelope>
```

3.2.2. SOAP-Response

Die Antwort des PersonFactory-Webservice zeigt Listing 3.3. Sie beginnt in Zeile 1 mit dem HTTP-Statuscode 200, der die Anfrage als erfolgreich kennzeichnet. Die eigentliche Nutzlast in Form von XML-Daten (Zeile 3) folgt dann ab Zeile 7. Sie besteht aus dem Element Person und seinen Unterelementen, umschlossen vom Element CreatePersonRepsonse, das die Antwort-Nachricht aus der WSDL repräsentiert.

Listing 3.3: SOAP-Response von PersonFactory per HTTP

```
1  HTTP/1.1 200 OK
2  Server: Apache-Coyote/1.1
3  Content-Type: text/xml;charset=UTF-8
4  Transfer-Encoding: chunked
5  Date: Wed, 13 Jun 2007 11:14:38 GMT
6
7  <?xml version='1.0' encoding='UTF-8'?>
8  <soapenv:Envelope
9    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
10   <soapenv:Body>
11     <pf:CreatePersonResponse
12       xmlns:pf="http://xml.stefanmacke.com/PersonFactory">
13       <person:Person xmlns:person="http://xml.stefanmacke.com/Person">
14         <person:Name>Stefan Macke</person:Name>
15         <person:Age>25</person:Age>
16       </person:Person>
17     </pf:CreatePersonResponse>
18   </soapenv:Body>
19 </soapenv:Envelope>
```

4. Aufzählungen und Tabellen

Eine normale Punktliste:

- Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla ac ipsum a metus viverra tempor.
- Nunc sem. Nulla nec urna eu nibh vehicula convallis. Integer ac turpis. Donec mauris enim, dignissim quis, scelerisque ac, rhoncus id, sapien.
- Donec turpis felis, cursus in, varius vitae, mollis ac, lorem. Integer a dui sit amet eros nonummy aliquet. Donec egestas adipiscing tellus. Nulla iaculis.
- Aliquam erat volutpat. Curabitur posuere, eros vitae accumsan semper, risus erat viverra erat, eu vehicula mi leo at elit. Fusce luctus. Fusce vehicula pretium diam. Nunc sed arcu ut erat suscipit fermentum.

Eine nummerierte Liste:

1. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla ac ipsum a metus viverra tempor.
2. Nunc sem. Nulla nec urna eu nibh vehicula convallis. Integer ac turpis. Donec mauris enim, dignissim quis, scelerisque ac, rhoncus id, sapien.
3. Donec turpis felis, cursus in, varius vitae, mollis ac, lorem. Integer a dui sit amet eros nonummy aliquet. Donec egestas adipiscing tellus. Nulla iaculis.
4. Aliquam erat volutpat. Curabitur posuere, eros vitae accumsan semper, risus erat viverra erat, eu vehicula mi leo at elit. Fusce luctus. Fusce vehicula pretium diam. Nunc sed arcu ut erat suscipit fermentum.

4.1. Vom Autor verwendete Software

Im Folgenden werden die Programme vorgestellt, die der Autor zum Erstellen dieser Arbeit und vor allem zur Entwicklung der Webservices verwendet hat. Soweit es möglich war, wurden Open-Source-Programme eingesetzt.

- **Microsoft Visio**
  Die EPKs der BAP wurden mit Microsoft Visio erstellt. Der Autor hat zwar verschiedene Open-Source-Programme[10] ausprobiert, mit denen EPKs erstellt werden könnten, die grafischen Ergebnisse waren aber nicht zufriedenstellend. Die Symbole von Visio sehen den „originalen“ ARIS-Symbolen am ähnlichsten und können darüber hinaus mit zusätzlichen Informationen wie Dauer und Kosten versehen werden.

- **PSPad**
  Für die Bearbeitung von verschiedenen (Text-)Dateien wurde der Texteditor PSPad verwendet. Mit diesem konnten z. B. auch die regulären Ausdrücke für die XML-Schemas entwickelt werden. Website: http://www.pspad.com/

- **Eclipse**
  Sowohl der ActiveBPEL Designer als auch die EntireX Workbench sind Plug-ins für die IDE Eclipse. Auch zur Java- und PHP-Entwicklung wurde dieses Werkzeug verwendet. Website: http://www.eclipse.org/

- **XML Copy Editor**
  Für die Entwicklung der XML-Schemas und die Bearbeitung von XML-Dateien wurde der XML Copy Editor eingesetzt. Mit diesem können u. a. XML-Dateien auf Wohlgeformtheit geprüft und gegen ihr Schema validiert werden. Website: http://xml-copy-editor.sourceforge.net/

- **soapUI**
  Mit soapUI können Webservices getestet werden, ohne einen Client zu programmieren. Die SOAP-Anfragen werden automatisch anhand der WSDL generiert und die Antworten können gegen die WSDL-Datei validiert werden. Website: http://www.soapui.org/

- **Ethereal**
  Die Netzwerkkommunikation beim Aufrufen der Webservices wurde mit Ethereal, einem umfangreichen Werkzeug zur Analyse des Netzwerkverkehrs, mitgeschnitten. Website: http://www.ethereal.com/

- **LaTeX**
  Diese Arbeit wurde mit LaTeX geschrieben. Als Distribution wurde MiKTeX verwendet und als Editor der LaTeX Editor. Websites: http://miktex.org/, http://www.latexeditor.org/

[10] Dia, OpenOffice Draw und die EPC Tools.

4.2. Elemente der Ereignisgesteuerten Prozesskette

Tabelle 4.1.: Elemente der Ereignisgesteuerten Prozesskette

Element	Symbol
Funktion Funktionen beschreiben Tätigkeiten, die im Verlauf des Geschäftsprozesses anfallen. Sie können von Mitarbeitern oder einem Informationssystem durchgeführt werden und benötigen evtl. Ressourcen, die ihnen zugewiesen werden. Beispiele: *Auftrag anlegen*, *Rechnung schreiben*, *Konto abschließen*	
Ereignis Ereignisse sind betriebswirtschaftlich relevante Ereignisse, die den Geschäftsprozess in irgendeiner Weise steuern oder beeinflussen. Ereignisse sind immer Auslöser oder Ergebnisse von Funktionen. Ein Geschäftsprozess beginnt und endet stets mit einem Ereignis. Beispiele: *Auftrag eingetroffen*, *Überweisung getätigt*, *Rechnung erstellt*	
Operatoren Operatoren steuern den Kontrollfluss eines Geschäftsprozesses. Sie machen z. B. deutlich, dass eine Funktion mehrere Ereignisse auslöst, oder zeigen alternative Vorgehensweisen an. Es gibt drei Operatoren (v. l. n. r.): UND, ODER und XODER (exklusives ODER).	
Organisationseinheit Organisationseinheiten werden Funktionen zugeordnet und beschreiben, wo die Funktionen ausgeführt werden bzw. wer sie ausführt. Die Bezeichnung der Symbole enthält zusätzlich zur Abteilung noch die Namen der Mitarbeiter. Beispiele: *Vertrieb*, *Personal*, *Produktion*	
Informationsobjekt Auch Informationsobjekte werden Funktionen zugewiesen und beschreiben die von diesen benötigten oder erstellten Informationen. Dabei sind sämtliche Formen von Informationen auf verschiedenen Datenträgern möglich und nicht etwa nur digitale Daten. Die Bezeichnung der Symbole enthält zusätzlich das Informationssystem, aus dem die Informationen stammen. Beispiele: *Kundendatenbank*, *Versicherungsantrag*, *Rechnung*	

Prozesswegweiser Mit Prozesswegweisern werden Prozesse, die in anderen EPKs beschrieben sind, referenziert. So können z. B. unübersichtliche Prozesse in Teilprozesse gegliedert und häufig verwendete Prozesse an zentraler Stelle modelliert werden. Prozesswegweiser stehen in einer EPK immer anstelle von Funktionen.	

5. Fazit und kritische Bewertung

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla ac ipsum a metus viverra tempor. Nunc sem. Nulla nec urna eu nibh vehicula convallis. Integer ac turpis. Donec mauris enim, dignissim quis, scelerisque ac, rhoncus id, sapien. Donec turpis felis, cursus in, varius vitae, mollis ac, lorem. Integer a dui sit amet eros nonummy aliquet. Donec egestas adipiscing tellus. Nulla iaculis. Aliquam erat volutpat. Curabitur posuere, eros vitae accumsan semper, risus erat viverra erat, eu vehicula mi leo at elit. Fusce luctus. Fusce vehicula pretium diam. Nunc sed arcu ut erat suscipit fermentum.

Proin id magna eu sem tincidunt feugiat. Sed tincidunt massa sed eros. Fusce condimentum eros et lectus. Pellentesque lectus tortor, mattis in, dapibus a, lobortis ut, justo. Sed id dolor ut nibh varius ultrices. Quisque tincidunt nisl vel nibh. Suspendisse sodales massa non magna. In porttitor augue nonummy nunc. Nam quis enim quis ante dapibus interdum. Morbi nec neque. Fusce pharetra consectetuer magna. Etiam laoreet, augue nec lacinia ornare, risus purus lobortis erat, eu consequat urna orci vel arcu. Integer cursus, augue sed tempor dapibus, erat tortor rutrum elit, sit amet fermentum purus neque vitae tortor. Donec vulputate, ipsum vel viverra pretium, purus orci mattis nulla, nec tincidunt leo metus sed ipsum. Fusce eget lectus sed lectus molestie tincidunt. Etiam tincidunt urna eget tortor.

Sed sit amet magna at lectus interdum blandit. Proin vitae metus eget leo bibendum ornare. Morbi sit amet nisl ac odio accumsan laoreet. Etiam luctus massa vel enim. Vestibulum nulla tellus, viverra at, malesuada vel, volutpat quis, lorem. Vestibulum quis nulla. Curabitur neque nibh, bibendum vel, eleifend sit amet, euismod at, leo. Duis auctor lobortis justo. Donec in tortor vel nibh rutrum pellentesque. Curabitur blandit pede quis neque. Nam sem eros, ornare a, pretium eget, condimentum sed, leo. Curabitur orci felis, elementum eget, aliquet vel, porta id, velit. Etiam justo neque, rhoncus quis, elementum vel, auctor vitae, urna.

Eidesstattliche Erklärung

Ich, Armin Schlegel, Matrikel-Nr. 2020863, versichere hiermit, dass ich meine Masterarbeit mit dem Thema

> *Entwicklung und Optimierung von Display-Schnittstellen fuer embedded Linux Boards -*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mir ist bekannt, dass ich meine Masterarbeit zusammen mit dieser Erklärung fristgemäß nach Vergabe des Themas in dreifacher Ausfertigung und gebunden im Prüfungsamt der Ohm-Hochschule abzugeben oder spätestens mit dem Poststempel des Tages, an dem die Frist abläuft, zu senden habe.

Nuernberg, den 29. März 2014

Armin Schlegel

A. Anhang

A.1. Liste möglicher Aktivitäten der BPEL

Die folgenden Listen basieren auf [?, Abschnitt 10 u. 11].

Tabelle A.1.: Ausgewählte elementare BPEL-Aktivitäten

Aktivität	Beschreibung	Beispiel
invoke	Aufruf einer Operation eines Webservice. Dabei wird zwischen *One-way*- und *Request-response*-Kommunikation unterschieden. Eventuelle Input- und Output-Nachrichten werden hierbei angegeben. invoke-Elemente können weitere Elemente (wie z. B. faultHandler zur Fehlerbehandlung) beinhalten.	`<invoke partnerLink="PLName" portType="PTName" operation="OName" inputVariable="VarName" outputVariable="VarName">`
receive	Empfängt eine Nachricht von einem Partner. Dazu muss die Operation angegeben werden, die der Prozess anbietet um die Nachricht entgegenzunehmen. Die Nachricht kann in einer variable gespeichert werden.	`<receive partnerLink="PLName" portType="PTName" operation="OName" variable="VarName">`
reply	Antwortet auf die Nachricht eines Partners (nur sinnvoll bei *Request-Response*-Kommunikation).	`<reply partnerLink="PLName" portType="PTName" operation="OName" variable="VarName">`
assign	Zuweisung von Werten zu Variablen.	`<assign> <copy> <from> <literal> <![CDATA[Wert]]> </literal> </from> <to variable="myVar" /> </copy> </assign>`
throw	Signalisierung eines Fehlers (analog zu Programmiersprachen).	`<throw faultName="FName" faultVariable="VarName">`

wait	Lässt den Prozess eine gewisse Zeit lang warten.	`<wait>`
		`  <until>`
		`  '2002-12-24T18:00'`
		`  </until>`
		`</wait>`

exit	Beendet den Prozess sofort.	`<exit>`

Tabelle A.2.: Ausgewählte strukturierte BPEL-Aktivitäten

Aktivität	**Beschreibung**	**Beispiel**

sequence	Sequentielles Abarbeiten der angegebenen Aktivitäten.	`<sequence>`
		`  <invoke>...</invoke>`
		`  <invoke>...</invoke>`
		`</sequence>`

flow	Paralleles Abarbeiten der angegebenen Aktivitäten.	`<flow>`
		`  <invoke>...</invoke>`
		`  <invoke>...</invoke>`
		`</flow>`

if	Konditionale Abfragen (vergleichbar zur Programmierung).	`<if>`
		`  <condition>`
		`  ...`
		`  </condition>`
		`  <sequence>`
		`  ...`
		`  </sequence>`
		`  <elseif>`
		`  ...`
		`  </elseif>`
		`  <else>`
		`  ...`
		`  </else>`
		`</if>`

while	Wiederholung der angegebenen Aktivitäten (vergleichbar zur Programmierung).	`<while>`
		`  <condition>`
		`  $orderDetails > 100`
		`  </condition>`
		`  <scope>...</scope>`
		`</while>`

scope	Verändern des Kontextes in dem die angegebenen Aktivitäten ablaufen. So können z. B. neue Variablen deklariert oder eine andere Fehlerbehandlung definiert werden. Das scope-Element stellt eigentlich keine Aktivität dar, soll hier aber trotzdem erwähnt werden.	`<scope>`
		`  <faultHandlers>`
		`  ...`
		`  </faultHandlers>`
		`  <flow>`
		`  <invoke>...</invoke>`
		`  </flow>`
		`</scope>`