Sind Prozessmanagement-Systeme auch auf eingebetteten Systemen einsetzbar?

Hendrik Bohn[*], Frank Golatowski[+], Dirk Timmermann[*]

[*] Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock, 18051 Rostock, {hendrik.bohn, dirk.timmermann}@uni-rostock.de
[+] Center for Life Science Automation (celisca), Friedrich-Barnewitz-Strasse 8, 18119 Rostock, frank.golatowski@celisca.de

1. Einleitung

Service-Orientierte Architekturen (SOA) haben in den letzten Jahren zunehmend an Bedeutung gewonnen. SOA (Jeckle et al. 2005) stellen Konzepte zur Beschreibung der Funktionalität von heterogenen Hard- und Softwarekomponenten (deren wohl definierte Schnittstellen *Dienste* genannt werden) und deren Interaktion bereit. Dazu gehören Themen wie Adressierung, Bekanntmachung, (Selbst-) Beschreibung und das Auffinden von Diensten. Der SOA Ansatz unterscheidet zwischen drei Rollen: Der *Dienstanbieter* stellt seine Funktionalität über eine Dienst-Schnittstelle bereit, die von einem *Dienstnutzer* verwendet werden kann. Einige SOA-Implementierungen spezifizieren zusätzlich auch ein *Dienstverzeichnis*, bei dem Dienstanbieter die Beschreibung von Diensten deren Nutzern zentral zur Verfügung stellen können.

Die am weitesten verbreitete Implementierung von SOA ist Web Services. *Web Services* (W3C 2004) ist ein Satz von modularen Protokollbausteinen, die sich verschiedenen Aspekten der Interaktion von Diensten widmen und entsprechend der gewünschten Anwendung zusammengesetzt werden können. Eine Übersicht über gängige Web Services Protokolle ist in Abbildung 1 dargestellt. XML Technologien und Transportprotokolle (z.B. HTTP) bilden das *Fundament* für Web Services. Die *Datentransferprotokolle* definieren Nachrichtenstruktur, Format eingebetteter Binärdaten und unterstützte Nachrichtenaustauschmuster (z.B. Request-Response, Notification). *Metadatenprotokolle* spezifizieren das Format der Dienstbeschreibung, Datenformate, Auffinden von Diensten, Anforderungen an die Dienstnutzung sowie Nutzung von Dienstverzeichnissen. Die *Zusicherungsprotokolle* behandeln Themen wie Sicherheit, Authentisierung und Zuverlässigkeit in der Interaktion mit Web Services. Verwaltung und Überwachung von einzelnen Diensten kann durch *Infrastrukturprotokolle* gewährleistet werden. Am oberen Ende befindet sich die Modellierung und Ausführung von (Geschäfts-)Prozessen. Mit Hilfe der *Web Services Business Execution Langua-*

ge (WS-BPEL) können Dienste zu einem Interaktionsprozess kombiniert werden, der wiederum einen Dienst darstellt. Die Ausführung von Prozessen wird von einer *WS-BPEL Engine* (Prozessmanagement-System) koordiniert und überwacht.

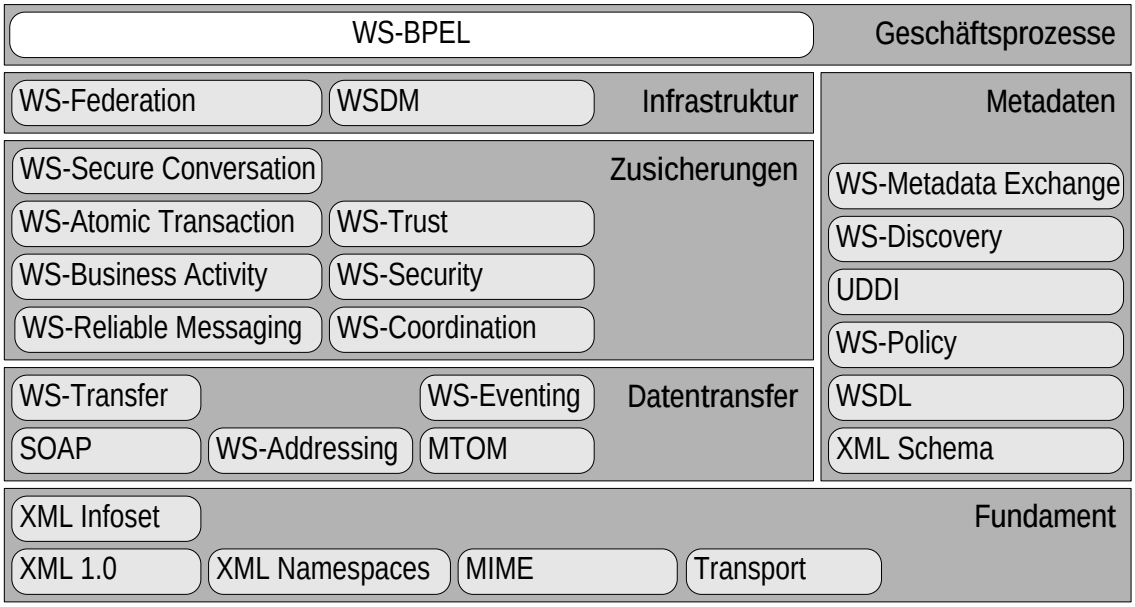

Abb. 1: Übersicht über die Web Services Protokolle und die Einordnung von WS-BPEL (angepasst von Ryan et al. 2007)

Mit dem Aufkommen von SOA-Implementierungen für Geräteensembles liegt auch ein Übertragen der Geschäftsprozessmodellierung auf Interaktionsprozesse zwischen Geräten nahe, welche unter anderem die Modellierung des Ablaufs in industriellen Fertigungsstraßen vereinfachen könnte. Aus der Initiative mehrerer Industrie- und Softwareunternehmen entstand 2004 das *Devices Profile for Web Services* (DPWS), welches eine Reihe von Web Services Protokollen anpasst (z.B. WS-Addressing, WS-Eventing und WS-Discovery), um den Anforderungen von eingebetteten Systemen zu genügen und trotzdem eine größtmögliche Interoperabilität mit anderen Web Services zu gewährleisten (Chan et al. 2006). Diese Arbeit diskutiert den Einsatz von Prozessmanagement-Systemen auf Basis von Web Services (WS-BPEL Engine) auf eingebetteten Systemen und geht auf die Vor- und Nachteile der Anwendungsmöglichkeiten ein. Auf die darunter liegenden Protokolle wird nicht im Einzelnen eingegangen.

2. Web Services Business Execution Language (WS-BPEL)

WS-BPEL (OASIS 2007) spezifiziert eine auf andere Web Services Standards aufbauende, XML-basierte Sprache zur Beschreibung von Web Services Prozessen. Ein *WS-BPEL Prozess* ist ein aus Web-Services-Interaktionen zusammengesetzter Dienst, der wie jeder andere Dienst über eine eigene Beschreibung verfügt. WS-BPEL Prozesse können mit Hilfe von sogenannten *WS-BPEL Designern* grafisch erzeugt werden und ermöglichen auf diese Weise „Programming in the large“ (Ryan et al. 2007). D.h. WS-BPEL Prozesse können auch von Per-

sonen, die keine Programmierer sind, erstellt werden. Ein WS-BPEL Prozess wird nach dessen Erstellung auf eine WS-BPEL Engine ausgebracht und erzeugt beim Aufruf immer eine Instanz desselben Prozesses, die dann ausgeführt wird. So kann zum Beispiel ein Bestellprozess von verschiedenen Anwendern gleichzeitig genutzt werden (jeder mit seiner eigenen Instanz).

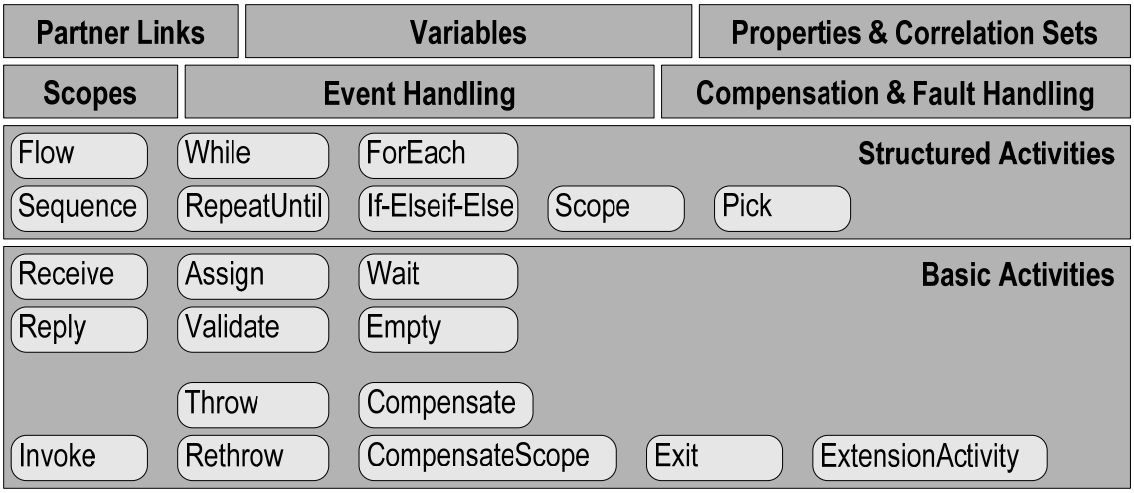

Abb2: Komponenten von WS-BPEL

Abbildung 2 stellt die Komponenten der WS-BPEL Spezifikation dar. *Partner-Links* definieren die an einer Aktion (*Activity*) beteiligten Web Services und ihre Rollen in der Interaktion. Die Komponente *Variables* spezifiziert Methoden zur Definition von Datentypen, der Speicherung und Konvertierung von Daten, die für die Interaktionen notwendig sind. *Properties & Correlation Sets* ermöglichen eine eindeutige Identifizierung einer Prozessinstanz, die wichtig wird, wenn mehrere Instanzen des gleichen Prozesses gleichzeitig aktiv sind. So können mit Hilfe von *Correlations* jeder Bestellung eine Bestellnummer zur eindeutigen Identifizierung zugeordnet und verwendet werden. *Scopes* ermöglichen die Definition von Gültigkeitsbereichen innerhalb einer Prozessdefinition (z.B. für Variablen). Das *Event Handling* ermöglicht die Definition und Behandlung von Ereignissen, welche entweder auf eine bestimmte Zeitspanne oder das Eintreffen einer bestimmten Nachricht basieren. Das *Compensation & Fault Handling* dient dem Abfangen und der Behandlung von Fehlern sowie die Kompensierung von fehlerhaften Aktionen, die nicht mehr rückgängig gemacht werden können.
Die Aktionen (Activities) sind unterteilt in elementare Aktionen (*Basic Activities*) und Aktionen, die eine Strukturierung des Prozesses ermöglichen (*Structured Activities*). Der Einsprungspunkt einer Prozessinstanz ist der Aufruf der *Receive*-Aktivität durch das Empfangen einer Nachricht. Die Prozessinstanz wird gestartet und gibt optional bei Beendigung eine Nachricht mit den Ergebnissen zurück (*Reply*). *Invoke* dient dem Aufruf eines Web Services, *Assign* weißt Variablen einen Wert zu und *Validate* überprüft das Format einer Variablen. Fehlermeldungen können mit *Throw* erzeugt werden, die an darüberliegende Gültigkeitsbereiche mit *Rethrow* weitergegeben werden können. Treten Fehler auf,

so können sie innerhalb des aktuellen Gültigkeitsbereiches mit *Compensate* oder innerhalb eines bestimmten Gültigkeitsbereiches mit *CompensateScope* kompensiert werden. *Wait* definiert ein bestimmtes Zeitintervall, um mit der weiteren Ausführung des Prozesses zu warten. *Empty* spezifiziert eine leere Aktivität, die genutzt werden kann, um parallele Aktivitäten zu synchronisieren. *Exit* beendet den Prozess und *ExtensionActivity* dient der Erweiterung von WS-BPEL durch neue Aktivitäten.
Bei den Structured Activities werden *While*, *RepeatUntil* und *ForEach* genutzt, um Schleifen innerhalb eines Prozesses zu definieren. *Flow* dient der Programmierung von Parallelität und *Sequence* der strikten Reihenfolge von Anweisungen. *If-Elseif-Else* ermöglicht die Umsetzung von bedingten Anweisungen. *Scope* definiert Gültigkeitsbereiche, und *Pick* hält die Ausführung so lange an, bis eine bestimmte Nachricht eingetroffen ist.

3. Möglichkeiten des Einsatzes von WS-BPEL auf eingebetteten Systemen

Im herkömmlichen Einsatz von WS-BPEL wird der Prozessablauf mit einem WS-BPEL Designer (grafisches Tool oder einfacher Texteditor) erstellt. Dazu können die Web Services Beschreibungen (in Web Service Description Language – WSDL) der zu kombinierenden Dienste sowie Datentypdefinitionen (definiert in XML Schema) verwendet werden. Weiterhin ergibt sich aus dem Prozessablauf die WSDL Beschreibung für den dazu gehörenden, zusammengesetzten Dienst. Sind der Prozessablauf und seine WSDL erstellt, so werden beide auf eine WS-BPEL Engine ausgebracht und somit aufruf- bzw. ausführbar gemacht. Jeder Aufruf des zusammengesetzten Dienstes/Prozesses erstellt eine eigene Prozessinstanz. Abbildung 3 illustriert den beschriebenen Ablauf.

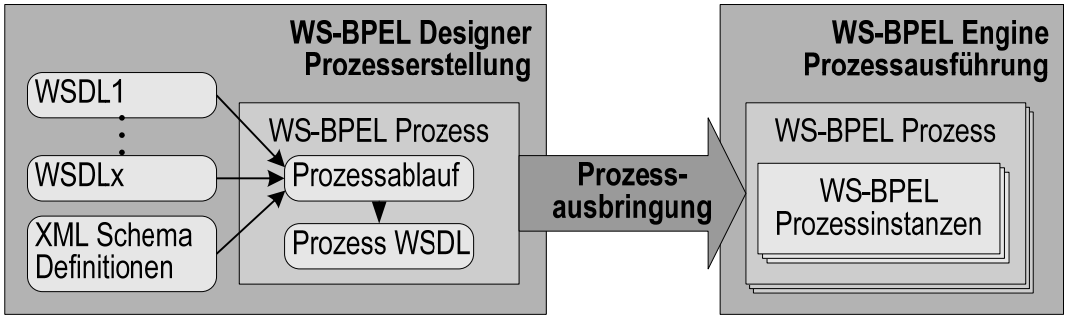

Abb. 3: Erstellung, Ausbringung und Ausführung von Prozessen

Es ergeben sich drei Möglichkeiten für den Einsatz von WS-BPEL Engines in Umgebungen mit eingebetteten Systemen: 1. Installation der Engine auf einem eingebetteten System, 2. Steuerung von Diensten auf eingebetteten Systemen durch eine zentrale WS-BPEL Engine und 3. Übersetzung des Prozesses in ein lauffähiges Programm, das auf einem eingebetteten System installiert wird.
Installation der Engine auf einem eingebetteten System: WS-BPEL Engines haben auf Grund ihrer Komplexität sehr hohe Anforderungen an zugrunde lie-

gende Hard- und Software. Das liegt vor allen Dingen an dem Umstand, dass jeglicher WS-BPEL konformer Prozessablauf auf eine entsprechende Engine ausgebracht werden kann und dort nach deren Aufruf zur Laufzeit interpretiert wird. In ihrer Studie „Vergleich von BPEL Laufzeitumgebungen“ (Hantschel et al. 2006) stellen die Autoren Hantschel, Ruf und Strotbek fest, dass aktuelle WS-BPEL Engines zwischen 20 MByte und 2,8 GByte Festplattenspeicher, 256 – 512 MByte RAM und einen Prozessor ab 300 MHz benötigen. Weiterhin setzen einige WS-BPEL Engines vorhandene Software wie Webserver oder Betriebsysteme wie Windows voraus. Diese Anforderungen und Voraussetzungen übersteigen in den meisten Fällen die Leistungsgrenzen von herkömmlichen eingebetteten Systemen.

Steuerung von Diensten auf eingebetteten Systemen: Auf der anderen Seite ist die Verwendung von eingebetteten Systemen als Plattform für ein oder mehrere Web Services bestens geeignet. So wurden im SIRENA Projekt (Service Infrastructure for Real-time Embedded Networked Applications – Bohn et al. 2006) auf eingebettete Systeme der Industrie, Telekommunikation, Heimautomation und Automobilelektronik unterschiedliche Dienste ausgebracht, die miteinander kommunizieren und interagieren konnten. Leider unterstützt WS-BPEL das DPWS und einige von dessen zugrundeliegenden Protokollen noch nicht. Unter anderem werden von WS-BPEL nur die Nachrichtenmuster Request-Response und One-Way unterstützt. DPWS implementiert zusätzlich ein Publish/Subscribe Mechanismus, um Dienstnutzern die Möglichkeit zu geben, über eventuelle Zustandsänderungen der Dienste informiert zu werden. Das wird über die Nachrichtenmuster Notification und Solicit-Reponse realisiert. Weiterhin interagiert jeder Dienst in einem herkömmlichen Prozess ausschließlich mit der WS-BPEL Engine. DPWS hingegen unterstützt auch die Kommunikation mit dritten Diensten innerhalb einer Interaktion. Dadurch kann zum Beispiel eine Kontrolleinheit veranlassen, dass Videodaten von einem Videoserver auf einem Abspielgerät angezeigt werden können. Ausserdem sind die von WS-BPEL genutzten Protokolle noch sehr stark an darunter liegende Transportprotokolle gebunden. Diese Bindung wurde von DPWS aufgebrochen, um eine höhere Interoperabilität zu erreichen.

Übersetzung von WS-BPEL Prozessen in ausführbare Programme: Eine interessante Möglichkeit WS-BPEL Prozesse von eingebetteten Systemen mit begrenzten Ressourcen zu steuern, ist die Beschränkung einer WS-BPEL Engine auf einen einzigen ausführbaren Prozess. Dieser Prozess wird mit Hilfe eines WS-BPEL Designers erstellt und zur Designzeit in ein ausführbares Programm übersetzt, anstelle den Prozess zur Laufzeit durch eine WS-BPEL Engine zu interpretieren. Dadurch wird der Speicherbedarf des zusammengesetzten Prozesses erheblich reduziert, die Performance eines solchen Prozesses kann durch Quellcodeoptimierung gesteigert und der Stromverbrauch kann gesenkt werden, da ein Prozess nur dann aktiv ist, wenn er benutzt wird. Ein weiterer Vorteil dieser Lösung ist, dass sie die gleiche Flexibilität wie der herkömmliche Ansatz zur

Nutzung einer WS-BPEL Engine als Interpreter von Prozessen (siehe Fall 1) besitzt. Den Autoren ist keine Arbeit bekannt, die sich mit der Kompilierung von WS-BPEL Code beschäftigt.

4. Fazit

Der Einsatz von WS-BPEL Engines auf einem eingebetteten System stellt durch deren große Komplexität hohe Anforderungen an die Ressourcen eines eingebetteten Systems.
Ein interessanter Ansatz hingegen ist die Benutzung eines Übersetzers für WS-BPEL Prozesse, der einen Prozess in ein ausführbares Programm auf einem eingebetteten System umsetzt. Leider existiert nach dem Wissen der Autoren bisher keine Arbeit über diesen Ansatz. Ein zunehmender Bedarf für einen solchen Übersetzer ist aber zu erwarten, da er vielfältigen Einsatzmöglichkeiten in der Industrie und Telekommunikation bietet.

Danksagung

Diese Arbeit wurde über das LOMS Projekt vom Bundesministerium für Bildung und Forschung (BMBF) unter der Referenznummer 01|SF11H gefördert.

5. Literatur

Bohn H., Bobek A., Golatowski F. (2006), 5th International Conference on Networking ICN'06, Mauritius.

Dostal W., Jeckle, M., Melzer I., Zengler B. (2005), Service-Orientierte Architekturen mit Web Services, Spektrum Akademischer Verlag, Heidelberg.

Chan S., Conti D., Kaler C., Kuehnel T., Regnier A., Roe B., Sather D., Schlimmer J., Sekine H., Thelin J., Walter D., Weast J., Whitehead D., Wright D., Yarmosh Y. (2006), Devices Profile for Web Services, Microsoft.

Hantschel R., Ruf F., Strotbek H. (2006), Vergleich von BPEL Laufzeitumgebungen, Fachstudie Nr. 54, Institut für Architektur von Anwendungssystemen, Universität Stuttgart.

Local Mobile Services (LOMS) (2007), http://www.loms-itea.org.

Ryan F., König D., Barreto C. (2007), WS-BPEL Technical Overview for Developers and Architects, Präsentation von OASIS WS-BPEL 2.0, http://www.oasis-open.org/events/webinars/2007-03-13-WS-BPEL-Technical-Overview.wmv.

OASIS (2007), Web Services Business Process Execution Language Version 2.0, OASIS Standard.

W3C (2004), Web Services Architecture, W3C Note.