Erweiterung einer Open–Source BPEL–Engine um eine mobile Komponente

Stephanie Gamm[1], Olaf Zukunft[2]

[1] sd&m, software design & management AG, Lübecker Straße 128,
22087 Hamburg, stephanie.gamm@sdm.de
[2] HAW Hamburg, Department Informatik, Berliner Tor 7,
20099 Hamburg, zukunft@informatik.haw-hamburg.de

Abstract: Für Service–orientierte Architekturen steht mit BPEL eine standardisierte Sprache für die Koordination von Abläufen zur Verfügung. Mobile Nutzer werden dagegen in BPEL nicht explizit unterstützt. Jedoch bietet BPEL auf Sprachebene eine standardisierte Möglichkeit für die Funktionserweiterung an. Basierend auf dem Entwurf einer Spracherweiterung zur Unterstützung mobiler Nutzer wird hier beschrieben, wie mit ActiveBPEL, einer Open–Source–Realisierung einer BPEL–Ausführungsumgebung, die Umsetzung solcher Spracherweiterungen möglich ist.

1 Einleitung

Ein *Geschäftsprozess* ist nach van der Aalst und van Hee die Abfolge von Aktivitäten zur Erreichung eines gemeinsamen Ziels [vdAvH02]. Ein Geschäftsprozess setzt sich aus einzelnen Aktivitäten und einer Menge an Bedingungen zusammen, die die Reihenfolge der auszuführenden Aktivitäten beschreiben. Dabei wird eine Aktivität als eine logische Einheit verstanden, die von einer Person, einer Maschine oder einer Gruppe von Personen oder Maschinen ausgeführt wird. Weiterhin kann ein Geschäftsprozess Sub-Prozesse enthalten, die sich ihrerseits wiederum wie Geschäftsprozesse zusammensetzen.

Während beim Geschäftsprozess die betriebswirtschaftliche Sichtweise im Vordergrund steht, handelt es sich bei einem *Workflow* um die IT-seitige Repräsentation eines Geschäftsprozesses [LR00]. Systeme der Informationstechnik implementieren die zugrundeliegenden Geschäftsprozesse und führen diese als Workflows aus. Dabei kann der Ablauf in einer verteilten und heterogenen Umgebung sowie unternehmensübergreifend stattfinden.

Die traditionelle Abarbeitung von Workflows berücksichtigt insbesondere Nutzer, die über ein Netzwerk ständig miteinander verbunden sind. Darüber hinaus müssen jedoch auch mobile Nutzer wie reisende Vertreter, mobile Servicetechniker u.a. in solche Workflowsysteme integriert werden [Zuk98]. Dies gilt auch und insbesondere in Systemen, die auf Web Services als Schnittstelle zum Aufruf von Aktivitäten zurückgreifen. Im nächsten Abschnitt wird BPEL als sprachliche Grundlage solcher Systeme beschrieben. Anschließend wird dargestellt, mit welchen Konzepten Workflowsprachen erweitert werden müssen, um mobile Nutzer zu unterstützen. In Abschnitt 4 wird danach die Open–Source–Aus-

führungsumgebung ActiveBPEL vorgestellt, die die beschriebene Spracherweiterung unterstützen kann. Der Abschnitt 5 diskutiert den Entwurf und die Realisierung der ActiveBPEL–Erweiterung, bevor dieser Aufsatz mit einer Bewertung und einem Ausblick endet.

2 BPEL

In den letzten Jahren sind verschiedene Sprachen im Bereich der Web Service Orchestration entwickelt worden. Zu den ersten Ansätzen gehören die blockbasierte, algebraische Prozessmodellierungssprache XLANG, die im Jahr 2000 von Microsoft veröffentlicht wurde, und die graphbasierte Web Services Flow Language (kurz: WSFL), eine Entwicklung von IBM aus dem Jahr 2001 [Pel03]. Beide unterstützen die Interaktion verschiedener Web Services in Form von Geschäftsprozessen.

Aus einer Zusammenführung von XLANG und WSFL ist die *Business Process Execution Language* (kurz: *BPEL*) entstanden [BPE07]. BPEL vereint beide Ansätze und bietet als XML-basierte Sprache ein umfangreiches Vokabular zur Beschreibung von Geschäftsprozessen durch Komposition von Web Services an.

In BPEL wird die exakte Reihenfolge festgelegt, in der die am Prozessablauf beteiligten Web Services aufgerufen werden. Die wichtigsten Elemente für einen BPEL-Prozess bilden die *Aktivitäten* (engl.: activities). Dabei unterscheidet BPEL zwischen den sogenannten Basic Activities und den Structured Activities. Basic Activities umfassen insbesondere den Aufruf von Web Service–Interaktionen. Structured Activities bieten die üblichen Kontrollstrukturen von Workflowbeschreibungen an.

Ein ausführbarer BPEL-Prozess beschreibt aus der Sicht eines Teilnehmers den gesamten internen Ablauf eines Geschäftsprozesses durch Kombination einzelner Web Service-Aufrufe. Die resultierende Prozessdefinition wird durch Deployment auf einer BPEL-Engine zur Ausführung gebracht. Ein solcher Prozess stellt selbst wieder einen Web Service dar. Dadurch lassen sich auf Basis von Prozessbeschreibungen verschiedene Abstraktionsebenen schaffen, und die verschiedenen, selbst als Service fungierenden Prozesse miteinander kombinieren.

3 Spracherweiterung

Zur Unterstützung mobiler Nutzer eines BPEL–basierenden Workflowsystems bietet es sich an, BPEL um zusätzliche Sprachkonstrukte zu erweitern. Dies ist möglich, da bei der Standardisierung von WS-BPEL 2.0 Erweiterungsmöglichkeiten in der Sprache verankert wurden. Bei den Erweiterungen kann es sich um neue Attribute oder Elemente handeln. Ebenso können Zuweisungsoperationen und Aktivitäten ergänzt werden, um neue Funktionalitäten zu realisieren. Dabei darf die Semantik von bestehenden BPEL-Elementen nicht verändert werden. Erweiterungen müssen in einem eigenen Namensraum definiert werden. Anschließend können sie hierüber qualifiziert und in BPEL-eigene Elemente in-

tegriert werden. Das folgende Listing gibt die Syntax der Elemente *<extensions>*, *<extension>* und *<extensionActivity>* unter Verwendung der im BPEL-Standard festgelegten Konventionen wieder, mittels derer die Einbindung von Erweiterungen in einen BPEL-Prozess vorgenommen wird. Für weitere Details sei auf [BPE07] verwiesen.

```
<process ...>
  <extensions>?
    <extension namespace="anyURI"
               mustUnderstand="yes|no" />+
  </extensions>
  ...
  <extensionActivity>
    <anyElementQName standard-attributes>
      standard-elements
    </anyElementQName>
  </extensionActivity>
  ...
</process>
```

Für mobile Nutzer wurden in [Gam08] mehrere Spracherweiterungen konzipiert, die folgende Zusatzfunktionalität bieten:

1. Ausführung einer Aktivität abhängig vom aktuellen Kontext
2. Auswahl und dynamisches Binding eines Service-Providers abhängig vom aktuellen Kontext
3. Definition eines Sub-Prozesses und dessen Übertragung zur Laufzeit an andere Ausführungsumgebung
4. Empfang eines Sub-Prozesses und dessen Ausführung

Unter Kontext kann z.B. die Position eines mobilen Teilnehmers, seine aktuelle Netzwerkanbindung, seine Arbeitssituation oder ähnliches formal gefasst werden. Abhängig vom modellierten Kontext und dem situationsabhängigen Kontext des mobilen Teilnehmers können dann Aktivitäten gestartet werden.

Insgesamt wurden fünf Aktivitäten als BPEL-Erweiterung entworfen. Mittels `<contextActivity>` kann die Ausführung einer eingebundenen BPEL-Aktivität abhängig vom Übereinstimmen des aktuellen Kontexts mit definierten Vorgaben gemacht werden. Durch `<dynamicInvoke>`, einer spezialisierten Form des Standard-Invoke, wird der aufzurufende Partner-Service erst zur Laufzeit anhand von Kontextvorgaben bestimmt und dynamisch an den Operationsaufruf gebunden. Die Definition eines Sub-Prozesses und dessen Übertragung an eine andere Ausführungsumgebung kann über die neuen Konstrukte `<sendProcess>`, `<receiveProcess>` und `<startProcess>` vorgenommen werden. Das Starten eines Prozesses wurde vom eigentlichen Empfang entkoppelt, um andere Aktivitäten zwischenschieben zu können, z.B. um die Ausführung des Sub-Prozesses abzulehnen. Für weitere Details sei auf [Gam08] verwiesen.

4 ActiveBPEL

Zur experimentellen Umsetzung der konzipierten Spracherweiterungen wurde ActiveBPEL als Basis gewählt. Die *ActiveBPEL Engine* ist ein Open–Source-Projekt von Active Endpoints, bei dem es sich um eine Ausführungsumgebung für BPEL-Prozesse handelt [Act07a]. Sowohl die Spezifikation BPEL4WS 1.1 als auch der Standard WS-BPEL 2.0 wird von der Engine in der Version 4.0 von Juli 2007 unterstützt.

Die quelloffene ActiveBPEL Engine ist in Java geschrieben und setzt einen installierten Servlet Container wie beispielsweise Apache Tomcat voraus. Während der Installation wird die Engine in den Web Server deployed und kann anschließend zusammen mit diesem automatisch gestartet werden. Abbildung 1 zeigt die Engine-Architektur und ihre wesentlichen Komponenten.

Für die Kommunikation via SOAP-basierten Web Services wird Apache Axis verwendet. Die Anbindung der BPEL-Engine an den Web Service Container realisieren sogenannte WS-Handler. Sie regeln die Weitergabe von ein- und ausgehenden Nachrichten zwischen Axis und der ActiveBPEL Engine.

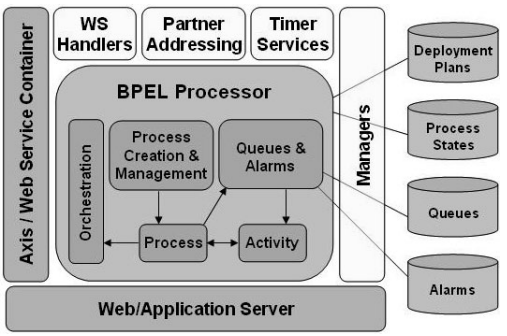

Abbildung 1: Architektur der ActiveBPEL Engine (aus [Act07a])

5 Entwurf und Realisierung der ActiveBPEL–Erweiterung

Damit eine sinnvolle Integration der neuen Funktionalitäten in die ActiveBPEL Engine erfolgen kann, muss beim Erstellen der Architektur die vorliegende Engine-Architektur berücksichtigt werden. Hierzu dienten die unter [Act07a] und [Act07b] veröffentlichten Teile der Dokumentation wie auch die Angaben im Quelltext der Engine, die über Javadoc zur Verfügung stehen. Darüber hinausgehende Zusammenhänge bezüglich des konzeptionellen Aufbaus der ActiveBPEL Engine wurden durch Reverse Engineering erarbeitet, da die verfügbaren Dokumente hinsichtlich ihres Informationsgehalts nicht ausreichend waren. Ein Großteil der entwickelten Konzepte beruht demnach auf den so erzielten Erkenntnissen, die in einem Closed–Source–Produkt nicht verfügbar wären.

Da der BPEL-Standard wie oben dargestellt die Erweiterung der Sprache um neue Aktivitäten durch Verwendung des `<extensionActivity>`-Konstrukts vorsieht, wäre eine definierte, in sich geschlossene Schnittstelle seitens der BPEL–Engine wünschenswert. Weder die zuvor genannte Dokumentation noch die eigenen Untersuchungen deuten auf eine solche Unterstützung in ActiveBPEL hin. Dies führt dazu, dass die Integration der neuen Aktivitäten direkt in die bestehenden Komponenten vorgenommen werden muss, beispielsweise durch Unterklassen-Bildung oder ergänzende Anweisungen beim Visitor-Pattern. Für die anschließende Realisierung bedeutet dies, dass die Erweiterungen direkt im Quellcode der ActiveBPEL Engine ansetzen müssen und der originale Engine-Code nicht unverändert als Basis dienen kann.

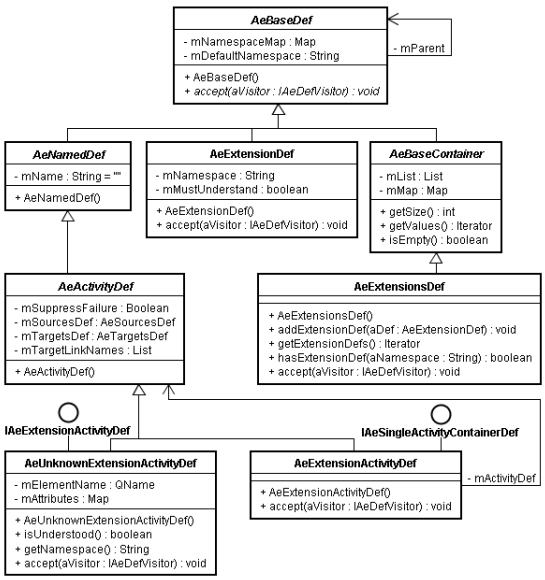

Abbildung 2: Klassendiagramm der Sprachdefinitionskomponente

Abbildung 2 zeigt exemplarisch das Klassendiagramm für die Definitions-Objekte, die den BPEL-Erweiterungsmechanismus umsetzen. Hierzu gehören die Sprachkonstrukte `<extensions>`, `<extension>` und `<extensionActivity>`.

6 Bewertung und Ausblick

In dieser Arbeit wurden Erweiterungsmöglichkeiten von BPEL beschrieben, mit denen sich eigene Sprachelemente entwickeln und in den bestehenden Sprachumfang integrieren lassen. Dieser Mechanismus wird für die Entwicklung der mobilen Spracherweiterungen genutzt. Als Ausführungsumgebung für BPEL-Prozesse wird eine Engine benötigt. Statt eine komplette BPEL-Engine zu entwickeln, wurden die mobilen Spracherweiterungen in

eine verfügbare Engine integriert. Hierfür kommt nur eine Open–Source–Implementierung in Frage, die den direkten Zugriff auf die Interna der Engine ermöglicht. Ausgewählt wurde die ActiveBPEL Engine, nachdem ein Vergleich u.a. mit dem Apache–Produkt ODE durchgeführt wurde und einen deutlichen Entwicklungsvorsprung von ActiveBPEL ergeben hat.

Die Integration der neuen Funktionalitäten in die Engine wurde unter Berücksichtigung der bereits existierenden Engine-Architektur vorgenommen und sogenannte Definitions- und Implementierungs-Objekte entworfen, um die Sprachkonstrukte in engine-interne Abbildungen transformieren und interpretieren zu können. Für die Umsetzung der jeweiligen Aktivitäts-Logik sind Lösungen realisiert worden.

In einem Anschlussprojekt werden derzeit die entworfenen Spracherweiterungen in einen grafischen BPMN–Editor eingebunden und ein automatischer Übersetzungsprozess in das erweiterte BPEL realisiert. Die Anwendbarkeit wird in einem Anwendungsszenario für mobile Versicherungsmakler erprobt.

Literatur

[Act07a] ActiveBPEL Engine. `http://www.active-endpoints.com/active-bpel-engine-overview.htm5`, 2007.

[Act07b] ActiveBPEL InfoCenter v4.0. `http://www.activebpel.org/infocenter/ActiveBPEL/v40/index.jsp`, 2007.

[BPE07] Web Services Business Process Execution Language Version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`, 11.04.2007.

[Gam08] Stephanie Gamm. Konzeption und Realisierung einer Sprache und Engine für mobile Workflows. *Masterarbeit. HAW Hamburg*, 2008.

[IBM05] WS-BPEL Extension for Sub-processes – BPEL-SPE. `http://www.ibm.com/developerworks/library/specification/ws-bpelsubproc/`, September 2005.

[LR00] Frank Leymann und Dieter Roller. *Production Workflow – Concepts and Techniques*. Prentice Hall International, Upper Saddle River, 2000.

[Pel03] Chris Peltz. Web services orchestration and choreography. IEEE Computer Vol 36(10):46-52, Oct. 2003.

[vdAvH02] Wil van der Aalst und Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, 2002.

[Zuk98] Olaf Zukunft. Integration mobiler und aktiver Datenbankmechanismen als Basis für die ortsungebundene Vorgangsbearbeitung. *Dissertation. Universität Oldenburg*, 1998.