

# UEBERSETZEN VON SCHRITTMOTORPROTOKOLLEN

## Entwurf eines Hardwareübersetzers

### Praxisbericht

im Fachgebiet Mess- und Sensortechnik



vorgelegt von: Johannes Dielmann  
Studienbereich: Technik  
Matrikelnummer: 515956  
Erstgutachter: Prof. Dr. Carstens-Behrens

© 2012

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

## Zusammenfassung

**(TODO: BESSER SCHREIBEN!)** Im Praxisprojekt wurde eine gegebene Aufgabe selbstständig umgesetzt. Dabei wurde Hardware unter anderem in Form von Platinenlayouts entwickelt, benötigte Bauteile recherchiert und bestellt, der Aufbau verändert und verbessert. Es wurde Software entwickelt und auftrende Probleme gelöst.

## Abstract

**(TODO: WTH IS AN ABSTACT??)**

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Verzeichnis der Listings

# 1 Einleitung

**(TODO: KLARSTELLEN DER BEGRIFFLICHKEITEN)** In der **(TODO: CAD ERKLÄREN)** CAD-Entwicklung kommt es vor das für ein real existierendes Objekt eine Erweiterung konstruiert werden muss. Um die Erweiterung sinnvoll konstruieren zu können müssen dazu die Abmessungen des Objektes möglichst genau bekannt sein. Das übertragen der Abmessungen, kann insbesondere für komplexe Objekte, sehr aufwendig sein. Abhilfe soll ein Laserscanner schaffen der das Objekt aus mehreren Richtungen vermisst und aus diesen Informationen ein genaues 3D-Modell davon generiert.

## 1.1 Motivation

Mit dem Aufbau aus RapidForm2004, Lasererfassungssystem VI-900 und Drehtisch sollen auf einfachem Wege 3D-Modelle eines Objektes erzeugt werden. Diese sollen anschließend in einer CAD-Software wie *Solidworks* nutzbar sein.

## 1.2 Ziel der Arbeit

Die Kommunikation zwischen der Software RapidForm2004 und dem gegebenen Drehtisch soll ermöglicht werden. Dazu werden die ASCII-Befehle der Software mit einem Mikrocontroller ausgewertet und in für den gegebenen Schrittmotor verständliche Befehle übersetzt. Es wird also ein Mikrocontroller mit 2 RS-232 Schnittstellen so programmiert das er die Befehle der Software übersetzen kann. Um den den Drehtisch manuell bedienen zu können und den aktuellen Status zu überprüfen sind noch ein LC-Display und mehrere Bedientaster vorgesehen.

Die Ansteuerung des Schrittmotors ist als Einschub für ein 19Rack realisiert. Daher wird die Platine für den Mikrocontroller auch als 19Einschub realisiert.

## 1.3 Aufbau der Arbeit

**(TODO: AUSBAUEN!)** Überblick verschaffen. Kommunikation mit Schrittmotor-Karte von PC aus. Aufbauen des STK500. Einarbeiten in uC Entwicklung. Steuern der Schrittmotor Karte vom uC aus. Erarbeiten der Protokolle (Reverse Engi-

neering). C-Programm zum verstehen eingehender Befehle. Übersetzen der Befehle. Neuer Mikrocontroller. Umgebungs wechsel. LC-Display. Endschalter. Max232. Platinenlayout.

## 1.4 Typographische Konventionen

**(TODO: WAS HIER?)** 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.



## 2 Einführung in Webservices

Wie bereits in Kapitel ?? 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.

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

#### 2.1.1 Service-orientierte Architektur

?<sup>1</sup> 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.“<sup>2</sup>

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.<sup>3</sup> Dabei wird zunächst noch kein Bezug zur Informationstechnologie hergestellt, sondern allgemein von Fähigkeiten gesprochen, die Personen, Unternehmen,

---

<sup>1</sup>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.

<sup>2</sup>[?, S. 8]

<sup>3</sup>Vgl. [?, S. 4]

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.<sup>4</sup>

---

<sup>4</sup>Vgl. [?, S. 8f.]

## 3 Die technische Implementierung von Webservices

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**<sup>5</sup> als Servlet-Engine, **Apache Axis2**<sup>6</sup> als SOAP-Engine und **ActiveBPEL**<sup>7</sup> als Workflow-System. Die Installation und Konfiguration der benötigten Anwendungen wird in Kapitel ?? im Anhang 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 ?? zeigt die Struktur eines einfachen *HelloWorld*-Webservice.

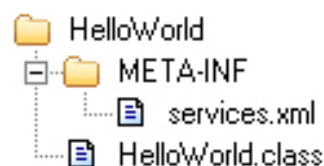


Abbildung 3.1: *HelloWorld*-Webservice: Dateistruktur

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

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

---

<sup>5</sup>Website: <http://tomcat.apache.org/>

<sup>6</sup>Website: <http://ws.apache.org/axis2/>

<sup>7</sup>Website: <http://www.activebpel.org/>

## 4 Netzwerkverkehr beim Aufruf von *PersonFactory*

### 4.1 SOAP-Request

Listing ?? zeigt die mitgeschnittene SOAP-Anfrage per HTTP an den Webservice *PersonFactory*. Wie am Ende von Kapitel ?? 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 aufzurufende Funktion wird noch einmal im SOAP-Body in Zeile 15 definiert.

Listing 4.1: SOAP-Request an *PersonFactory* per HTTP

### 4.2 SOAP-Response

Die Antwort des *PersonFactory*-Webservice zeigt Listing ?. 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 *CreatePersonResponse*, das die Antwort-Nachricht aus der WSDL repräsentiert.

Listing 4.2: SOAP-Response von *PersonFactory* per HTTP

## 5 Fazit und kritische Bewertung

Lorem ipsum dolor sit amet, consectetur 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 consectetur 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 versichere hiermit, dass ich meine Diplomarbeit mit dem Thema

*Uebersetzen von Schrittmotorprotokollen Entwurf eines Hardwareübersetzers*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Die Ergebnisse der Arbeit stehen ausschließlich dem auf dem Deckblatt angeführten Unternehmen zur Verfügung (**Arbeit mit Sperrvermerk**).

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

Vechta, den 31. Januar 2012

---

JOHANNES DIELMANN

# 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 <i>One-way</i> - und <i>Request-response</i> -Kommunikation unterschieden. Eventuelle Input- und Output-Nachrichten werden hierbei angegeben. <i>invoke</i> -Elemente können weitere Elemente (wie z. B. <i>faultHandler</i> zur Fehlerbehandlung) beinhalten.	<pre>&lt;invoke   partnerLink="PLName"   portType="PTName"   operation="OName"   inputVariable="VarName"   outputVariable="VarName"&gt;</pre>
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 <i>variable</i> gespeichert werden.	<pre>&lt;receive   partnerLink="PLName"   portType="PTName"   operation="OName"   variable="VarName"&gt;</pre>
reply	Antwortet auf die Nachricht eines Partners (nur sinnvoll bei <i>Request-Response</i> -Kommunikation).	<pre>&lt;reply   partnerLink="PLName"   portType="PTName"   operation="OName"   variable="VarName"&gt;</pre>
assign	Zuweisung von Werten zu Variablen.	<pre>&lt;assign&gt;   &lt;copy&gt;     &lt;from&gt;       &lt;literal&gt;         &lt;![CDATA[Wert]]&gt;       &lt;/literal&gt;     &lt;/from&gt;     &lt;to variable="myVar" /&gt;   &lt;/copy&gt; &lt;/assign&gt;</pre>
throw	Signalisierung eines Fehlers (analog zu Programmiersprachen).	<pre>&lt;throw   faultName="FName"   faultVariable="VarName"&gt;</pre>






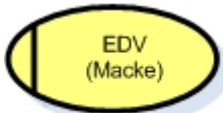

wait	Lässt den Prozess eine gewisse Zeit lang warten.	<pre> &lt;wait&gt;   &lt;until&gt;     '2002-12-24T18:00'   &lt;/until&gt; &lt;/wait&gt;                     </pre>
exit	Beendet den Prozess sofort.	<pre> &lt;exit&gt;                     </pre>

Tabelle A.2: Ausgewählte strukturierte BPEL-Aktivitäten

Aktivität	Beschreibung	Beispiel
sequence	Sequentielles Abarbeiten der angegebenen Aktivitäten.	<pre> &lt;sequence&gt;   &lt;invoke&gt;...&lt;/invoke&gt;   &lt;invoke&gt;...&lt;/invoke&gt; &lt;/sequence&gt;                     </pre>
flow	Paralleles Abarbeiten der angegebenen Aktivitäten.	<pre> &lt;flow&gt;   &lt;invoke&gt;...&lt;/invoke&gt;   &lt;invoke&gt;...&lt;/invoke&gt; &lt;/flow&gt;                     </pre>
if	Konditionale Abfragen (vergleichbar zur Programmierung).	<pre> &lt;if&gt;   &lt;condition&gt;     ...   &lt;/condition&gt;   &lt;sequence&gt;     ...   &lt;/sequence&gt;   &lt;elseif&gt;     ...   &lt;/elseif&gt;   &lt;else&gt;     ...   &lt;/else&gt; &lt;/if&gt;                     </pre>

while	Wiederholung der angegebenen Aktivitäten (vergleichbar zur Programmierung).	<pre> &lt;while&gt;   &lt;condition&gt;     \$orderDetails &gt; 100   &lt;/condition&gt;   &lt;scope&gt;...&lt;/scope&gt; &lt;/while&gt; </pre>
scope	Verändern des Kontextes in dem die angegebenen Aktivitäten ablaufen. So können z.B. neue Variablen deklariert oder eine andere Fehlerbehandlung definiert werden. Das <b>scope</b> -Element stellt eigentlich keine Aktivität dar, soll hier aber trotzdem erwähnt werden.	<pre> &lt;scope&gt;   &lt;faultHandlers&gt;     ...   &lt;/faultHandlers&gt;   &lt;flow&gt;     &lt;invoke&gt;...&lt;/invoke&gt;   &lt;/flow&gt; &lt;/scope&gt; </pre>

Tabelle A.3: Elemente der Ereignisgesteuerten Prozesskette

Element	Symbol
<b>Funktion</b> 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: <i>Auftrag anlegen, Rechnung schreiben, Konto abschließen</i>	
<b>Ereignis</b> 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: <i>Auftrag eingetroffen, Überweisung getätigt, Rechnung erstellt</i>	
<b>Operatoren</b> 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).	
<b>Organisationseinheit</b> 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: <i>Vertrieb, Personal, Produktion</i>	
<b>Informationsobjekt</b> 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: <i>Kundendatenbank, Versicherungsantrag, Rechnung</i>	

### 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.



## A.2 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<sup>8</sup> 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 Plugins 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,

---

<sup>8</sup>Dia, OpenOffice Draw und die EPC Tools.

einem umfangreichen Werkzeug zur Analyse des Netzwerkverkehrs, mitgeschnitten.

Website: <http://www.ethereal.com/>

- **L<sup>A</sup>T<sub>E</sub>X**

Diese Arbeit wurde mit L<sup>A</sup>T<sub>E</sub>X geschrieben. Als Distribution wurde MiKTeX verwendet und als Editor der LaTeX Editor.

Websites: <http://miktex.org/>, <http://www.latexeditor.org/>