

Spezifikation

Version 1.1

19. Oktober 2009



Inhaltsverzeichnis

1	Einleitung	4
1.1	Zweck des Dokuments	4
1.2	Einsatzbereich und Ziele	4
1.3	Evolution des Dokuments	5
1.3.1	Basisfunktionalität	5
1.3.2	Ausblick	5
1.4	Definitionen	6
1.5	Überblick	6
2	Allgemeine Beschreibung	6
2.1	Einbettung in die Systemumgebung	6
2.2	Funktionen	7
2.3	Sprache	7
2.4	Distributionsform und Installation	8
2.5	Benutzerprofile	8
2.6	Einschränkungen	8
3	Nichtfunktionale Anforderungen	8
3.1	Mengengerüst	8
3.2	Benutzbarkeit	8
3.3	Verfügbarkeit	9
3.4	Robustheit	9
3.5	Sicherheit	9
3.6	Portabilität	9
3.7	Erweiterbarkeit	9
3.8	Wartbarkeit	9
3.9	Skalierbarkeit	9
4	Benutzeroberfläche des SIMPL Eclipse Plug-Ins	10
4.1	Admin-Konsole	10
4.1.1	Globale Einstellungen	11
4.1.2	Auditing	12
4.1.3	Referenzen	13
4.2	Data-Management-Aktivitäten	15
5	Akteure	16
5.1	Prozess-Modellierer	16
5.2	Workflow-Administrator	16
5.3	ODE Workflow-Engine	16
5.4	Eclipse BPEL Designer	16
5.5	SIMPL Core	16
6	Anwendungsfälle (Use-Cases)	17
6.1	Diagramm der Anwendungsfälle	17
6.2	Anwendungsfälle des Prozess-Modellierers	17
6.2.1	Data-Management-Aktivität erstellen	18
6.2.2	Data-Management-Aktivität bearbeiten	18
6.2.3	Data-Management-Aktivität löschen	19
6.2.4	Prozess ausführen	19
6.3	Anwendungsfälle des Workflow-Administrators	19
6.3.1	Admin-Konsole öffnen	20

6.3.2	Admin-Konsole zurücksetzen	21
6.3.3	Admin-Konsole speichern	21
6.3.4	Admin-Konsole schließen	21
6.3.5	Admin-Konsole Defaults laden	22
6.3.6	Auditing aktivieren	22
6.3.7	Auditing deaktivieren	22
6.3.8	Auditing-Datenbank festlegen	22
6.3.9	Globale Einstellungen festlegen	23
6.3.10	Neue Referenz in RRS einfügen	23
6.3.11	Referenz aus RRS bearbeiten	23
6.3.12	Referenz aus RRS löschen	24
6.4	Anwendungsfälle der ODE Workflow-Engine	24
6.4.1	Data-Management-Aktivität ausführen	25
6.4.2	Data-Management-Aktivität kompensieren	25
6.5	Anwendungsfälle des Eclipse BPEL Designers	25
6.5.1	Admin-Konsole laden	26
6.6	Anwendungsfälle des SIMPL Core	26
6.6.1	BPEL-Datei transformieren	27
6.6.2	Neues Datenquellen-Plug-In in RRS einfügen	27
6.6.3	Referenz auflösen/dereferenzieren	27
7	Konzepte und Realisierungen	27
7.1	Referenzen in BPEL	28
7.1.1	Reference Resolution System	28
7.1.2	Referenzen und Endpoint References	30
7.1.3	Realisierung von Referenzen in BPEL	31
7.2	Data-Management-Aktivitäten	33
7.2.1	Datenmanagement-Patterns	33
7.2.2	Umsetzung der Datenmanagement-Patterns	36
7.2.3	Resultierende BPEL-Aktivitäten	40
7.3	Authentifizierung und Autorisierung	46
7.3.1	Authentifizierung	46
7.3.2	Autorisierung	46
7.4	Transaktionen	46
7.5	Auditing	46
7.5.1	Momentane Situation - Monitoring von ODE	46
7.5.2	Auditing von SIMPL	47
7.5.3	Event Modell	47
8	Technologien und Werkzeuge	49
8.1	Technologien	49
8.2	Werkzeuge	50

Änderungsgeschichte

Version	Datum	Autor	Änderungen
0.1	11.09.2009	hahnml	Erstellung des Dokuments.
0.2	14.09.2009	zoabfs, hahnml, xitu	Kapitel 3 eingefügt.
0.3	17.09.2009	hahnml, zoabisfs	Kapitel 7 eingefügt.
0.4	21.09.2009	hahnml, zoabisfs, rehnre	Kapitel 5 und 6 eingefügt.
1.0	28.09.2009	hahnml, zoabisfs, schneimi, bruededl, huettiwg, rehnre	Abschließende Überarbeitung des Dokuments.
1.1	19.10.2009	hahnml, schneimi, rehnre	Korrektur der Spezifikation nach dem Review mit den Kunden.

1 Einleitung

In diesem Kapitel wird der Zweck dieses Dokuments, sowie der Einsatzbereich und die Ziele der zu entwickelnden Software beschrieben. Weiterhin werden die in diesem Dokument verwendeten Definitionen erläutert und ein Überblick über das restliche Dokument gegeben.

1.1 Zweck des Dokuments

Diese Spezifikation ist die Grundlage für alle weiteren Dokumente, die im Rahmen dieses Projekts entstehen. In ihr sind sämtliche Anforderungen an die zu entwickelnde Software festgelegt. Sie muss stets mit den anderen Dokumenten, insbesondere mit dem Entwurf und der Implementierung, konsistent gehalten werden. Die Spezifikation dient den Team-Mitgliedern als Grundlage und Richtlinie für die Entwicklung der Software und den Kunden als Zwischenergebnis zur Kontrolle.

Zum Leserkreis dieser Spezifikation gehören:

- Die Entwickler der Software,
- die Kunden und
- die Gutachter der Spezifikationsreviews.

1.2 Einsatzbereich und Ziele

Das Entwicklungsteam soll ein erweiterbares und generisches Rahmenwerk für die Modellierung und Ausführung von Workflows erstellen, welches den Zugriff auf nahezu beliebige Datenquellen ermöglichen soll. Bei den Datenquellen kann es sich beispielsweise um Sensornetze, Datenbanken und Dateisysteme handeln. Der Schwerpunkt soll klar auf wissenschaftlichen Workflows liegen, in denen es möglich sein muss große heterogene Datenmengen verarbeiten zu können. Über das Rahmenwerk sollen beliebige Datenmanagement-Funktionen in einen Business Process Execution Language (BPEL)-Prozess eingebunden werden können. Dafür werden bereits vorhandene Konzepte evaluiert, wie z.B. die Sprache BPEL (siehe [5]), und falls nötig erweitert und angepasst. Für eine möglichst hohe Flexibilität soll ein dynamischer Ansatz gewählt werden, so dass auch erst zur Laufzeit des Systems die Datenquellen festgelegt werden können. Nichtsdestotrotz sollte auch die Möglichkeit bestehen, die Datenquellen statisch anbinden zu können. Eine Anforderung des Kunden ist, dass eine vorhandene BPEL-Workflow-Engine sowie ein vorhandenes Modellierungstool um diese gewünschten Funktionen erweitert bzw. angepasst werden. Die BPEL-Prozesse sollen mit dem entsprechenden Modellierungstool spezifiziert und mit der BPEL-Workflow-Engine ausgeführt werden können.

1.3 Evolution des Dokuments

Das Projekt SIMPL ist in drei Iterationsstufen aufgeteilt. Die vorliegende Spezifikation spiegelt die erste Iterationsstufe des Rahmenwerks wieder und stellt dessen Grundfunktionalität dar. Die Spezifikation und damit die Beschreibung der weitergehenden Funktionalität wird in späteren Iterationen vervollständigt.

Darüber hinaus wird in weiterführenden Kundengesprächen über die Präferenzordnung von optionalen Implementierungen diskutiert, welche in den entsprechenden Iterationen umgesetzt werden sollen. In den folgenden Abschnitten (1.3.1 und 1.3.2) sind detaillierte Informationen enthalten, in welche Aufgabenbereiche sich die Basisfunktionalität gliedert und welche Funktionen in späteren Iterationen geplant sind.

1.3.1 Basisfunktionalität

Folgende Funktionen bilden die Basisfunktionalität des SIMPL-Rahmenwerks

- Die statische Anbindung von Datenquellen: Relationale Datenbanken, XML-Datenbanken, Sensornetze und Dateisysteme.
- Eine grundlegende Adminkonsole, die Funktionen wie das An- und Ausschalten des Auditings und die Eingabe globaler Einstellungen für das Rahmenwerk bereitstellt.
- Bereitstellung von generischen BPEL-Aktivitäten im Eclipse BPEL Designer für den Zugriff auf Datenquellen.

1.3.2 Ausblick

In diesem Abschnitt geht es darum, welche Funktionalität in weiteren Iterationen hinzugefügt wird. Die folgende Liste enthält die Punkte, die auf jeden Fall in einer der folgenden Iterationen umgesetzt werden.

- Es wird ein Plug-In System für die Anbindung von verschiedenen Datenquellen umgesetzt.
- Unterstützung von Referenzen in BPEL (siehe [3]), damit auf Daten auch per Referenz im Workflow zugegriffen werden kann. Dies wird aus Gründen der Performanz benötigt, da die Datenübergabe standardmäßig per Value erfolgt, was bei großen Datenmengen (bis zu Gigabytes im wissenschaftlichen Bereich) zu erheblichen Performanzeinbußen führt.
- Bereitstellung einer Datenquellen-Registry mit Hilfe derer Datenquellen über den Eclipse BPEL Designer (siehe [11]) manuell durch den Benutzer oder dynamisch durch das SIMPL Eclipse Plug-In ausgewählt werden können.
- Implementierung eines Dateisystem-Adapters am Beispiel des CSV-Dateiformats.
- Unterstützung einer automatischen Auswahl von Datenquellen zur Laufzeit. Dabei kann der Benutzer Anforderungen an Datenquellen formulieren und eine Strategie auswählen, mit deren Hilfe eine Datenquelle, die seine Anforderungen erfüllt, ausgewählt wird.
- Einstellung der Granularität von Auditing und Monitoring.

Im Folgenden wird beschrieben, was von uns (unter Absprache mit dem Kunden) als optionale Anforderungen definiert wurden. Diese werden umgesetzt, wenn der zeitliche Rahmen des Projekts es zulässt.

- Implementierung eines Monitorings, welches das bestehende Auditing nutzt, um dessen Daten in definierten Abständen auszulesen und dem Benutzer, zur Beobachtung und Überwachung von BPEL-Prozessen, anzuzeigen. Die kann später auch einfach vom Kunden nachgerüstet werden.

1.4 Definitionen

Die in der Spezifikation verwendeten Begriffe, Definitionen und Abkürzungen werden in einem separaten Begriffslexikon eindeutig definiert und erklärt. Dadurch werden Missverständnisse innerhalb des Projektteams oder zwischen Projektteam und Kunde vermieden.

Auf alle in Abschnitt 7.2.3 beschriebenen Aktivitäten wird in diesem Dokument mit dem Sammelbegriff Data-Management-Aktivität verwiesen. Wird also von einer Data-Management-Aktivität gesprochen, ist indirekt eine dieser Aktivitäten gemeint. Über die Definition und Verwendung dieses Sammelbegriffs soll lediglich die Allgemeingültigkeit der getroffenen Aussagen, für jede der in Abschnitt 7.2.3 beschriebenen Aktivitäten, erreicht werden. Nachfolgend wird weiterhin für den Begriff "Data-Management-Aktivität" die Abkürzung DM-Aktivität in diesem Dokument verwendet.

1.5 Überblick

In diesem Dokument soll die zu entwickelnde Software spezifiziert werden. Dazu werden in Kapitel 2 die spätere Systemumgebung, die Kernfunktionen, die Sprache und weitere Aspekte der Software beschrieben. So erhält der Leser einen Überblick über die Funktionalität der Software und deren Verwendung. Weiterhin werden die Ziele und Aufgaben, die für die Realisierung der Software bestehen, aufgezeigt. Anschließend werden die vom Kunden genannten Anforderungen durch die Kapitel 3, 5, 6 und die vorläufige Benutzeroberfläche in Kapitel 4 aufgezeigt. Dabei werden durch die nichtfunktionalen Anforderungen qualitative (Robustheit, Portabilität, usw.) und quantitative (Mengengerüst) Anforderungen an die Software spezifiziert. Die Anwendungsfälle in Kapitel 6 beschreiben die funktionalen Anforderungen. Es werden also konkret die Funktionen, die z.T. schon in der Übersicht der Kernfunktionalität weiter oben aufgeführt sind, die die Software enthalten soll, beschrieben. Im Anschluss folgen in Kapitel 7 die Beschreibungen und Definitionen einiger Konzepte bzw. Ansätze, die zur Umsetzung der gewünschten Funktionalität benötigt werden. Am Ende des Dokuments werden in Kapitel 8 die verwendeten Werkzeuge und Technologien, die für die Erstellung der Software benötigt werden, vorgestellt.

2 Allgemeine Beschreibung

Dieses Kapitel liefert allgemeine Informationen über die zu entwickelnde Software. Dazu gehören beispielsweise die Beschreibung der späteren Systemumgebung, die wichtigsten Funktionen, die verwendete Sprache und Informationen über den Benutzerkreis der Software.

2.1 Einbettung in die Systemumgebung

Das SIMPL Rahmenwerk, bestehend aus dem SIMPL Core und dem SIMPL Eclipse Plug-In soll in die in Abbildung 1 dargestellte Systemumgebung eingebettet werden. Die Systemumgebung besteht dabei aus Eclipse mit dem BPEL-Designer Plug-In, einem Web-Server, wie z.B. dem Apache Tomcat (siehe [8]), auf dem der SIMPL Core und eine Workflow-Engine (z.B. Apache Orchestration Director Engine (ODE), siehe [7]) ausgeführt werden und den Datenquellen auf die zugegriffen wird. SIMPL unterstützt die in Abbildung 1 dargestellten Datenquellen und kann um weitere Datenquellen ergänzt werden. Der SIMPL Core läuft als Web-Service auf dem Web-Server und liefert die Funktionalität, die während der Laufzeit von Prozessen mit Data-Management-Aktivitäten benötigt werden. Das SIMPL Eclipse Plug-In erweitert die grafische Oberfläche von Eclipse und des BPEL-Designer Plug-Ins und liefert die benötigte Funktionalität um Prozesse mit Data-Management-Aktivitäten zu modellieren und Einstellungen für das Rahmenwerk vorzunehmen. Dabei läuft die benötigte Software auf dem lokalen Rechner des Benutzers, die Datenquellen können auf verschiedene Server verteilt sein.

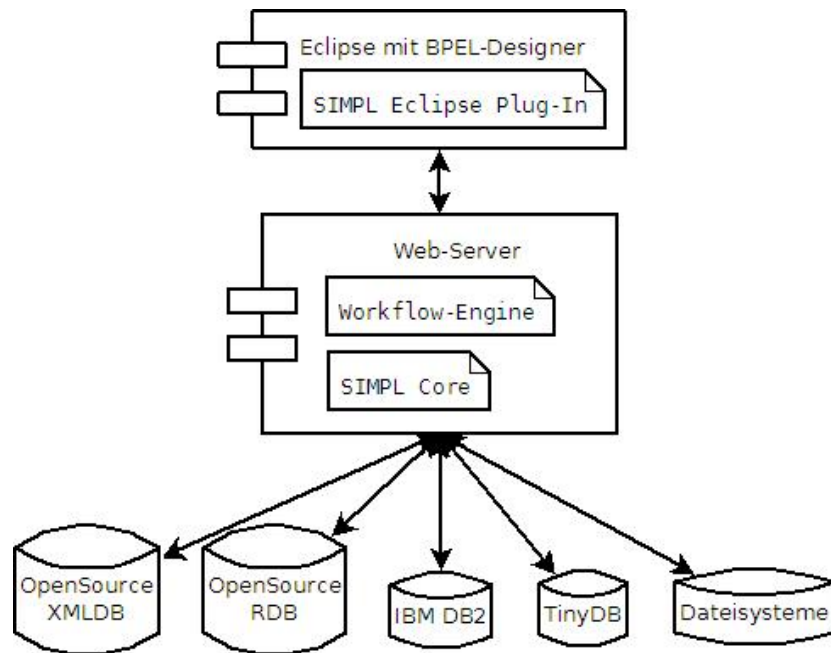


Abbildung 1: Übersicht über die Systemumgebung von SIMPL

2.2 Funktionen

In diesem Abschnitt folgen die wichtigsten Funktionen des Rahmenwerks, die später dessen Kernfunktionalität bilden sollen.

Das Rahmenwerk soll als Eclipse Plug-In verwendet werden und mit der Laufzeitumgebung integriert sein. Es soll die Verarbeitung von großen, heterogenen Datenmengen im Rahmen eines wissenschaftlichen Workflows ermöglichen.

Die vorhandenen BPEL-Aktivitäten des Eclipse BPEL Designer werden dazu um neue Aktivitäten für die Verwaltung von Daten (Data-Management-Aktivitäten) erweitert. Mit deren Hilfe wird die Anbindung von Datenquellen in BPEL-Prozessen vereinfacht. In der ersten Iteration werden die Datenquellen über ihre physikalische Adresse angebunden. In einer späteren Iteration wird eine Datenquellen-Registry bereitgestellt. Diese ermöglicht die Realisierung einer grafischen Auswahlmöglichkeit für Datenquellen im Eclipse BPEL Designer. Eine nähere Beschreibung der Data-Management-Aktivitäten wird in Abschnitt 7.2 gegeben.

Alle Ereignisse, die während der Laufzeit eines Prozesses auftreten (z.B. Zugriff auf eine Datenquelle), werden durch ein Auditing in einer vom Nutzer definierten Datenbank gespeichert.

Für die Verwaltung des SIMPL-Rahmenwerks und zur Änderung von Einstellungen auch während der Laufzeit von Prozessen, wird eine Admin-Konsole implementiert. Über diese kann zum Beispiel das Auditing an und ausgeschaltet und eine Datenbank zur Speicherung der Auditinginformationen angegeben werden.

2.3 Sprache

Generell gilt, dass alle Dokumente auf Deutsch und jeder Quellcode einschließlich Kommentaren auf Englisch verfasst und ausgeliefert werden sollen. Eine Ausnahme bilden das Handbuch und die verschiedenen Dokumentationen der von uns durchgeführten Erweiterungen, wie z.B. die Erweiterungen von Apache ODE oder dem Eclipse BPEL Designer. Diese Dokumente werden auf Deutsch und auf Englisch verfasst, um sie einem breiteren Leserkreis zur Verfügung stellen zu können.

2.4 Distributionsform und Installation

Das Rahmenwerk wird als Teil eines großen Installationspakets ausgeliefert. Dieses Installationspaket besteht aus allen Programmen, die für die Verwendung des Rahmenwerks benötigt werden. Dazu gehört ein Modellierungstool (Eclipse BPEL Designer), ein Web-Server (Apache Tomcat), der eine Workflow-Engine ausführen kann, eine Workflow-Engine (Apache ODE) und natürlich das Rahmenwerk selbst. Mithilfe des Installationspakets ist es möglich viele Einstellungen bereits vorzudefinieren und dem Benutzer die Installation zu erleichtern. Das Installationspaket wird dabei als RAR-Archiv zusammen mit allen wichtigen Dokumenten auf einer CD/DVD ausgeliefert. So können nachträgliche Erweiterungen/Korrekturen des Rahmenwerks mithilfe der Dokumentationen leichter realisiert werden. Die Installation der einzelnen Komponenten wird dann anhand der mitgelieferten Installationsanleitung durchgeführt. Nähere Informationen liefert [4].

2.5 Benutzerprofile

Die Benutzer sind im Normalfall Wissenschaftler und Ingenieure. Sie haben meist keine bis wenig Vorkenntnisse in den Bereichen Workflow und Informatik und stellen so entsprechende Anforderungen an die Benutzbarkeit des Rahmenwerks (siehe Kapitel 3).

2.6 Einschränkungen

Für die Erstellung und Verwendung des SIMPL Rahmenwerks gelten die folgenden Einschränkungen.

- Die Modellierung von BPEL-Prozessen ist an das Modellierungswerkzeug Eclipse BPEL Designer gebunden.
- Das Auditing der Prozessausführung wird nur für die Workflow-Engine Apache ODE bereitgestellt.
- Als Workflow-Modellierungssprache dient die Business Process Execution Language (BPEL).
- Als Programmiersprache für das SIMPL Rahmenwerk kommt Java zum Einsatz.

3 Nichtfunktionale Anforderungen

In diesem Kapitel werden die nichtfunktionalen Anforderungen an die zu entwickelnde Software beschrieben. Dafür werden die entsprechenden Software-Qualitäten aufgeführt und ihre Bedeutung für die zu entwickelnde Software erläutert.

3.1 Mengengerüst

Das Mengengerüst beinhaltet alle quantifizierbaren Anforderungen an das Rahmenwerk.

- Für die Speicherung der Auditing-Daten kann nur eine Datenbank gleichzeitig ausgewählt sein.
- Alle laufenden Prozesse einer Workflow-Engine können zu einem Zeitpunkt gemeinsam nur Daten in Höhe des Speichers, der durch das Betriebssystem zur Verfügung gestellt wird, halten.

3.2 Benutzbarkeit

Die Benutzbarkeit soll sich vor allem an Nutzer mit wenig Kenntnissen im Umgang mit Workflows und BPEL richten und die dafür größtmögliche Transparenz liefern. Dass bedeutet, dass die interne Prozesslogik der Software bestmöglich vom Benutzer abgeschirmt wird. Dadurch erhält der Benutzer eine möglichst einfache und schnell verständliche Schnittstelle zur Software.

3.3 Verfügbarkeit

Die Verfügbarkeit des Rahmenwerks soll mindestens so hoch sein, dass sie der Verfügbarkeit der späteren Systemumgebung entspricht oder diese übersteigt, damit durch die Verwendung des Rahmenwerks keine zusätzlichen Ausfallzeiten entstehen.

3.4 Robustheit

Unter Robustheit ist hier zu verstehen, dass selbst wenn es zu ungünstigen Bedingungen kommt, SIMPL weiterhin weitgehend fehlerfrei verwendet werden kann. Ungünstige Bedingungen sind dabei z.B. der Ausfall des Servers oder Probleme bei der Verbindung mit Datenquellen. Dazu sollen Prozesse fehlerfrei ausgeführt werden und entsprechend korrekte Ergebnisse liefern oder das Rahmenwerk sicher beendet werden können. Nach dem Neustart des Rahmenwerks müssen auch die Prozesse neu gestartet werden.

Benutzereingaben werden nicht vom Rahmenwerk überprüft. Fehlerhafte Eingaben resultieren während dem Deployment bzw. dem Prozessablauf jedoch immer in stabilen Zuständen, die über entsprechende Fehlermeldungen dem Benutzer mitgeteilt werden. Anhand der Fehlermeldungen kann der Benutzer seine Eingaben korrigieren und den Prozess neu deployen bzw. ausführen.

3.5 Sicherheit

Da das Rahmenwerk nur lokal ausgeführt wird und alle lokalen Benutzer momentan die gleichen Rechte besitzen, wird vorerst auf Authentifizierungs- und Autorisierungsmaßnahmen innerhalb des Rahmenwerks verzichtet. Um eine spätere Realisierung zu vereinfachen, werden bereits jetzt die Rollen Prozess-Modellierer und Workflow-Administrator (siehe Kapitel 5) definiert. Die Authentifizierung und Autorisierung bei Datenquellen wird in Abschnitt 7.3 beschrieben.

3.6 Portabilität

Die Portabilität des SIMPL Eclipse Plug-Ins ist durch die Integration in Eclipse gewährleistet. Der SIMPL Core wird dahingehend implementiert, dass er in allen Java unterstützenden Web-Containern lauffähig ist.

3.7 Erweiterbarkeit

Die Erweiterbarkeit des Systems ist eine zentrale Anforderung, da es über einen langen Zeitraum genutzt und in Zukunft um die Anbindung weiterer Datenquellen, Konzepte für den Datenzugriff und den Umgang mit weiteren Datenformaten ergänzt werden soll. Um die Erweiterbarkeit des Systems zu gewährleisten, werden ein modularer Aufbau zugrunde gelegt und entsprechende Schnittstellen geschaffen.

3.8 Wartbarkeit

Durch eine qualitativ hochwertige Dokumentation und ein strukturiertes, geplantes und sauberes Entwicklungsvorgehen soll eine hohe Wartbarkeit erreicht werden. Dazu werden alle Dokumente entsprechend gepflegt und laufend aktualisiert. Weiterhin werden nach jedem Wartungsintervall Tests durchgeführt und deren Ergebnisse protokolliert.

3.9 Skalierbarkeit

Die Skalierbarkeit des Rahmenwerks soll garantieren, dass für die Verarbeitung von steigenden Datenmengen die benötigten Ressourcen höchstens in der gleichen Größenordnung steigen.

4 Benutzeroberfläche des SIMPL Eclipse Plug-Ins

In diesem Kapitel wird die grafische Benutzeroberfläche des SIMPL Eclipse Plug-Ins beschrieben. Abbildung 2 zeigt den erweiterten Eclipse BPEL Designer. In der Palette befinden sich die SIMPL Data-Management-Aktivitäten, die wie bereits vorhandene Aktivitäten zur Modellierung von Prozessen verwendet werden können. Für SIMPL wird ein Menü bereitgestellt, über das alle wichtigen Einstellungen und Informationen des SIMPL Rahmenwerks vorgenommen bzw. angezeigt werden können.

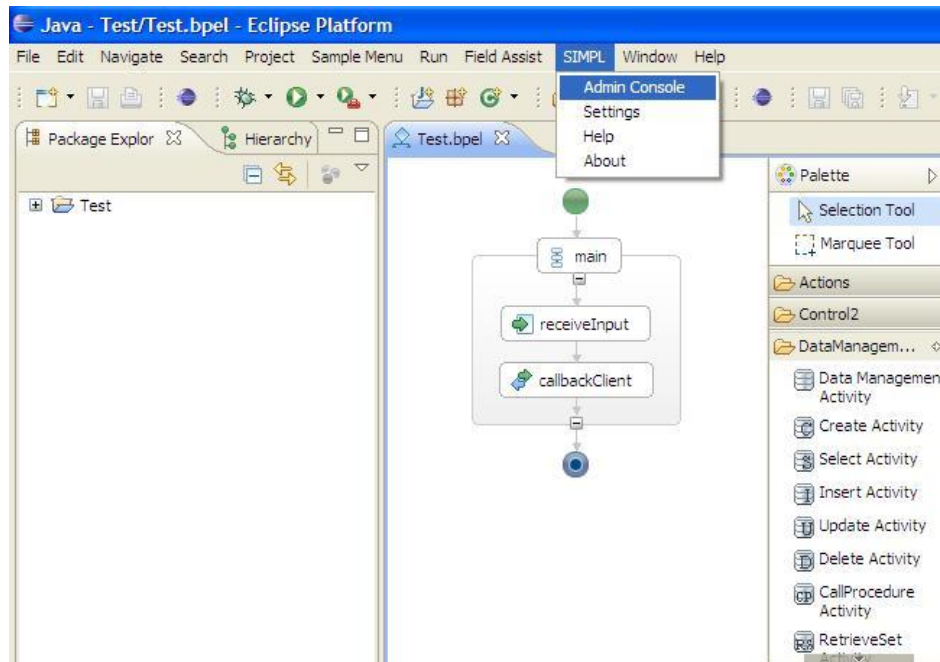


Abbildung 2: SIMPL Menü und Eclipse BPEL Designer mit einigen Data-Management-Aktivitäten

4.1 Admin-Konsole

Die Admin-Konsole kann über das SIMPL Menü geöffnet werden und bietet die Möglichkeit, Einstellungen für das Rahmenwerk vorzunehmen. Dazu gehört beispielsweise die Angabe von globalen Einstellungen und die Verwaltung des Auditings (siehe Abbildung 3).

Folgende Schaltflächen stehen durchgängig zur Verfügung:

- [Default]: Laden der Standard-Einstellungen
- [Reset]: Zurücksetzen aller durchgeführten Änderungen auf den letzten Speicherstand
- [Save]: Speichern aller durchgeführten Änderungen
- [Close]: Schließen der Admin-Konsole und Verwerfen aller Änderungen

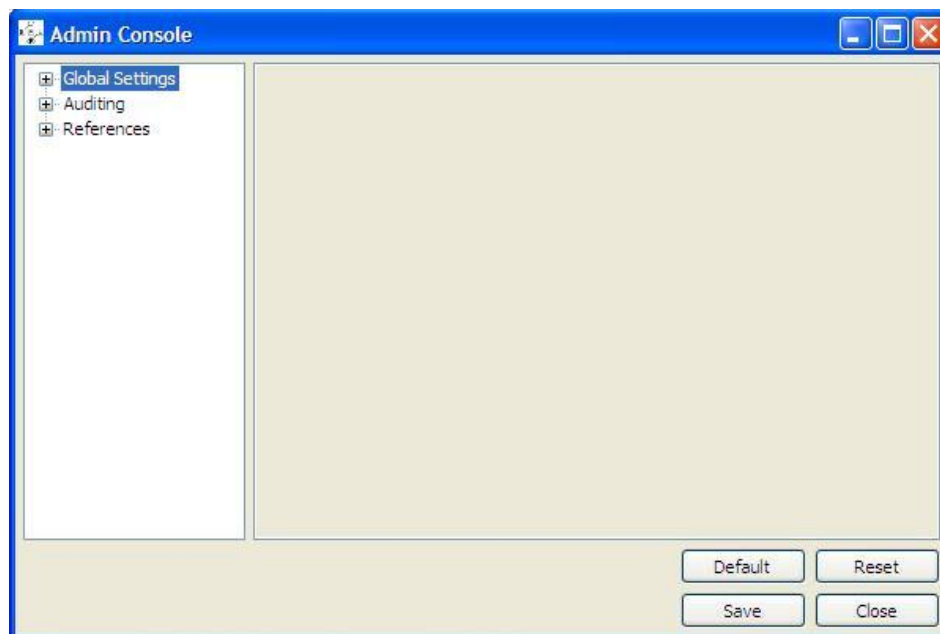


Abbildung 3: Admin-Konsole

4.1.1 Globale Einstellungen

In Abbildung 4 wird der Unterpunkt "Global Settings" der Admin-Konsole gezeigt. Hier können Standardwerte für die Authentifizierung bei Datenquellen angegeben werden.

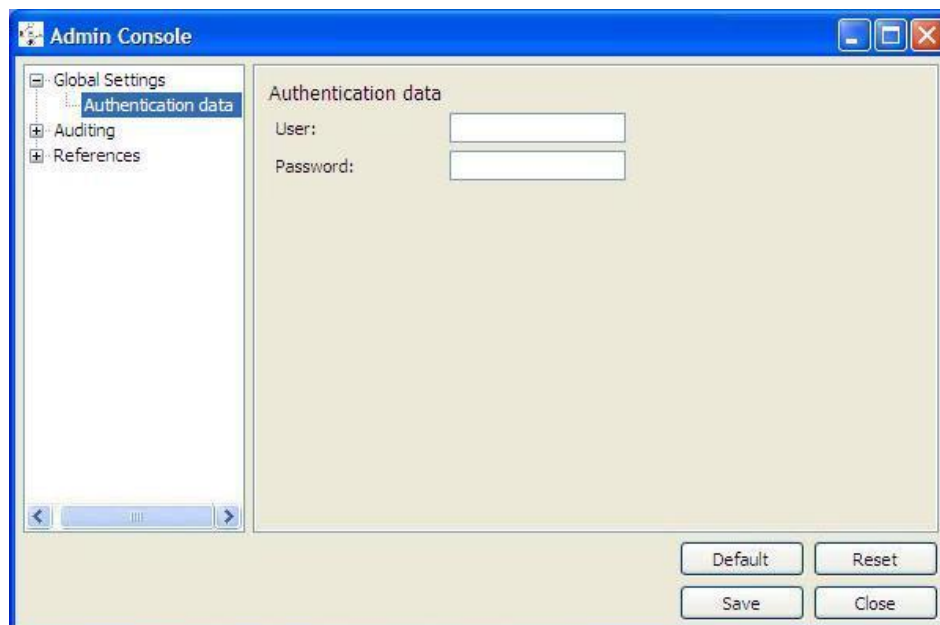


Abbildung 4: Dialog der globalen Eigenschaften

4.1.2 Auditing

In Abbildung 5 wird der Unterpunkt “Auditing” der Admin-Konsole gezeigt. Hier kann das Auditing an- und abgeschaltet werden und eine Datenbank für das Auditing angegeben werden.

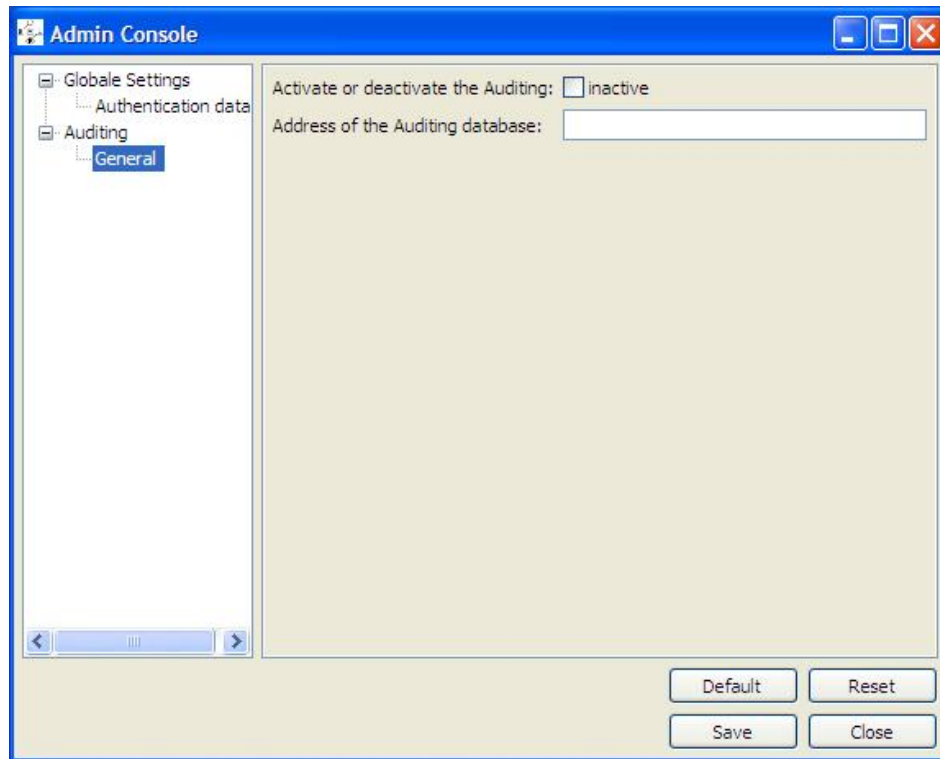


Abbildung 5: Dialog für die Auditing Einstellungen

4.1.3 Referenzen

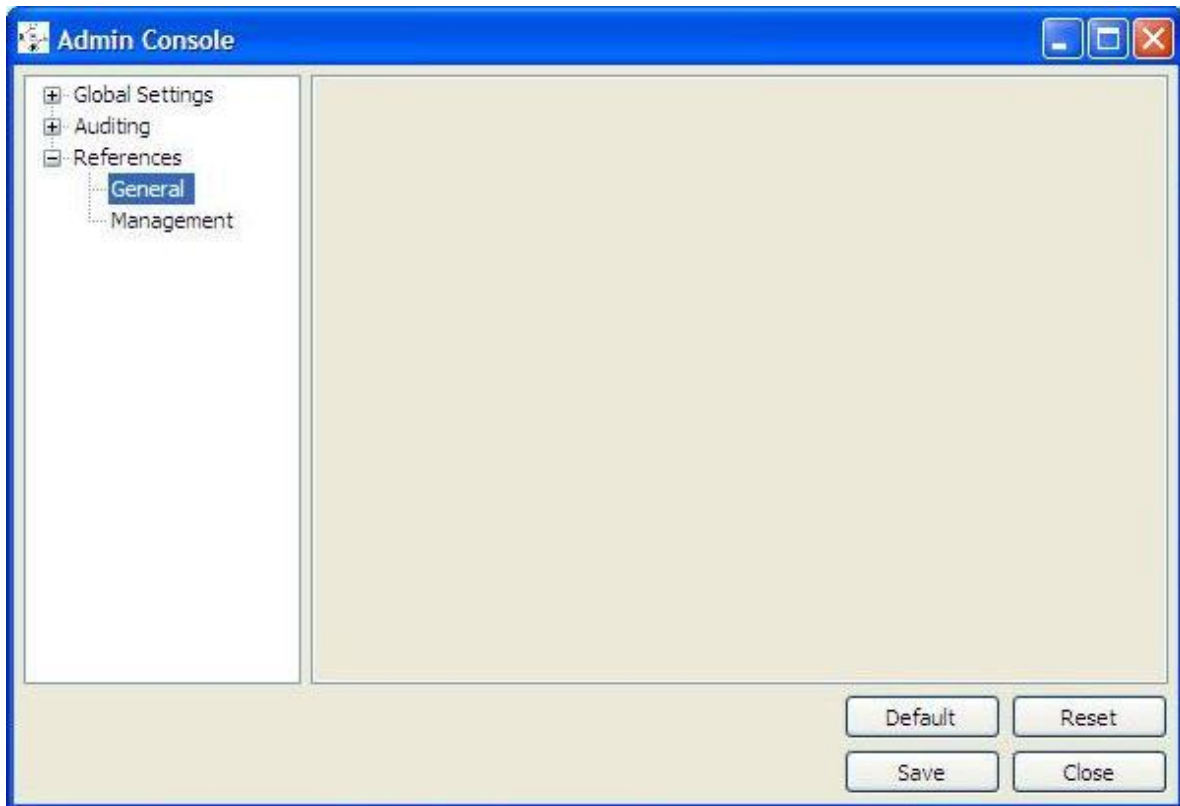


Abbildung 6: Dialog für allgemeine Referenzen Einstellungen

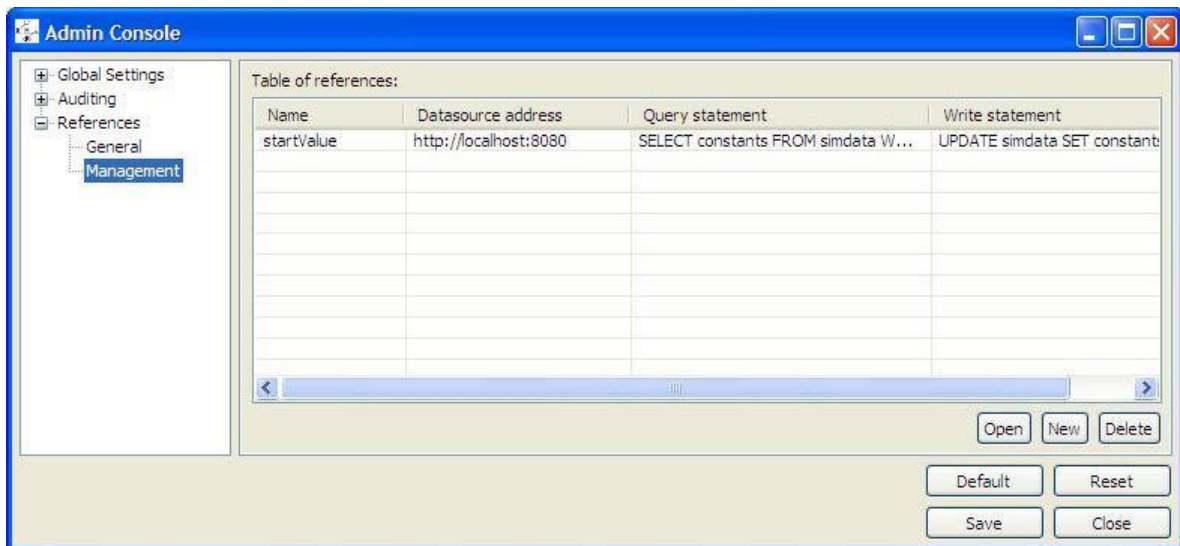
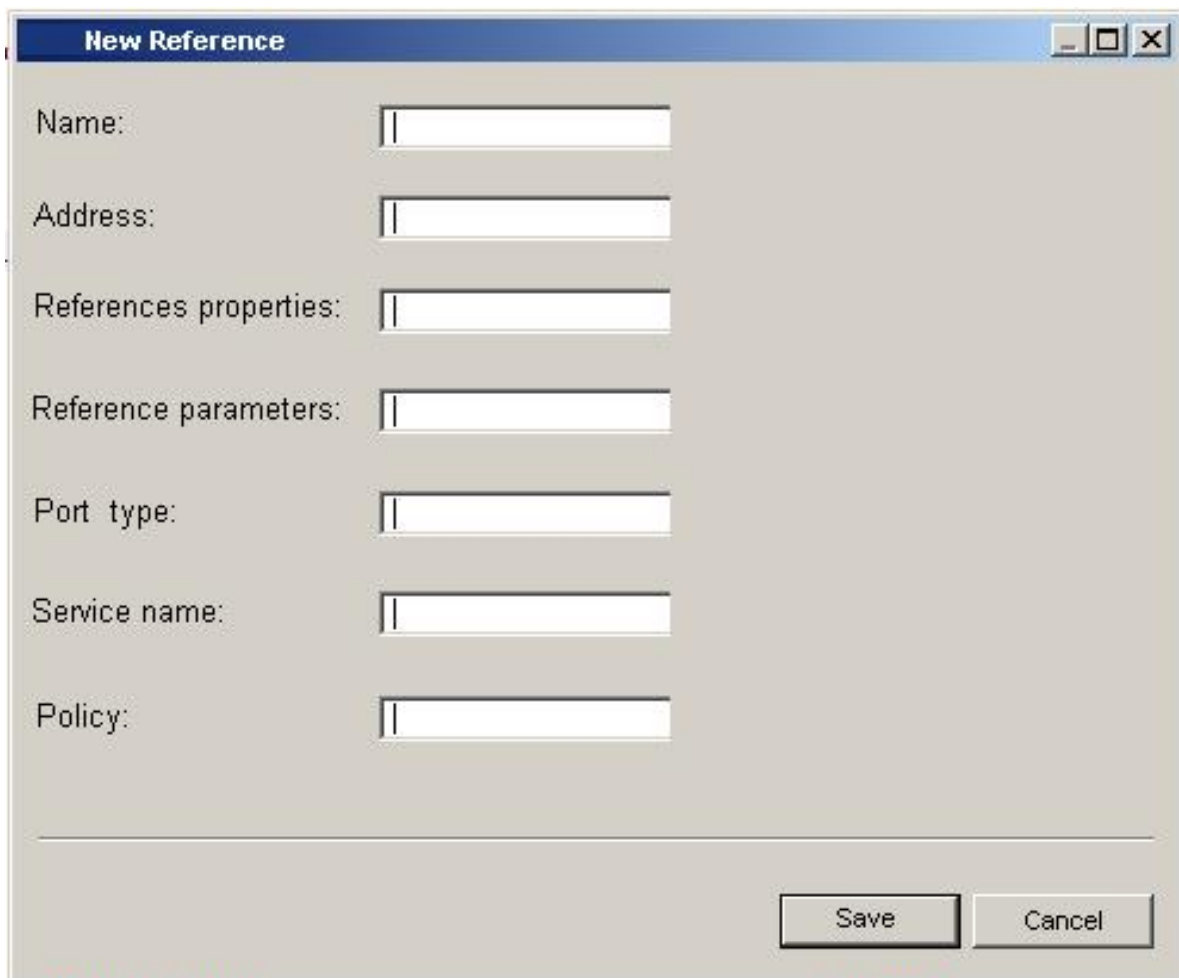


Abbildung 7: Dialog für die Verwaltung von Referenzen



The image shows a Windows-style dialog box titled "New Reference". It has a blue title bar with standard minimize, maximize, and close buttons. The main area is light gray and contains seven labels on the left, each followed by a white text input field: "Name:", "Address:", "References properties:", "Reference parameters:", "Port type:", "Service name:", and "Policy:". At the bottom right, there are two buttons: "Save" and "Cancel".

Field Label	Input Field
Name:	<input type="text"/>
Address:	<input type="text"/>
References properties:	<input type="text"/>
Reference parameters:	<input type="text"/>
Port type:	<input type="text"/>
Service name:	<input type="text"/>
Policy:	<input type="text"/>

Buttons: Save, Cancel

Abbildung 8: Dialog für das Anlegen neuer Referenzen

The image shows a Windows-style dialog box titled "Edit Reference". It contains the following fields and values:

- Name: Reference name
- Address: Example Address
- References properties: (empty)
- Reference parameters: (empty)
- Port type: type A
- Service name: HalloService
- Policy: (empty)

At the bottom right, there are two buttons: "Cancel" and "Save".

Abbildung 9: Dialog zur Bearbeitung von Referenzen

4.2 Data-Management-Aktivitäten

In Abbildung 10 wird die "PropertyView" einer Query-Aktivität gezeigt. Hier kann die im Prozess ausgewählte Aktivität parametrisiert werden. Das bedeutet, dass die Aktivität hier mit Inhalt gefüllt wird, wie z.B. der Zieldatenquelle oder dem Befehl, der auf dieser ausgeführt werden soll. Dazu wird die Art der Datenquelle ausgewählt, ein Befehl über entsprechende grafische Elemente erstellt und die physikalische Adresse der Datenquelle angegeben. Falls die Checkbox "Show resulting statement" gesetzt ist, wird der durch die grafischen Elemente definierte Befehl im Textfeld "Resulting statement" angezeigt. Im Textfeld wird dabei der Befehl in seiner vollen Länge angezeigt, genau so wie er auf der Datenquelle ausgeführt wird.

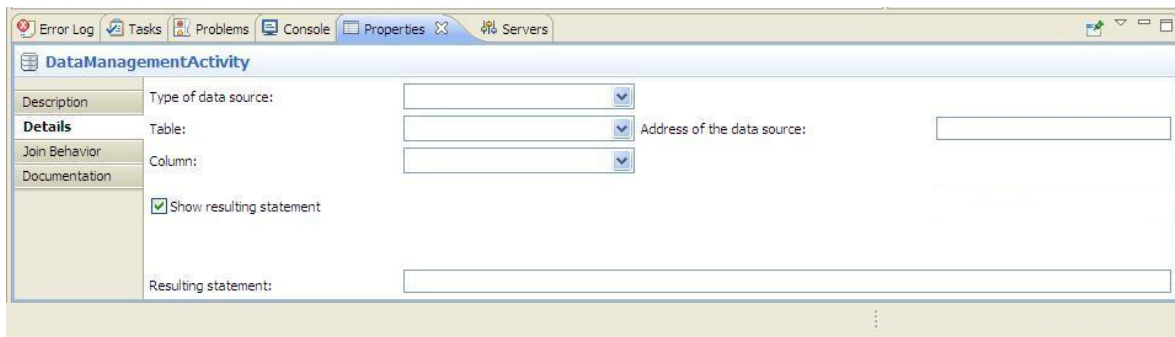


Abbildung 10: Eigenschaftsfenster einer Data-Management-Aktivität am Beispiel einer Query Activity

5 Akteure

In diesem Kapitel werden die einzelnen Akteure der Software beschrieben und ihre Abhängigkeiten untereinander definiert.

5.1 Prozess-Modellierer

Ein Prozess-Modellierer besitzt Fachwissen (z.B. aus der Biologie), das er bei der Modellierung eines wissenschaftlichen Workflows verwendet. Zur Modellierung der Workflows nutzt er den Eclipse BPEL Designer. Dazu kann er beispielsweise BPEL-Aktivitäten erstellen, bearbeiten und auch löschen. Hat der Prozess-Modellierer den BPEL-Prozess fertig modelliert, kann er diesen im Anschluss auf einer Workflow-Engine deployen und ausführen lassen.

5.2 Workflow-Administrator

Ein Workflow-Administrator ist eine Spezialisierung des Prozess-Modellierers, d.h. er kann alle Anwendungsfälle des Prozess-Modellierers und noch weitere administrative Anwendungsfälle ausführen. Seine Kenntnisse liegen im technischen Bereich, wie z.B. bei der Konfiguration des Rahmenwerks. Er kann beispielsweise über die Admin-Konsole während der Prozesslaufzeit das Auditing an- und abschalten. Ebenso legt er die Datenbank für das Speichern der Auditing-Daten fest.

5.3 ODE Workflow-Engine

Die ODE Workflow-Engine ist ein durch Software realisierter Akteur. Sie führt interne Anwendungsfälle aus, die von einem Benutzer durch andere Anwendungsfälle indirekt aufgerufen werden. Durch die Erweiterungen kann sie Data-Management-Aktivitäten ausführen und kompensieren.

5.4 Eclipse BPEL Designer

Der Eclipse BPEL Designer ist ebenfalls ein durch Software realisierter Akteur. Er führt interne Anwendungsfälle aus, die von einem Benutzer durch andere Anwendungsfälle indirekt aufgerufen werden. Der Eclipse BPEL Designer sorgt für das Laden der Einstellungen der Admin-Konsole und das Speichern dieser nach Änderungen.

5.5 SIMPL Core

Der SIMPL Core ist ebenfalls ein durch Software realisierter Akteur. Er führt interne Anwendungsfälle aus, die von einem Benutzer durch andere Anwendungsfälle indirekt aufgerufen werden. Der SIMPL

Core löst dabei die Transformation der modellierten BPEL-Prozesse aus und sorgt für das automatische Auflösen von Referenzen.

6 Anwendungsfälle (Use-Cases)

Dieses Kapitel beschreibt die funktionalen Anforderungen an die Software. Dazu werden alle Anwendungsfälle eines jeden Akteurs beschrieben und deren Zusammenhänge in entsprechenden Diagrammen graphisch dargestellt.

6.1 Diagramm der Anwendungsfälle

Abbildung 11 zeigt das Diagramm aller Anwendungsfälle der gesamten Software. Dadurch sollen die Funktionalität und die Akteure des späteren Gesamtsystems sichtbar werden. Die einzelnen Anwendungsfälle der verschiedenen Akteure werden in den folgenden Abschnitten näher beschrieben.



Abbildung 11: Anwendungsfall-Diagramm des gesamten Softwaresystems

6.2 Anwendungsfälle des Prozess-Modellierers

Ein Prozess-Modellierer kann folgende Anwendungsfälle (siehe Abbildung 12) ausführen:

- Data-Management-Aktivität erstellen
- Data-Management-Aktivität bearbeiten
- Data-Management-Aktivität löschen
- Prozess ausführen

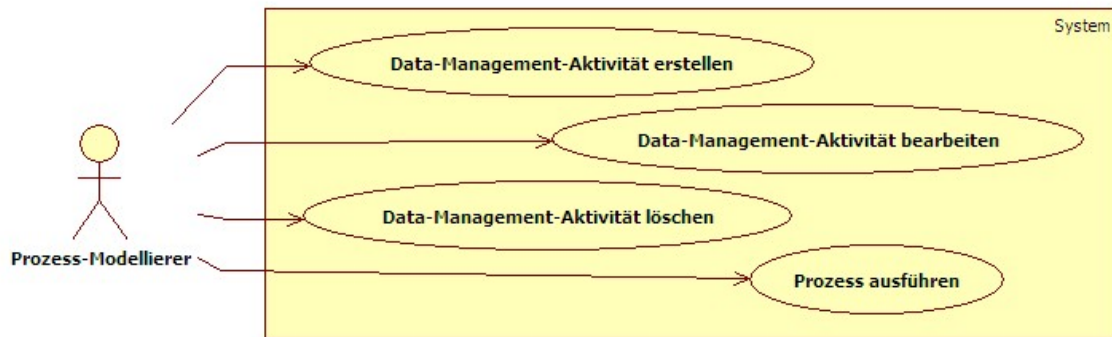


Abbildung 12: Anwendungsfall-Diagramm für den Prozess-Modellierer

6.2.1 Data-Management-Aktivität erstellen

Ziel	Erstellung einer neuen DM-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess ist im Eclipse BPEL Designer geöffnet und die BPEL Designer-Palette wird angezeigt.
Nachbedingung	Die erstellte DM-Aktivität wurde an der selektierten Position in den Prozess eingefügt, und der vom Benutzer eingegebene Name wird angezeigt.
Nachbedingung im Sonderfall	Die erstellte DM-Aktivität wurde an der selektierten Position in den Prozess eingefügt, und der vom Eclipse BPEL Designer vorgeschlagene Name wird angezeigt.
Normalablauf	1a. Selektion einer DM-Aktivität, durch Auswahl mit der linken Maustaste, aus der Palette und anschließend Selektion der Stelle des Prozesses an der die ausgewählte DM-Aktivität eingefügt werden soll 1b. (alternativ) Drag&Drop einer DM-Aktivität aus der Palette an die gewünschte Stelle im Prozess 2. Eingabe eines Aktivitätsnamens durch den Benutzer
Sonderfälle	2a. Der angezeigte Namensvorschlag wird vom Benutzer bestätigt

6.2.2 Data-Management-Aktivität bearbeiten

Ziel	Bearbeitung der Eigenschaften einer vorhandenen DM-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess ist im Eclipse BPEL Designer geöffnet, die zu bearbeitende DM-Aktivität ist ausgewählt und der "Properties-View" von Eclipse wird angezeigt.
Nachbedingung	Alle durchgeführten Änderungen der Eigenschaften wurden korrekt übernommen und werden in der "Properties-View" angezeigt.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Selektierung einer DM-Aktivität, durch Auswahl mit der linken Maustaste, aus dem Prozess 2. Eigenschaften der DM-Aktivität werden innerhalb der "Properties-View" angezeigt 3. Änderung der Eigenschaften
Sonderfälle	keine

6.2.3 Data-Management-Aktivität löschen

Ziel	Löschen der ausgewählten DM-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess ist im Eclipse BPEL Designer geöffnet.
Nachbedingung	Die ausgewählte DM-Aktivität wurde vollständig und korrekt aus dem Prozess gelöscht.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Selektierung einer DM-Aktivität, durch Auswahl mit der linken Maustaste, aus dem Prozess 2. Löschen der DM-Aktivität
Sonderfälle	keine

6.2.4 Prozess ausführen

Ziel	Ausführen des Prozesses auf der Apache ODE Workflow-Engine.
Vorbedingung	Ein Prozess ist im Eclipse BPEL Designer geöffnet, die Apache ODE Workflow-Engine korrekt in Eclipse eingebunden und die Server-View wird angezeigt.
Nachbedingung	Die Prozess-Dateien wurden auf die Apache ODE Workflow-Engine kopiert, der Prozess wurde erfolgreich deployed und wird ausgeführt.
Nachbedingung im Sonderfall	3a. Der Prozess wurde nicht gestartet. 3b. Der Prozess wurde nicht gestartet. 3c. Der Prozess wurde beendet und es wird eine Fehlermeldung angezeigt.
Normalablauf	1. Erstellung eines ODE Deployment-Deskriptors über File->New->Other->BPEL2.0->Apache ODE Deployment Descriptor 2. Hinzufügen des Prozesses zum ODE-Server in der Eclipse Server-View: <ul style="list-style-type: none">• Rechter Mausklick auf den ODE-Server• Auswahl des Menüpunkts "Add and Remove"• Hinzufügen der BPEL Prozess-Datei 3. Starten des ODE-Servers über rechten Mausklick und "Start"
Sonderfälle	3a. ODE-Server startet nicht 3b. Beim Starten des Prozesses tritt ein Fehler auf 3c. Während der Ausführung des Prozesses tritt ein Fehler auf

6.3 Anwendungsfälle des Workflow-Administrators

Ein Workflow-Administrator kann folgende Anwendungsfälle (siehe Abbildung 13) ausführen:

- Admin-Konsole öffnen
- Admin-Konsole zurücksetzen
- Admin-Konsole speichern
- Admin-Konsole schließen
- Admin-Konsole Defaults laden

- Auditing aktivieren
- Auditing deaktivieren
- Auditing-Datenbank festlegen
- Referenz in RRS einfügen
- Referenz aus RRS bearbeiten
- Referenz aus RRS löschen

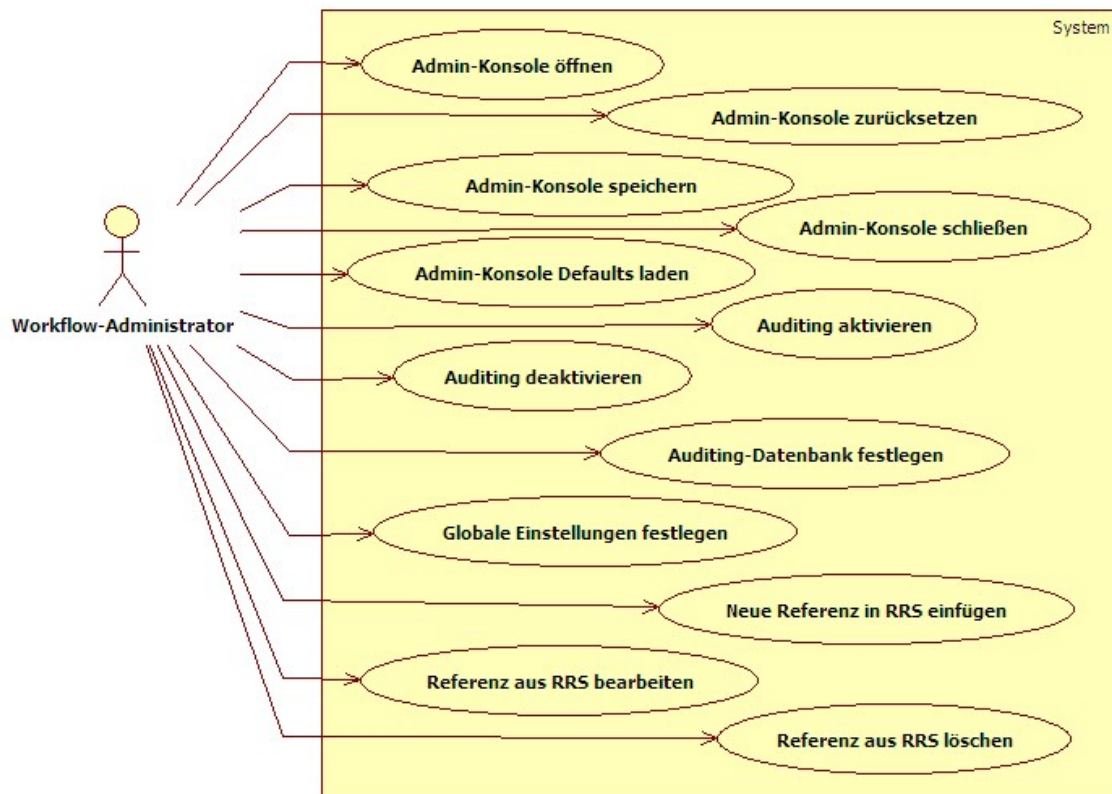


Abbildung 13: Anwendungsfall-Diagramm für den Workflow-Administrator

6.3.1 Admin-Konsole öffnen

Ziel	Öffnen der Admin-Konsole.
Vorbedingung	Eclipse mit dem SIMPL Eclipse Plug-In ist geöffnet.
Nachbedingung	Die Admin-Konsole wird angezeigt und kann verwendet werden.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Klick auf das SIMPL Menü in der Menüleiste 2. Klick auf den Menüeintrag [Admin Console]
Sonderfälle	keine

6.3.2 Admin-Konsole zurücksetzen

Ziel	Zurücksetzen des Inhalts der Admin-Konsole.
Vorbedingung	Die Admin-Konsole wird angezeigt und es wurde mindestens ein Wert geändert.
Nachbedingung	Alle geänderten Werte der Admin-Konsole wurden auf die zuletzt gespeicherten Einstellungen zurückgesetzt.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Klick auf den Button [Reset] 2. Ausführung des Anwendungsfalls "Admin-Konsole laden".
Sonderfälle	keine

6.3.3 Admin-Konsole speichern

Ziel	Speicherung des Inhalts der Admin-Konsole in eine Datenbank.
Vorbedingung	Die Admin-Konsole wird angezeigt.
Nachbedingung	Alle Werte der Admin-Konsole wurden korrekt gespeichert und alle veralteten Werte mit Neuen überschrieben.
Nachbedingung im Sonderfall	1a. Alle geänderten Werte der Admin-Konsole bleiben unverändert. Es wird eine entsprechende Fehlermeldung angezeigt und ein erneuter Speicherversuch kann durchgeführt werden. Die Werte bleiben dabei solange erhalten, bis die Admin-Konsole geschlossen wird. 1b.&1c. Der Benutzer erhält eine Fehlermeldung, die ihn über den entsprechenden Fehler informiert.
Normalablauf	1. Klick auf den Button [Save]
Sonderfälle	1a. Beim Speichern der Werte tritt ein Fehler auf 1b. Vom Benutzer wurde keine Auditing-Datenbank zum Speichern der Auditing-Daten festgelegt. 1c. Die angegebene Auditing-Datenbank kann nicht verwendet werden, da sie z.B. nicht erreichbar ist oder die Authentifizierung fehlgeschlagen ist

6.3.4 Admin-Konsole schließen

Ziel	Schließen der Admin-Konsole.
Vorbedingung	Die Admin-Konsole wird angezeigt.
Nachbedingung	Die Admin-Konsole wurde geschlossen und alle nicht gespeicherten Änderungen wurden verworfen.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Klick auf den Button [Close]
Sonderfälle	keine

6.3.5 Admin-Konsole Defaults laden

Ziel	Laden der Standardwerte in der Admin-Konsole.
Vorbedingung	Die Admin-Konsole wird angezeigt.
Nachbedingung	Alle geänderten Werte der Admin-Konsole wurden auf die gespeicherten Standardwerte zurückgesetzt.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Klick auf den Button [Default] 2. Ausführung des Anwendungsfalls "Admin-Konsole laden".
Sonderfälle	keine

6.3.6 Auditing aktivieren

Ziel	Auditing von SIMPL aktivieren.
Vorbedingung	Die Admin-Konsole ist geöffnet, der Unterpunkt "Auditing" wird angezeigt und das Auditing ist nicht aktiv.
Nachbedingung	Das Auditing ist aktiv.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Setzen der Auditing-Checkbox 2. Ausführung des Anwendungsfalls "Admin-Konsole speichern"
Sonderfälle	keine

6.3.7 Auditing deaktivieren

Ziel	Auditing von SIMPL deaktivieren.
Vorbedingung	Die Admin-Konsole ist geöffnet, der Unterpunkt "Auditing" wird angezeigt und das Auditing ist aktiv.
Nachbedingung	Das Auditing ist deaktiviert.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Zurücksetzen der Auditing-Checkbox
Sonderfälle	keine

6.3.8 Auditing-Datenbank festlegen

Ziel	Festlegung einer Datenbank für das Auditing.
Vorbedingung	Die Admin-Konsole ist geöffnet und der Unterpunkt "Auditing" wird angezeigt.
Nachbedingung	Die Datenbank für das Auditing wurde festgelegt, gespeichert und kann verwendet werden.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Angabe einer Datenbank-URL (optional Auswahl über Datenbank-Registry) 2. Ausführung des Anwendungsfalls "Admin-Konsole speichern"
Sonderfälle	keine

6.3.9 Globale Einstellungen festlegen

Ziel	Festlegung der globalen Einstellungen.
Vorbedingung	Die Admin-Konsole ist geöffnet und der Unterpunkt "Global Settings" wird angezeigt.
Nachbedingung	Die globalen Einstellungen wurden festgelegt und gespeichert.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Eingabe/Änderung der Werte 2. Ausführung des Anwendungsfalls "Admin-Konsole speichern"
Sonderfälle	keine

6.3.10 Neue Referenz in RRS einfügen

Ziel	Referenz in Reference Resolution System (RRS) einfügen.
Vorbedingung	Das RRS ist angebunden und erreichbar.
Nachbedingung	Die neue Referenz wurde korrekt in das RRS eingefügt und kann nun verwendet werden.
Nachbedingung im Sonderfall	Die Referenz wurde nicht in das RRS eingefügt und es wird ein entsprechender Fehlerdialog angezeigt, der den Benutzer über die aufgetretenen Fehler informiert.
Normalablauf	1. Erstellung einer neuen Referenz 2. Eingeben aller benötigten Parameter 3. Referenz einfügen
Sonderfälle	3a. Fehler beim Einfügen der Referenz in das RRS

6.3.11 Referenz aus RRS bearbeiten

Ziel	Bearbeiten einer vorhandenen Referenz des RRS.
Vorbedingung	Das RRS ist angebunden und erreichbar.
Nachbedingung	Die geänderten Werte der Referenz wurden korrekt im RRS gespeichert und die Referenz wurde aktualisiert.
Nachbedingung im Sonderfall	Die lokal veränderte Referenz bleibt im RRS unverändert und es wird ein entsprechender Fehlerdialog angezeigt, der den Benutzer über die aufgetretenen Fehler informiert.
Normalablauf	1. Die zu bearbeitende Referenz auswählen 2. Werte der Referenz ändern 3. Änderungen im RRS speichern
Sonderfälle	3a. Beim Speichern der Änderungen tritt ein Fehler auf

6.3.12 Referenz aus RRS löschen

Ziel	Löschen einer vorhandenen Referenz aus dem RRS.
Vorbedingung	Das RRS ist angebunden und erreichbar.
Nachbedingung	Die entsprechende Referenz wurde vollständig und korrekt aus dem RRS entfernt und kann nun nicht mehr verwendet werden.
Nachbedingung im Sonderfall	Die zu löschende Referenz bleibt im RRS unverändert und es wird ein entsprechender Fehlerdialog angezeigt, der den Benutzer über die aufgetretenen Fehler informiert.
Normalablauf	1. Die zu entfernende Referenz auswählen 2. Referenz aus RRS löschen
Sonderfälle	2a. Beim Löschen der Referenz tritt ein Fehler auf

6.4 Anwendungsfälle der ODE Workflow-Engine

Die ODE Workflow-Engine kann durch unsere Erweiterungen folgende Anwendungsfälle (siehe Abbildung 14) ausführen:

- Data-Management-Aktivität ausführen
- Data-Management-Aktivität kompensieren

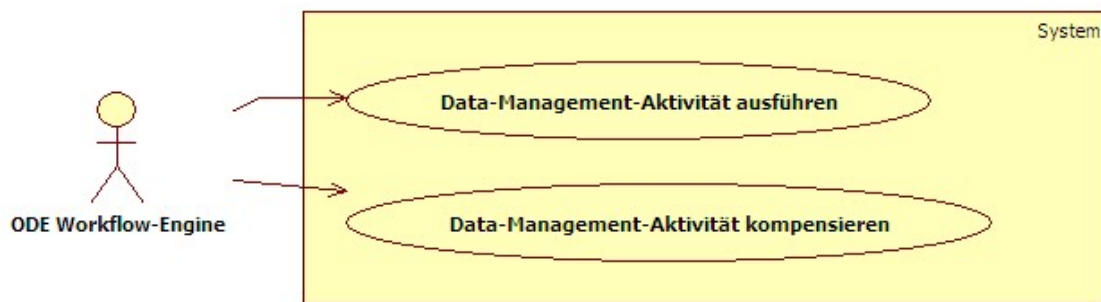


Abbildung 14: Anwendungsfall-Diagramm für die ODE Workflow-Engine

6.4.1 Data-Management-Aktivität ausführen

Ziel	Ausführen einer DM-Aktivität.
Vorbedingung	Es wurde ein Prozess, der eine DM-Aktivität enthält, deployt und es wurde eine Instanz des Prozesses erzeugt.
Nachbedingung	Die DM-Aktivität und die darin enthaltene Datenmanagementoperation wurden erfolgreich ausgeführt.
Nachbedingung im Sonderfall	1a. Die Aktivität befindet sich im Endzustand "Terminated" 1b. Die Aktivität befindet sich im Zustand "Waiting" 2. Die Aktivität befindet sich im Endzustand "Terminated" 3. Die Aktivität befindet sich im Endzustand "Faulted" 4. Die Aktivität befindet sich im Endzustand "Terminated"
Normalablauf	1. Die DM-Aktivität befindet sich im Zustand "Ready" 2. Die Ausführung der DM-Aktivität wird gestartet 3. Senden der in der DM-Aktivität enthaltenen DM-Operationen zur Ausführung an die Datenquelle 4. Die Ausführung der DM-Aktivität ist beendet und sie befindet sich im Zustand "Waiting" 5. Die DM-Aktivität wird als "Complete" gekennzeichnet
Sonderfälle	1a.1. In der Vateraktivität tritt ein Fehler auf 1a.2. Es wird ein "Terminate_Activity Event" ausgelöst 1b.1. Es wird ein "Complete_Activity Event" an die DM-Aktivität gesendet 1b.2. Es wird ein "Activity_Executed Event" ausgelöst 2a.1. In der Vateraktivität tritt ein Fehler auf 2a.2. Es wird ein "Terminate_Activity Event" ausgelöst 3a.1. Während der Durchführung der DM-Operationen tritt ein Fehler in der Datenquelle auf (z.B. ein Syntaxfehler oder die Datenbank ist nicht erreichbar) 3a.2. Es wird ein "Activity_Faulted Event" ausgelöst 4a.1. In der Vateraktivität tritt ein Fehler auf 4a.2. Es wird ein "Terminate_Activity Event" ausgelöst

6.4.2 Data-Management-Aktivität kompensieren

Ziel	Rückgängig machen einer DM-Aktivität, so dass der Zustand der Datenquelle vor Ausführung der DM-Aktivität wiederhergestellt wird.
Vorbedingung	Im Normalablauf einer DM-Aktivität tritt ein Fehler auf (siehe Sonderfälle des Anwendungsfalls "Data-Management-Aktivität ausführen").
Nachbedingung	Der Zustand vor Ausführung der DM-Aktivität wurde erfolgreich wiederhergestellt.
Nachbedingung im Sonderfall	keine
Normalablauf	1. Alle Änderungen werden zurückgesetzt
Sonderfälle	keine

6.5 Anwendungsfälle des Eclipse BPEL Designers

Der Eclipse BPEL Designer kann durch unsere Erweiterungen folgende Anwendungsfälle (siehe Abbildung 15) ausführen:

- Admin-Konsole laden

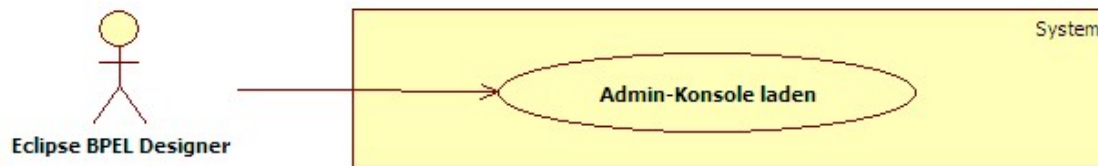


Abbildung 15: Anwendungsfall-Diagramm für den Eclipse BPEL Designer

6.5.1 Admin-Konsole laden

Ziel	Laden der Inhalte der Admin-Konsole.
Vorbedingung	Eclipse mit dem SIMPL Eclipse Plug-In ist geöffnet.
Nachbedingung	Alle Werte der Datei mit den Einstellungen der Admin-Konsole wurden geladen und können im Fenster “Admin Console” und dessen Unterfenstern angezeigt werden.
Nachbedingung im Sonderfall	Es wird eine entsprechende Fehlermeldung angezeigt, die hinterlegten Standard-Einstellungen wurden geladen und werden im Fenster “Admin Console” und dessen Unterfenstern angezeigt.
Normalablauf	1. Laden der Werte aus einer Datenquelle 2. Füllen der Felder der Admin-Konsole
Sonderfälle	1a. Beim Laden der Werte tritt ein Fehler auf und die Standardwerte werden geladen

6.6 Anwendungsfälle des SIMPL Core

Der SIMPL Core kann folgende Anwendungsfälle (siehe Abbildung 16) ausführen:

- BPEL-Datei transformieren
- Neues Datenquellen-Plug-In in RRS einfügen
- Referenz auflösen/dereferenzieren

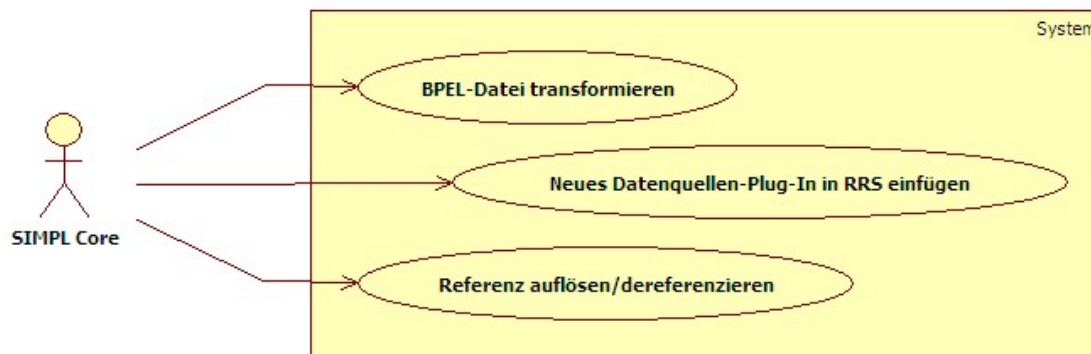


Abbildung 16: Anwendungsfall-Diagramm für die Prozess-Modelierer

6.6.1 BPEL-Datei transformieren

Ziel	Umwandlung einer BPEL-Datei in Standard-BPEL-Quellcode.
Vorbedingung	Eine BPEL-Datei (BPEL-Prozess) muss vorhanden sein und das Reference Resolution System (RRS) muss angebunden und erreichbar sein.
Nachbedingung	Es wurde ein korrekter standard-konformer BPEL-Prozess, der die gleiche Kern-Funktionalität wie der originale Prozess besitzt und zur Verwendung von Referenzen um entsprechende Aktivitäten oder Events erweitert wurde, erzeugt.
Nachbedingung im Sonderfall	Es wurde kein neuer BPEL-Prozess erzeugt, der originale BPEL-Prozess ist unverändert und es wird eine entsprechende Fehlermeldung ausgegeben.
Normalablauf	1. Auslesen des originalen Prozesses 2. Positioniertes Einfügen entsprechender Dereferenzierungsaktivitäten oder Events für Referenzen (siehe 7.1.2) 3. Generierung des neuen BPEL-standard-konformen erweiterten Prozesses
Sonderfälle	3a. Bei der Generierung tritt ein Fehler auf

6.6.2 Neues Datenquellen-Plug-In in RRS einfügen

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

6.6.3 Referenz auflösen/dereferenzieren

Ziel	Der zugrundeliegende Wert einer Referenz wird ausgelesen.
Vorbedingung	Die angegebene Referenz muss existieren und das RRS muss angebunden und erreichbar sein. Ebenso muss die Quelle des referenzierten Werts erreichbar sein.
Nachbedingung	Der korrekte Wert einer Referenz wird geliefert.
Nachbedingung im Sonderfall	
Normalablauf	1. Wert der Referenz wird im RRS abgefragt
Sonderfälle	

7 Konzepte und Realisierungen

In diesem Kapitel werden alle Konzepte, die für SIMPL benötigt werden, und deren Realisierung erläutert. Dazu zählt z.B. die Beschreibung eines Referenzen-Konzepts für BPEL, dass es ermöglicht mit Referenzen innerhalb von Workflows zu arbeiten oder die Identifizierung und Definition von benötigten Data-Management-Aktivitäten, die zur Realisierung einer generischen Unterstützung von verschiedenen Abfragesprachen, wie z.B. SQL oder XQuery, für BPEL benötigt werden. Ebenso werden die Realisierung des Datenquellen-Auditing und das dafür zugrunde liegende Event-Modell beschrieben.

7.1 Referenzen in BPEL

Da bereits ein Konzept und umfassende Erkenntnisse zur Bereitstellung und Verwaltung von Referenzen in BPEL, durch [3] (*“Towards Reference Passing in Web Service and Workflow-based Applications“*) vorlag, wurden an dieser Stelle nur die Ergebnisse des Dokuments zusammengefasst und an die Anforderungen dieses Projekts angepasst.

In BPEL werden Daten immer in Variablen gespeichert und die Werte (*by value*) innerhalb des Workflows weitergegeben. Dies hat zur Folge, dass gerade bei großen Datenmengen, wie sie in wissenschaftlichen Workflows normal sind, die Performanz stark durch das Transportieren der Daten beeinflusst wird. Um dies zu verhindern sollen Referenzen eingeführt werden, diese werden dann zwischen den einzelnen Web-Services weitergegeben (Daten werden *by reference* übergeben) und die Daten bleiben, sofern sie nicht im Workflow benötigt werden, auf ihrer Datenquelle und werden dort auch bearbeitet.

Zur Umsetzung dieses Konzepts muss ein neue Art von BPEL Variablen eingeführt werden, dies sind sogenannte Referenz-Variablen, die dazu genutzt werden können auf Daten zu verweisen. Dadurch müssen nur noch Referenzen zwischen den Web-Services und Workflows weitergeleitet werden und nicht mehr die Daten selbst. Für wissenschaftliche Workflows reduziert sich dadurch der Datentransport von großen Datenmengen zwischen den Web-Services und dem Workflow erheblich.

Zur Bereitstellung solcher Referenzen für Web-Services und Workflows müssen diese extern verwaltet werden und auch extern abrufbar sein. So kann ein globales oder unternehmensweites Variablenkonzept erstellt werden, dass es ermöglicht dieselben Daten in mehreren Workflows zu nutzen und nur einmal zentral zur Verfügung zu stellen.

Die Umsetzung der Referenzen wird durch sogenannte Endpoint References realisiert, die in Abschnitt 7.1.2 beschrieben werden. Das externe Verwaltungssystem der Referenzen, das sogenannte Reference Resolution System, wird in Abschnitt 7.1.1 näher erläutert.

7.1.1 Reference Resolution System

Abbildung 17 zeigt die Architektur des Reference Resolution Systems (RRS). Am unteren Ende der Abbildung werden die verschiedenen möglichen Nutzer des Systems gezeigt: Workflows oder Web-Services, die eingangs nur die Rechte haben, um den Wert einer Referenz auszulesen und ein menschlicher Nutzer, der z.B. ein Administrator ist und damit die Rechte besitzt Referenzen im System anzulegen, zu aktualisieren und zu löschen. Er kann auch anderen Nutzern wie Workflows und Web-Services Rechte erteilen, damit diese auch Referenzen anlegen, aktualisieren oder löschen können.

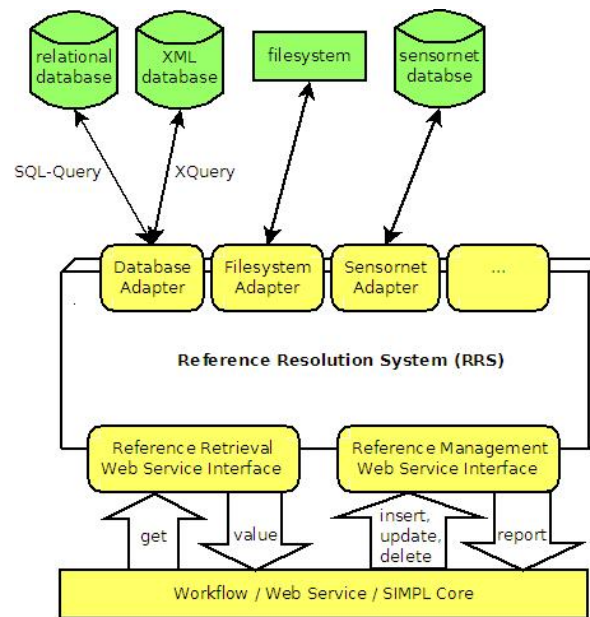


Abbildung 17: Die Architektur des Reference Resolution Systems (RRS).

Für jeden der verschiedenen Nutzer wird eine entsprechende Web-Service Schnittstelle bereitgestellt:

- Die erste Schnittstelle ist der *Reference Retrieval Web Service* mit der Methode *get* und einer *endpoint reference* (EPR) als Eingabe. Das Konzept der endpoint references wird im Web Services Addressing Standard (siehe [5]) beschrieben und kann ohne Anpassungen für die Repräsentation einer Referenz genutzt werden. Der Rückgabewert der Methode ist der Wert, der durch die angegebene EPR referenziert wurde.
- Die zweite Schnittstelle ist der *Reference Management Web Service*, der drei Methoden bereitstellt. Die Methode *insert* um eine neue Referenz zu erstellen, dabei ist die Eingabe der Wert der Referenz, wo dieser Wert gespeichert wird und wie er dort wieder ausgelesen werden kann. Dies könnte beispielsweise durch entsprechende SQL-Befehle angegeben werden. Der Rückgabewert ist eine Meldung, die die neu generierte EPR enthält. Als Zweites die *update* Methode, die für die Aktualisierung von gespeicherten Werten einer Referenz benötigt wird. Dabei ist die Eingabe der neue Wert der Referenz und wie dieser Wert gespeichert wird. Der Rückgabewert ist eine Meldung, ob die Aktualisierung erfolgreich war. Die *delete* Methode löscht eine Referenz aus dem RRS, d.h. der Wert und alle sonstigen Informationen der Referenz werden aus dem System entfernt. Die Eingabe ist dabei die EPR, die entfernt werden soll und der Rückgabewert ist eine Meldung, ob die Referenz erfolgreich entfernt wurde.

Die Hauptkomponente ist allerdings das Reference Resolution System selbst. Es verbindet beide Web-Service Schnittstellen und hält eine Menge von Adaptern für die Integration verschiedenster Datenquellen bereit. Jeder dieser Adapter besteht dabei aus einem Lookup-Service für gespeicherte Queries und Informationen, sowie aus einem ausführenden Service mit dem Queries auf den Datenquellen ausgeführt werden können. Anhand einer gegebenen Referenz wird dazu der passende Adapter gesucht, ausgewählt und für das Auflösen der Referenzen in Werte verwendet. Abbildung 17 zeigt am oberen Ende einige solcher Adapter, wobei das RRS erweiterbar ist und mit Adaptern für alle Arten von Datenquellen ergänzt werden kann.

In [3] werden mehrere mögliche Anbindungen des RRS in der Workflow-Umgebung genannt. Im Zusammenhang mit unseren Gegebenheiten, vorallem im Hinblick darauf, dass die Workflow-Engine

lokal auf dem Benutzer-Rechner ausgeführt wird, bietet sich die Variante “Ein RRS pro Workflow-Engine” an. Dadurch erhalten wir mehrere wichtige Vorteile:

- Jeder Prozess-Modellierer (Workflow-Administrator) kann Referenzen nach Bedarf selbst sofort erstellen und verwalten.
- Die Referenzen können nur lokal geändert werden und sind nicht global (von aussen) zugänglich, dadurch benötigen wir keinen Zugriffsschutz.
- Wird bei der Modellierung von BPEL-Prozessen festgestellt, dass entsprechende Referenzen benötigt werden, können diese sofort angelegt und auch sofort in der Prozess-Modellierung verwendet werden.

7.1.2 Referenzen und Endpoint References

Nachdem die Architektur des Systems beschrieben wurde, folgt die Beschreibung des wichtigsten Teils des Systems, die Referenzen selbst. Dazu wird die Repräsentation der Referenzen anhand des WSAddressing Konzepts der *endpoint references* (EPR's) [5] beschrieben. Das Ziel bei der Definition der Referenzen ist diese so flexibel und erweiterbar wie möglich zu realisieren und zu versuchen die vollständige Abgeschlossenheit der EPR's zu erreichen. Vollständige Abgeschlossenheit bedeutet dabei, dass alle für die Auflösung einer Referenz benötigten Informationen in der EPR hinterlegt sind. Listing 1 zeigt das Schema einer EPR und wie diese zur Repräsentation einer Referenz genutzt werden. Im Schema und allen nachfolgenden Listings werden dabei folgende BNF Konventionen verwendet: “?” bezeichnet Optionalität (0 oder 1), “*” (0 oder mehr), “+” (1 oder mehr) und “|” steht für eine Auswahlmöglichkeit.

```
<wsa:EndpointReference>
  <wsa:Address>
    xs:anyURI
  </wsa:Address>
  <wsa:ReferenceProperties>
    <rrs:resolutionSystem>
      (xs:String | xs:anyURI |
       xs:QName)
    </rrs:resolutionSystem>
  </wsa:ReferenceProperties>
  <wsa:ReferenceParameters>
    (xs:anyURI |
     xs:any)
  </wsa:ReferenceParameters>
  <wsa:PortType>xs:QName</wsa:PortType>
  <wsa:ServiceName PortName="xs:NCName"?>
    xs:QName
  </wsa:ServiceName>
  <wsp:Policy>Policy</wsp:Policy?>
</wsa:EndpointReference>
```

Listing 1: EPR Schema

Die verschiedenen Bestandteile einer EPR werden dabei für folgende Zwecke benötigt: Der **Address** Teil verweist auf den Endpunkt des RRS, das die Werte der Referenzen verwaltet. In den **ReferenceProperties** wird der Adapter des RRS, der für das Auflösen der Referenz zuständig ist, angegeben. Alle Informationen, die der entsprechende Adapter zur Auflösung der Referenz benötigt, werden in den **ReferenceParameters** angegeben. Diese Informationen können entweder direkt angegeben werden,

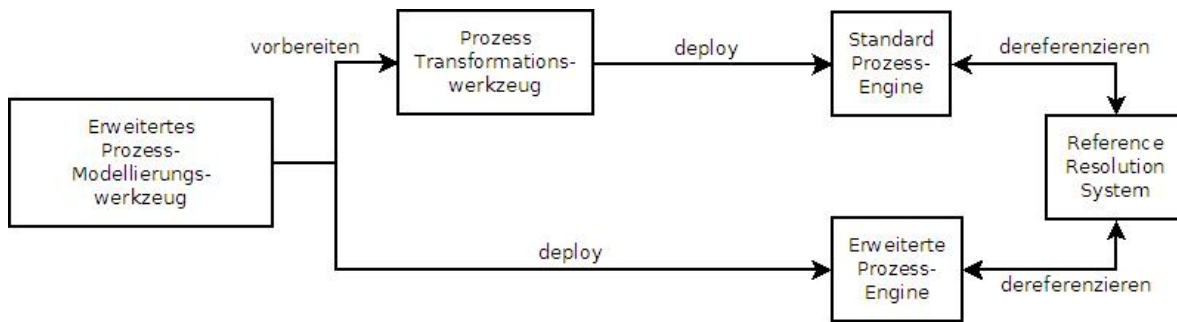


Abbildung 18: Alternativen zur Realisierung von Referenzen in BPEL

wie z.B. ein SQL-Query oder aber auch zentral gespeichert und anschließend über einen *Uniform Resource Identifier* (URI) referenziert werden. Beide Möglichkeiten haben Vorteile: Die direkte Angabe erlaubt es den Query in der EPR während der Laufzeit zu ändern. Mit der Verwendung von gespeicherten Queries wird dafür sichergestellt, dass alle den gleichen Query ausführen und niemand ungewollte Queries ausführen kann. **PortType** und **ServiceName** sind technische Parameter, die das dynamische Binden des RRS erlauben sollen. **Policy** ist optional und erlaubt das Hinzufügen von nichtfunktionalen Eigenschaften für die Ausführung der Queries. So kann z.B. *Quality of Context* (QoC) (Aktualität, Korrektheit, usw.), der für die zurückgelieferten Daten gelten soll, angegeben werden.

Weitere Details zu den Design-Entscheidungen und deren Nutzen finden sich in [3] unter „*Main Design Issues and Benefits of the Architecture*“.

7.1.3 Realisierung von Referenzen in BPEL

In [3] werden zwei verschiedene Ansätze (siehe Abbildung 18) zur Integration des Referenzen-Konzepts in BPEL beschrieben und deren Vor- und Nachteile erläutert.

Für beide Konzepte muss ein entsprechendes Modellierungswerkzeug erweitert werden, danach unterscheiden sich dann die Konzepte. Die erste Variante besteht darin den erweiterten BPEL-Quellcode unverändert auf einer erweiterten Workflow-Engine auszuführen, dass hat natürlich den entscheidenden Nachteil, dass nur noch eine entsprechend erweiterte Workflow-Engine diese Prozesse ausführen kann. Die zweite Variante umgeht dieses Problem, indem der veränderte BPEL-Quellcode in Standard-BPEL-Quellcode transformiert wird. Wir konzentrieren uns hier auf die Beschreibung der zweiten Variante, da diese einen universelleren Ansatz liefert, auf nahezu alle standard-konformen Workflow-Engines portiert werden kann und somit die Ausführungsumgebung des generierten BPEL-Quellcodes nicht eingeschränkt wird.

Erweiterung des Modellierungswerkzeugs

Wie schon gesagt, muss auf jeden Fall das Modellierungswerkzeug erweitert werden, um BPEL Prozesse mit Referenzen modellieren zu können. Zur Modellierung von solchen `<referenceVariable>` muss ein neuer Variablentyp eingeführt werden. Den Aufbau dieser Variablen zeigt das Schema in Listing 2.

```

<referenceVariable name="refName"
  valueType="xsd: schema"
  referenceType="onInstantiation | fresh |
periodic | external"
  period="duration"?
  external="partnerLink"? />*
  
```

Listing 2: Code eines `<referenceVariable>`-Schemas

Dieses Schema ist eine Erweiterung des Standard BPEL Variablen-Schemas. Das Attribut `valueType` spiegelt den Datentyp der referenzierten Variable wieder. Die Referenz-Variable selbst, ist implizit vom Type `xsd:EPR` und dient dazu die eigentliche Referenz zu speichern. Über das Attribut `referenceType` kann angegeben werden, wie aktuell referenzierte Werte sein sollen, also wann und wie oft die in den Referenzen hinterlegten Werte aktualisiert werden sollen. In [3] werden vier solcher Aktualitätskonstanten vorgestellt, wobei eine Vielzahl weiterer solcher Konstanten denkbar ist. Ihre Bedeutung und Verwendung wird nachfolgend bei der Beschreibung der Modelltransformation aufgezeigt.

Transformation des Modells

Die Integration eines Modelltransformation-Zwischenschritts vor dem Deployment des Prozesses erlaubt uns ein virtuelles Referenz-Handling für Prozesse zu realisieren ohne dabei die den Prozess ausführende Engine zu modifizieren. Die Grundidee ist dabei die Sprache BPEL nur im Modellierungswerkzeug, in dem wir zwischen normalen Variablen und Referenzen unterscheiden, zu erweitern. Der zusätzliche Transformationsschritt generiert dabei standard-konforme BPEL Konstrukte für die Handhabung von Referenzen und speist diese in das originale Prozess-Modell ein. Für jede Referenz, die im Prozessmodell deklariert ist, werden dafür zwei entsprechende Variablendeklarationen, wie in Listing 3 dargestellt, generiert.

```
<variable name="refName" type="xsd:schema"/>
<variable name="refNameEPR" type="xsd:EPR"/>
```

Listing 3: Code der generierten Variablen-Deklaration

Die erste Deklaration wird dabei für die Haltung des aktuellen Werts genutzt, die Zweite um die Referenz, die durch eine EPR repräsentiert wird, auf diesen Wert zu speichern. Weiterhin muss das RRS im Prozessmodell sichtbar sein, dies wird durch die Generierung eines entsprechenden `<partnerLink>` realisiert.

```
<invoke name="refNameRefresh_1" partnerLink="RRS"
  operation="GET" inputVariable="refNameEPR"
  outputVariable="refName"/>
```

Listing 4: Code einer Dereferenzierungs-Aktivität

Zu guter Letzt zeigt Listing 4 die tatsächliche Dereferenzierung einer Referenz über einen Web-Service Aufruf, der das RRS, das der Hauptbestandteil der Dereferenzierungsaktivität ist, aufruft. Der Modelltransformationsansatz speist Variablendeklarationen und Dereferenzierungsaktivitäten in den Workflow ein, um die Aktualisierung der Werte der Referenzen auszuführen. Wie bereits weiter oben erwähnt, ermöglicht es das Attribut `referenceType` dem Modellierer eine von verschiedenen Aktualisierungsoptionen für die Werte der Referenzen auszuwählen. Abhängig von der Auswahl des Modellierers müssen die Dereferenzierungsaktivitäten an der entsprechenden Position im Prozess folgendermaßen eingefügt werden:

- *onInstantiation (default)*: Bei der Instanzierung des Prozesses werden die Werte vom RRS abgefragt und die Variablen entsprechend gesetzt. Diese Einstellung ist für die Definition von Konstanten, die beim Prozessstart gesetzt werden und während der ganzen Laufzeit des Prozesses unverändert bleiben, sinnvoll. Für jede Referenz die auf *onInstantiation* gesetzt ist, wird die RRS Invoke-Aktivität aus Listing 4 zu einer Sequenz, die bei der Prozessinstanzierung ausgeführt wird, hinzugefügt.

- *fresh*: So frisch wie nur möglich - der Wert wird jedesmal abgefragt, wenn auf die Variable zugegriffen wird. Diese Einstellung ist nützlich, falls auf sich oft ändernde externe Werte, wie z.B. Sensordaten, zugegriffen werden soll. Hier muss die Dereferenzierungsaktivität direkt vor jeder Aktivität, die den Wert der Referenz liest, ausgeführt werden.
- *periodic*: Im Attribut **period** kann ein Zeitwert, wie z.B. 10 min, angegeben werden. Dieses Attribut beschreibt das maximale Alter (Zeitspanne seit letzter Aktualisierung), das ein lokal gespeicherter temporärer Wert einer Referenz haben darf. Nachdem diese Zeitspanne abgelaufen ist, wird der Wert aus dem RRS abgefragt und die temporäre Variable aktualisiert. Dafür wird ein `<onAlarm>` Element im globalen `<eventHandlers>` Element während der Transformation eingefügt. Diese Konstruktion liefert die periodische Aktualisierung von Werten über das wiederholte Abfragen dieser Werte aus dem RRS.
- *external*: Ein externer Event im Bereich der Web-Service Orchestrierung ist typischerweise eine Nachricht, die an die Prozessinstanz geschickt wird (oder ein Signal, das für alle Instanzen eines Prozessmodells gültig ist). Wird dieser Wert als **referenceType** gesetzt, können Aktualisierungen der Werte von Aussen, durch das Senden entsprechender Nachrichten an die Prozess-Engine, ausgelöst werden. Der Service von dem solch eine Nachricht erwartet wird, kann im Attribut **external** angegeben werden. Im Transformationsschritt wird dazu ein `<onEvent>` Konstrukt im globalen `<eventHandlers>` Element eingefügt.

7.2 Data-Management-Aktivitäten

In diesem Abschnitt werden die zu realisierenden Datenmanagement-Patterns zur Umsetzung einer generischen Unterstützung verschiedener Abfragesprachen für BPEL vorgestellt und im weiteren Verlauf deren Realisierung für die verschiedenen Datenquellen, d.h. für Dateisysteme, Datenbanken und Sensornetze, erläutert. Aus den in Abschnitt 7.2.2 identifizierten Funktionen, die für die Umsetzung der Datenmanagement-Patterns benötigt werden, werden dann in Abschnitt 7.2.3 neue benötigte BPEL-Aktivitäten abstrahiert, die dann später die entsprechenden Funktionen ausführen werden.

7.2.1 Datenmanagement-Patterns

In diesem Abschnitt werden alle Datenmanagement-Patterns, die zur Realisierung einer generischen Unterstützung von verschiedenen Abfragesprachen, aufgeführt und beschrieben.

Relationale Datenbanken

Dieser Abschnitt beschreibt alle Datenmanagement-Patterns, die zur Realisierung einer SQL-inline Unterstützung für BPEL benötigt werden (vgl. [2]).

Query Pattern Das Query Pattern beschreibt die Notwendigkeit, mithilfe von SQL-Befehlen externe Daten anfordern zu können. Die aus den Queries resultierenden Daten können dabei auf der Datenquelle zwischengespeichert oder direkt im BPEL-Prozess gehalten werden.

Set IUD Pattern Das Set IUD Pattern beschreibt die Möglichkeit auf mengenorientierten Daten außerhalb des Prozesses verschiedene Datenmanipulations-Operationen ausführen zu können. Zu diesen Operationen zählen das mengenorientierte Einfügen (insert), Aktualisieren (update) und Löschen (delete) von Daten.

Data Setup Pattern Das Data Setup Pattern liefert die Möglichkeit, benötigte Data Definition Language (DDL) Befehle auf einem relationalen Datenbanksystem auszuführen, um so während der Prozessausführung die Datenquelle zu konfigurieren oder neue Container (Tabellen, Schema, usw.) zu erstellen.

Stored Procedure Pattern Da die komplexe Verarbeitung von Daten meist durch Stored Procedures realisiert wird, ist es für die Verarbeitung von externen Daten unbedingt erforderlich, Stored Procedures auch aus einem Prozess heraus aufrufen zu können.

Set Retrieval Pattern Manchmal ist es nötig, Daten innerhalb des BPEL-Prozesses zu verarbeiten. Das Set Retrieval Pattern liefert dafür eine mengenorientierte Datenstruktur, in die man die angefragten externen Daten innerhalb des BPEL-Prozesses ablegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im BPEL-Prozess, der keine Verbindung zur originalen Datenquelle besitzt.

Set Access Pattern Das Set Access Pattern beschreibt die Notwendigkeit, auf den erzeugten Datencache sequentiell und direkt (random) zugreifen zu können.

Tuple IUD Pattern Das Tuple IUD Pattern beschreibt die Notwendigkeit, dass auch in einen Datencache Daten eingefügt, aktualisiert und wieder aus diesem gelöscht werden können.

Synchronization Pattern Das Synchronization Pattern realisiert die Synchronisation eines lokalen Datencaches mit der originalen Datenquelle.

XML-Datenbanken

Dieser Abschnitt beschreibt alle Datenmanagement-Patterns, die zur Realisierung einer XQuery-inline Unterstützung für BPEL benötigt werden.

Query Pattern Das Query Pattern beschreibt die Notwendigkeit, mithilfe von XQuery-Befehlen externe Daten anfordern zu können. Die aus den Queries resultierenden Daten können dabei auf der Datenquelle zwischengespeichert oder direkt im BPEL-Prozess gehalten werden.

Tree IUD Pattern Das Tree IUD Pattern beschreibt die Möglichkeit verschiedene Datenmanipulations-Operationen auf Knoten von baumartigen Daten außerhalb des Prozesses ausführen zu können. Zu diesen Operationen zählen das Einfügen, Ersetzen und Löschen von Knoten.

Node IUD Pattern Das Node IUD Pattern beschreibt die Möglichkeit verschiedene Datenmanipulations-Operationen in Knoten von baumartigen Daten außerhalb des Prozesses ausführen zu können. Zu diesen Operationen zählen das Einfügen, Ersetzen und Löschen von Werten innerhalb von Knoten.

Data Setup Pattern Das Data Setup Pattern liefert die Möglichkeit, benötigte Data Definition Language (DDL) Befehle auf einem XML-Datenbanksystem auszuführen, um so während der Prozessausführung die Datenquelle zu konfigurieren oder neue Container (Tabellen, Schema, usw.) zu erstellen.

Stored Procedure Pattern Da die komplexe Verarbeitung von Daten meist durch Stored Procedures realisiert wird, ist es für die Verarbeitung von externen Daten unbedingt erforderlich, Stored Procedures auch aus einem Prozess heraus aufrufen zu können.

Tree Retrieval Pattern Manchmal ist es nötig, Daten innerhalb des BPEL-Prozesses zu verarbeiten. Das Tree Retrieval Pattern liefert dafür eine baumartige Datenstruktur, in die man die angefragten externen Daten innerhalb des BPEL-Prozesses ablegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im BPEL-Prozess, der keine Verbindung zur originalen Datenquelle besitzt.

Tree Access Pattern Das Tree Access Pattern beschreibt die Notwendigkeit, auf den erzeugten Datencache sequentiell und direkt (random) zugreifen zu können.

Node IUD Pattern Das Node IUD Pattern beschreibt die Notwendigkeit, dass auch in einen Datencache baumartige Daten eingefügt, aktualisiert und wieder aus diesem gelöscht werden können.

Synchronization Pattern Das Synchronization Pattern realisiert die Synchronisation eines lokalen Datencaches mit der originalen Datenquelle.

Dateisysteme

Dieser Abschnitt beschreibt alle Datenmanagement-Patterns, die für die Unterstützung von Dateisystemen in BPEL benötigt werden.

Query Pattern Das Query Pattern beschreibt die Notwendigkeit externe Daten anfordern zu können. Die aus den Abfragen resultierenden Daten können dabei auf der Datenquelle zwischengespeichert oder direkt im BPEL-Prozess gehalten werden.

File IUD Pattern Das File IUD Pattern beschreibt die Möglichkeit verschiedene Datenmanipulations-Operationen auf Dateien eines Dateisystems ausführen zu können. Zu diesen Operationen zählen das Einfügen , Ersetzen und Löschen von Dateien.

Data IUD Pattern Das Data IUD Pattern beschreibt die Möglichkeit verschiedene Datenmanipulations-Operationen innerhalb von Dateien eines Dateisystems ausführen zu können. Zu diesen Operationen zählen das Einfügen , Ersetzen und Löschen von Dateiinhalten.

Data Setup Pattern Das Data Setup Pattern liefert die Möglichkeit, benötigte Data Definition Language (DDL) Befehle auf einem Dateisystem auszuführen, um so während der Prozessausführung die Datenquelle zu konfigurieren oder neue Container (Ordner, Dateien, usw.) zu erstellen.

Stored Procedure Pattern Da die komplexe Verarbeitung von Daten meist durch Stored Procedures realisiert wird, ist es für die Verarbeitung von externen Daten unbedingt erforderlich, Stored Procedures auch aus einem Prozess heraus aufrufen zu können.

File Retrieval Pattern Das File Retrieval Pattern liefert eine Datenstruktur, in die man, aus einem Dateisystem, abgefragte Dateien innerhalb des BPEL-Prozesses ablegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im BPEL-Prozess, der keine Verbindung zur originalen Datenquelle besitzt.

File Access Pattern Das File Access Pattern beschreibt die Notwendigkeit, auf den erzeugten Datencache sequentiell zugreifen zu können.

Data IUD Pattern Das Data IUD Pattern beschreibt die Notwendigkeit, dass auch in einen Datencache Daten eingefügt, aktualisiert und wieder aus diesem gelöscht werden können.

Synchronization Pattern Das Synchronization Pattern realisiert die Synchronisation eines lokalen Datencaches mit der originalen Datenquelle.

Sensornetze

Dieser Abschnitt beschreibt alle Datenmanagement-Patterns, die für die Unterstützung von Sensornetzen in BPEL benötigt werden.

Query Pattern Das Query Pattern beschreibt die Notwendigkeit, mithilfe von SQL-Befehlen externe Daten anfordern zu können. Die aus den Queries resultierenden Daten können dabei auf der Datenquelle in Buffer-Tabellen zwischengespeichert oder direkt im BPEL-Prozess gehalten werden.

Data Setup Pattern Das Data Setup Pattern liefert die Möglichkeit, benötigte Data Definition Language (DDL) Befehle auf dem Sensornetz-Datenbanksystem auszuführen, um so während der Prozessausführung die Datenquelle zu konfigurieren oder neue Container (Buffer-Tabellen) zu erstellen.

Stored Procedure Pattern Da die komplexe Verarbeitung von Daten meist durch Stored Procedures realisiert wird, ist es für die Verarbeitung von externen Daten unbedingt erforderlich, Stored Procedures auch aus einem Prozess heraus aufrufen zu können.

Set Retrieval Pattern Manchmal ist es nötig, Daten innerhalb des BPEL-Prozesses zu verarbeiten. Das Set Retrieval Pattern liefert dafür eine mengenorientierte Datenstruktur, in die man die angefragten externen Daten innerhalb des BPEL-Prozesses ablegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im BPEL-Prozesses, der keine Verbindung zur originalen Datenquelle besitzt.

Set Access Pattern Das Set Access Pattern beschreibt die Notwendigkeit, auf den erzeugten Datencache sequentiell und direkt (random) zugreifen zu können.

Tuple IUD Pattern Das Tuple IUD Pattern beschreibt die Notwendigkeit, dass auch in einen Datencache Daten eingefügt, aktualisiert und wieder aus diesem gelöscht werden können.

7.2.2 Umsetzung der Datenmanagement-Patterns

In diesem Abschnitt wird die Realisierung der verschiedenen Datenmanagement-Patterns aus Abschnitt 7.2.1 im Zusammenhang mit den jeweiligen Datenquellentypen beschrieben.

Relationale Datenbanken

Hier wird die Umsetzung der in Abschnitt 7.2.1 beschriebenen Patterns im Bereich der SQL-Datenbanken durch bestehende SQL-Befehle oder, falls erforderlich, durch die Definition neuer zu implementierender Funktionen beschrieben.

- Query Pattern: Realisierung durch SQL-SELECT.
 - Beispiel:
 - * SELECT info FROM customer
- Set IUD Pattern: Realisierung durch SQL-INSERT, SQL-UPDATE, SQL-DELETE.
 - Beispiele:
 - * INSERT INTO department VALUES ('E31', 'architecture', '00390', 'E01')
 - * INSERT INTO department SELECT * FROM old_department
 - * UPDATE employee SET job = 'laborer' WHERE empno = '000290'
 - * DELETE FROM department WHERE deptno = 'D11'
- Data Setup Pattern: Realisierung durch SQL-CREATE SCHEMA, SQL-CREATE TABLE, SQL-CREATE VIEW und weitere SQL-CREATE Befehle. Weiterhin sollte auch das Löschen dieser Objekte möglich sein, dies wird durch SQL-DROP realisiert.
 - Beispiele:

- * CREATE SCHEMA internal AUTHORIZATION admin
 - * CREATE TABLE tdept (deptno CHAR(3) NOT NULL, deptname VARCHAR(36) NOT NULL, mgrno CHAR(6), admrdept CHAR(3) NOT NULL, PRIMARY KEY(deptno)) IN internal
 - * CREATE VIEW internal.departmentView AS
SELECT deptno, deptname FROM internal.tdept
 - * DROP SCHEMA internal
 - * DROP TABLE tdept
 - * DROP VIEW internal.departmentView
- Stored Procedure Pattern: Realisierung durch SQL-CALL.
 - Beispiele:
 - * CALL parts_on_hand (?, ?, ?) (? = Parameter der Prozedur)
 - Set Retrieval Pattern: Realisierung durch die Definition und Bereitstellung einer entsprechenden BPEL-Aktivität. Diese Aktivität liefert dann die Möglichkeit, dass durch einen SQL-SELECT Befehl abgefragte Daten (siehe Query Pattern) in den BPEL-Prozess geladen und in einer entsprechenden BPEL-Variable abgelegt werden können. Dazu müssen neue Variablentypen bereitgestellt werden, die den Strukturen der zu haltenden Daten entsprechen, also mengenartige Daten halten können. Die Daten werden dazu über die *Service Data Objects API* (SDO API) abstrahiert. Wie in [2] beschrieben, wurde dieses Konzept bereits durch die *Business Integration Suite* und dem darin enthaltenen *WebSphere Integration Developer* mit der Bezeichnung “Retrieve Set Activity” realisiert. Dabei werden in einer Retrieve Set Activity externe Daten in eine XML-Struktur innerhalb des BPEL-Prozesses geladen, eine etwas ausführlichere Beschreibung liefert [2].
 - Set Access Pattern: Um das Set Access Pattern zu realisieren, müssen im Projektverlauf Methoden, die eine sequentielle und direkte Bearbeitung des durch eine RetrieveData Aktivität erzeugten Datencaches, bereitgestellt werden. D.h. konkret, dass es möglich sein muss, mit den definierten Variablen und deren Typen innerhalb von BPEL zu arbeiten. Die SDO API liefert dafür bereits einige Methoden, die verwendet werden können. Es muss weiterhin möglich sein, dass entsprechende Variablen auch in BPEL-Aktivitäten, wie z.B. einer ForEach Activity, verwendet werden können.
 - Tuple IUD Pattern: Erweitert die Set Access Pattern Methoden um die Möglichkeiten, Tupel innerhalb der mengenorientierten Datenstruktur im BPEL-Prozess zu aktualisieren, einzufügen und zu löschen. Die SDO API liefert dafür bereits einige Methoden, die verwendet werden können.
 - Synchronization Pattern: Um die Daten aus dem Prozesscache zurück auf die originale Datenbank zu übertragen, muss ebenfalls eine neue BPEL-Aktivität erstellt werden, durch die der Benutzer angibt, dass die Daten zurückgeschrieben werden sollen. Diese Aktivität nutzt dann intern die im Set IUD Pattern und Set Access Pattern beschriebenen Methoden und ebenso SQL-Befehle um die Daten aus dem Prozesscache auf die Datenbank zu übertragen. Die SDO API und die DAS API liefern dafür bereits einige Methoden, die verwendet werden können.

XML-Datenbanken

Hier wird die Umsetzung der in Abschnitt 7.2.1 beschriebenen Patterns im Bereich der XQuery-Datenbanken durch bestehende XQuery-Befehle oder, falls erforderlich, durch die Definition neuer zu implementierender Funktionen beschrieben.

- Query Pattern: Realisierung durch entsprechende FLWOR-XQuery-Befehle.
- Beispiel:

- FOR \$d IN \$doc/books/book RETURN {\$d/title/text}
- Tree IUD Pattern: Realisierung durch entsprechende XQuery-Befehle.
 - Beispiele:
 - * INSERT NODE <year>2005</year> AFTER fn:doc("bib.xml")/books/book[1]/publisher
 - * REPLACE NODE fn:doc("bib.xml")/books/book[1]/publisher WITH
fn:doc("bib.xml")/books/book[2]/publisher
 - * RENAME NODE fn:doc("bib.xml")/books/book[1]/author[1] AS "principal-author"
 - * DELETE NODE fn:doc("bib.xml")/books/book[1]/author[last()]
- Node IUD Pattern: Realisierung durch entsprechende XQuery-Befehle.
 - Beispiel:
 - * REPLACE VALUE OF NODE fn:doc("bib.xml")/books/book[1]/price WITH
fn:doc("bib.xml")/books/book[1]/price * 1.1
- Data Setup Pattern: Wird erst in einer späteren Iteration realisiert.
- Stored Procedure Pattern: Wird erst in einer späteren Iteration realisiert.
- Tree Retrieval Pattern: Wird erst in einer späteren Iteration realisiert.
- Tree Access Pattern: Wird erst in einer späteren Iteration realisiert.
- Node IUD Pattern: Wird erst in einer späteren Iteration realisiert.
- Synchronization Pattern: Wird erst in einer späteren Iteration realisiert.

Dateisysteme

Hier wird die Umsetzung der in Abschnitt 7.2.1 beschriebenen Patterns im Bereich der Dateisysteme durch entsprechende Systemaufrufe des dem Dateisystem zugrundeliegenden Betriebssystems beschrieben.

- Query Pattern: Realisierung durch einen GET-Befehl, der intern java.io Methoden verwendet, um Inhalte aus Dateien oder komplette Dateien aus einem Dateisystem zu lesen. Für das Abfragen von Dateiinhalten muss noch ein Konzept erarbeitet werden, wie entsprechende Daten gezielt über einen Befehl ausgewählt und abgefragt werden können. Eine Möglichkeit wäre z.B. einen Filter-Dialog zur Angabe von Suchparametern über die GUI bereitzustellen, der es ermöglicht Dateiinhalte zu suchen und zurückzugeben.
- File IUD Pattern: Realisierung durch einen entsprechenden PUT- und REMOVE-Befehl (RM), die intern java.io Methoden verwenden, um Inhalt in Dateien zu schreiben und aus ihnen zu löschen. Ein Update wird bei Dateien intern durch das Löschen des alten Wertes und anschließendem Einfügen des neuen Wertes ausgeführt. Dafür muss noch ein Konzept erarbeitet werden, wie entsprechende Informationen gezielt über einen Befehl eingefügt und gelöscht werden können.
- Data IUD Pattern: Wird erst in einer späteren Iteration realisiert.
- Data Setup Pattern: Realisierung durch die Verwendung entsprechender Erzeugungsbefehle, wie MKDIR und MKFILE zur Erzeugung von Ordnern und Dateien. Diese werden intern wieder durch java.io-Methoden realisiert. Hier sollte es auf jeden Fall möglich sein, Dateien und Ordner in einem Dateisystem zu erzeugen. Ebenso sollte das Löschen auf der gleichen Ebene, d.h. von ganzen Dateien und Ordnern, möglich sein.

- **Stored Procedure Pattern:** Wird nur eventuell in einer späteren Iteration umgesetzt. Eine Realisierung wäre z.B. über den Aufruf von auf dem Dateisystem hinterlegten Batch-Dateien denkbar.
- **File Retrieval Pattern:** Realisierung durch die Definition und Bereitstellung einer entsprechenden BPEL-Aktivität. Diese Aktivität liefert dann die Möglichkeit, dass durch einen GET-Befehl abgefragte Daten (siehe Query Pattern) in den BPEL-Prozess geladen und in einer entsprechenden BPEL-Variable abgelegt werden können. Dazu müssen neue Variablentypen bereitgestellt werden, die den Strukturen der zu haltenden Daten entsprechen. Die Daten werden dazu über die *Service Data Objects API* (SDO API) abstrahiert. Wie in [2] beschrieben, wurde dieses Konzept bereits durch die *Business Integration Suite* und dem darin enthaltenen *WebSphere Integration Developer* mit der Bezeichnung *“Retrieve Set Activity”* für mengenorientierte Daten realisiert. Dabei werden in einer Retrieve Set Activity externe Daten in eine XML-Struktur innerhalb des BPEL-Prozesses geladen, eine etwas ausführlichere Beschreibung liefert [2].
- **File Access Pattern:** Um das File Access Pattern zu realisieren, müssen im Projektverlauf Methoden, die eine sequentielle Bearbeitung des durch eine RetrieveData Aktivität erzeugten Datencaches, bereitgestellt werden. D.h. konkret, dass es möglich sein muss, mit den definierten Variablen und deren Typen innerhalb von BPEL zu arbeiten. Die SDO API liefert dafür bereits einige Methoden, die verwendet werden können. Es muss weiterhin möglich sein, dass entsprechende Variablen auch in BPEL-Aktivitäten, wie z.B. einer ForEach Activity, verwendet werden können.
- **Data IUD Pattern:** Erweitert die File Access Pattern Methoden um die Möglichkeiten Werte innerhalb des Datencaches des BPEL-Prozesses zu aktualisieren, einzufügen und zu löschen. Die SDO API liefert dafür bereits einige Methoden, die verwendet werden können.
- **Synchronization Pattern:** Um die Daten aus dem Prozesscache zurück auf die originale Datenquelle zu übertragen, muss ebenfalls eine neue BPEL-Aktivität erstellt werden, durch die der Benutzer angibt, dass die Daten zurückgeschrieben werden sollen. Diese Aktivität nutzt dann intern die im Set IUD Pattern und Set Access Pattern beschriebenen Methoden um die Daten aus dem Prozesscache auf die Datenquelle zu übertragen. Die SDO API und die *Data Access Services API* (DAS API) liefern dafür bereits einige Methoden die verwendet werden können. Die Umsetzung dieses Patterns wird erst in einer späteren Iteration realisiert, da zuerst Erkenntnisse im Bereich der Datenbanken gesammelt werden und mit diesen dann ein Ansatz für Dateisysteme erarbeitet wird.

Sensornetze

Hier wird die Umsetzung der in Abschnitt 7.2.1 beschriebenen Patterns im Bereich der Sensornetze und deren Datenbanken durch bestehende sensornetzspezifische SQL-Befehle oder, falls erforderlich, durch die Definition neuer zu implementierender Funktionen beschrieben. Alle SQL-Befehlsbeispiele beziehen sich hier auf den SQL-Dialekt der Sensornetz-Datenbank TinyDB (siehe [15]).

- **Query Pattern:** Realisierung durch ein entsprechendes SQL-SELECT der Sensornetz-Datenbank. Weiterhin besteht die Möglichkeit, sensornetzintern in einem Buffer Werte zwischenspeichern. Realisiert wird dies durch im Arbeitsspeicher von Sensoren (siehe Data Setup Pattern) erstellte Tabellen, die mit momentanen Sensorwerten gefüllt werden können, um später die Daten zeitversetzt abrufen zu können. Die gewünschten Werte werden dabei mit einem entsprechenden SQL-SELECT Befehl in den Buffer geschrieben.
 - Befehlsstruktur:


```
* SELECT select-list [FROM sensors] WHERE where-clause [GROUP BY gb-list [HAVING having-list]] [TRIGGER ACTION command-name[(param)]] [EPOCH DURATION integer]
```

- * Werte einfügen in Buffer-Tabelle: `SELECT field1, field2, ... FROM sensors SAMPLE PERIOD x INTO name`
- Beispiele:
 - * `SELECT temp FROM sensors WHERE temp > thresh TRIGGER ACTION SetSnd(512) EPOCH DURATION 512`
 - * `SELECT field1, field2 SAMPLE PERIOD 100 FROM name (SELECT auf Buffer-Tabelle name)`
- Data Setup Pattern: Es können Tabellen im Arbeitsspeicher der Sensoren erstellt werden, die dann Werte von Sensoren aufnehmen und halten können. Die Erstellung solcher Tabellen wird durch `SQL-CREATE BUFFER` und das Löschen des gesamten Buffers durch `SQL-DROP` realisiert.
 - Eine Buffer-Tabelle erstellen: `CREATE BUFFER name SIZE x (field1 type, field2 type, ...)`
 - * *Bemerkungen: **x** ist die Anzahl der Zeilen, **type** ist ein Datentyp aus der Menge {uint8, uint16, int8, int16, int32} und **field1**, **field2**, usw. sind Spaltennamen, die wie der Tabellename jeweils 8 Zeichen lang sein dürfen.*
 - Alle Buffer-Tabellen löschen: `DROP ALL`
- Stored Procedure Pattern: Es gibt für die TinyDB einen Stored Procedure-ähnlichen Ansatz. Dabei können, falls eine bestimmte Bedingung gilt (z.B. Temperatur > 20°C), sogenannte *commands* aufgerufen werden. Der Code für diese Methoden wird auf die entsprechenden Sensoren übertragen und dort dann bei Bedarf ausgeführt. Ein Zugriff von außerhalb wie bei Stored Procedures ist hier allerdings nicht möglich.
 - Beispiele:
 - * `SELECT temp FROM sensors WHERE temp > thresh TRIGGER ACTION SetSnd(512) EPOCH DURATION 512`
 - *Bemerkungen: Hier wird alle 512ms die Temperatur an den Sensoren abgefragt und falls diese einen gewissen Wert übersteigt, wird über den **command** SetSnd(512) für 512ms ein Signalton ausgegeben.*
- Set Retrieval Pattern: Realisierung ebenso wie bei SQL-Datenbanken beschrieben.
- Set Access Pattern: Realisierung ebenso wie bei SQL-Datenbanken beschrieben.
- Tuple IUD Pattern: Realisierung ebenso wie bei SQL-Datenbanken beschrieben.

7.2.3 Resultierende BPEL-Aktivitäten

In diesem Abschnitt werden die durch Abschnitt 7.2.2 identifizierten BPEL-Aktivitäten aufgezählt und ihre Funktion noch einmal kurz beschrieben. Dazu gehört z.B. auch, welche Attribute welchen Typs für die einzelnen Aktivitäten benötigt werden. Generell gilt, dass alle Aktivitäten mindestens die vier Variablen *type*, *kind*, *dsAddress* und *statement* vom Typ String besitzen. Die Variable *type* dient zur Angabe des Datenquellentyps, für den die Aktivität ausgeführt wird, also ob es sich um ein Dateisystem, eine Datenbank oder ein Sensornetz handelt. Die Variable *kind* dient zur Angabe der Datenquellenart, d.h. um was für ein Dateisystem, was für eine Datenbank oder welche Art von Sensornetz es sich handelt. Diese Informationen sind wichtig, um intern den richtigen SQL-Dialekt bzw. Befehlssatz für die entsprechende Datenquelle auszuwählen. Die Variable *dsAddress* dient zur Angabe der Datenquellenadresse und die Variable *statement* zur Haltung des entsprechenden Systemaufrufs bzw. SQL- oder XQuery-Befehls, der ausgeführt werden soll. Da jede der definierten Aktivitäten diese

vier Variablen besitzt, werden diese nachfolgend nicht mehr angegeben und nur falls benötigt weitere aktivitätsspezifische Variablen beschrieben. Weiterhin werden die verschiedenen Ausprägungen der einzelnen Aktivitäten im Hinblick auf die zugrundeliegende Datenquelle aufgezeigt, so dass am Ende ein vollständiger Überblick aller definierten Aktivitäten und ihrer Ausprägungen vorliegt. Generell gibt es durch die verschiedenen Datenquellen nur wenige strukturelle Unterschiede in den Aktivitäten. Das liegt vor allem daran, dass auf datenquellenspezifische Eigenschaften bereits in der graphischen Oberfläche, also dem Eclipse BPEL Designer, eingegangen werden soll und diese so durch entsprechende Dialoge intern immer auf dieselbe Struktur abgebildet werden können. Eben dazu sollen alle Befehle auch über entsprechende graphische Elemente angegeben werden können, indem sie einfach “zusammengeklickt” werden können (siehe Abbildung 10). Dadurch soll eine einfachere Handhabung realisiert werden, damit der Benutzer schnellstmöglich ohne detaillierte Kenntnisse der Abfragesprache alle zur Verfügung gestellten Daten- und Datenquellenbefehle für alle unterstützten Datenquellen ausführen kann. Trotz der Einschränkungen, die ohne Zweifel durch die grafische Erstellung der Befehle bestehen, steht dem Benutzer der volle Sprachumfang der jeweiligen Datenquellsprache bzw. der vollständige Befehlssatz der Datenquelle zur Verfügung. Der volle Sprachumfang kann dabei mindestens, wie in Abbildung 10 dargestellt, über die Angabe von Befehlen im “Resulting statement”-Textfeld genutzt werden.

Nachfolgend werden alle, durch die Abschnitte 7.2.1 und 7.2.2 identifizierten, Aktivitäten aufgezeigt und beschrieben. Dabei werden in den Listings der DM-Aktivitäten die in [5] beschriebenen Notationen und Definitionen verwendet. Die Schemata der DM-Aktivitäten werden im Verlauf der 1. Iteration vervollständigt.

Query Aktivität

Diese Aktivität ermöglicht es, aus jeder beliebigen Datenquelle Daten zu lesen. Weitere spezifische Attribute werden dafür nicht benötigt. Die Query Aktivität ist für alle Datenquellen gleich strukturiert, und nur die entsprechenden Query-Befehle unterscheiden sich je nach Datenquelle. Diese Aktivität wird auch für das sensornetzinterne Einfügen von Sensordaten in Buffer-Tabellen genutzt (wie auch in Abschnitt 7.2.2 beschrieben). Listing 5 zeigt das BPEL-Schema einer Query Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <queryActivity standard-attributes>

    standard-elements

  </queryActivity>
</extensionActivity>
```

Listing 5: Schema einer Query Aktivität

Insert Aktivität

Diese Aktivität ermöglicht es, Daten in eine Datenquelle einzufügen. Listing 6 zeigt das BPEL-Schema einer Insert Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <insertActivity standard-attributes>

    standard-elements

  </insertActivity>
</extensionActivity>
```

Listing 6: Schema einer Insert Aktivität

Update Aktivität

Diese Aktivität ermöglicht es, auf einer Datenquelle hinterlegte Daten durch aktuellere externe Daten zu ersetzen. Diese Aktivität kann nicht auf Sensornetze angewendet werden. Listing 6 zeigt das BPEL-Schema einer Update Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <updateActivity standard-attributes>

    standard-elements

  </updateActivity>
</extensionActivity>
```

Listing 7: Schema einer Update Aktivität

Delete Aktivität

Diese Aktivität ermöglicht es, Daten in Datenquellen zu löschen. Diese Aktivität kann nicht auf Sensornetze angewendet werden. Listing 8 zeigt das BPEL-Schema einer Delete Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <deleteActivity standard-attributes>

    standard-elements

  </deleteActivity>
</extensionActivity>
```

Listing 8: Schema einer Delete Aktivität

Create Aktivität

Diese Aktivität ermöglicht es, Zuordnungseinheiten (Dateien, Ordner, Tabellen, Schemata, usw.) auf beliebigen Datenquellen zu erstellen. Dabei werden die folgenden Befehle zur Erstellung von Zuordnungseinheiten auf den entsprechenden Datenquellen unterstützt:

- Dateisysteme
 - MKDIR FOLDER
 - MKFILE FILE
- Datenbanken
 - CREATE TABLE
 - CREATE SCHEMA
 - CREATE VIEW
 - optional:

- * CREATE DOMAIN
- * CREATE INDEX
- * CREATE TRIGGER
- Sensornetze
 - CREATE BUFFER

Listing 9 zeigt das BPEL-Schema einer Create Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <createActivity standard-attributes>

    standard-elements

  </createActivity>
</extensionActivity>
```

Listing 9: Schema einer Create Aktivität

Call Aktivität

Diese Aktivität ermöglicht es auf der Datenquelle hinterlegte Prozeduren auszuführen. Diese Aktivität kann vorerst nur für Datenbanken verwendet werden. Listing 10 zeigt das BPEL-Schema einer Call Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <callActivity standard-attributes>

    standard-elements

  </callActivity>
</extensionActivity>
```

Listing 10: Schema einer Call Aktivität

RetrieveData Aktivität

Diese Aktivität ermöglicht es, Daten von Datenquellen in den BPEL-Prozess zu laden. Listing 11 zeigt das BPEL-Schema einer RetrieveData Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <retrieveDataActivity standard-attributes>

    standard-elements

  </retrieveDataActivity>
</extensionActivity>
```

Listing 11: Schema einer RetrieveData Aktivität

Drop Aktivität

Diese Aktivität ermöglicht es Zuordnungseinheiten auf beliebigen Datenquellen zu löschen. Dazu werden die folgenden Befehle unterstützt:

- Dateisysteme
 - RMDIR FOLDER
 - RM FILE
- Datenbanken
 - DROP TABLE
 - DROP SCHEMA
 - DROP VIEW
 - optional:
 - * DROP DOMAIN
 - * DROP INDEX
 - * DROP TRIGGER
- Sensornetze
 - DROP ALL

Listing 12 zeigt das BPEL-Schema einer Drop Aktivität mit der in [5] definierten Notation.

```
<extensionActivity>
  <dropActivity standard-attributes>

      standard-elements

  </dropActivity>
</extensionActivity>
```

Listing 12: Schema einer Drop Aktivität

Optionale Aktivitäten:

- **Import Aktivität:** Diese Aktivität ermöglicht es, lokale Daten (z.B. Simulationsparameter) des Benutzers in einen Prozess einzubinden und in diesem zu verwenden.
- **Export Aktivität:** Diese Aktivität ermöglicht es, Daten eines Prozesses (z.B. Simulationsergebnisse) lokal auf den Benutzer-Rechner zu exportieren.
- **Move Aktivität:** Diese Aktivität ermöglicht es, Daten zwischen beliebigen Datenquellen zu kopieren/verschieben, d.h. es können beispielsweise Daten aus einer Datei in eine Datenbank-Tabelle verschoben/kopiert werden.
- **WriteBack Aktivität:** Diese Aktivität ermöglicht es, Daten aus dem BPEL-Prozess auf die originale Datenquelle zurückzuschreiben. Diese Aktivität existiert nicht für Sensornetze.
- **Alter Aktivität:** Diese Aktivität ermöglicht es, bereits bestehende Zuordnungseinheiten (Tabellen, Schemata, usw.) nachträglich gezielt zu verändern und an veränderte Anforderungen anzupassen. Es ist z.B. möglich in eine Tabelle neue Spalten einzufügen, bestehende Spalten neu zu benennen oder zu löschen.

In Tabelle 1 werden nocheinmal alle resultierenden DM-Aktivitäten und ihre Verwendbarkeit im Bezug auf die verschiedenen Datenquellen aufgezeigt. Dabei steht ein “x” für die vollständige Anwendbarkeit einer Aktivität auf eine Datenquelle, “(x)” steht für die teilweise Verwendbarkeit (z.B. in einer Move Aktivität können Sensornetze nur als Quelle nicht als Ziel angegeben werden) bzw. dass eventuell erst in einer späteren Iterationen diese Aktivität auf eine Datenquelle angewendet werden kann. Ein “-” steht dafür, dass eine Aktivität nicht mit einer entsprechenden Datenquelle verwendet werden kann.

Tabelle 1: Übersicht der resultierenden Aktivitäten und ihrer Anwendbarkeit auf verschiedene Datenquellen

Aktivität	RDB	XML-DB	Dateisysteme	Sensornetze (TinyDB)
Query Aktivität	x	x	x	x
Insert Aktivität	x	x	x	-
Update Aktivität	x	x	x	-
Delete Aktivität	x	x	x	-
Create Aktivität	x	x	x	x
Call Aktivität	x	x	x	x
RetrieveData Aktivität	x	(x)	(x)	x
Drop Aktivität	x	x	x	x
Import Aktivität	x	x	x	-
Export Aktivität	x	x	x	-
Move Aktivität	x	x	x	(x)
WriteBack Aktivität	(x)	(x)	(x)	(x)
Alter Aktivität	x	x	(x)	-

7.3 Authentifizierung und Autorisierung

Dieser Abschnitt behandelt die Authentifizierung und Autorisierung bei dem Zugriff auf Datenquellen und die dabei eingesetzten Technologien und Konzepte. Bei beiden Verfahren handelt es sich um Verfahren auf Seite der Datenquellen, d.h. es findet keine Autorisierung und Authentifizierung der Benutzer innerhalb eines Prozesses statt. Das Rahmenwerk bietet dem Benutzer lediglich die Möglichkeit benötigte Authentifizierungs- und Autorisierungsinformationen bereitzustellen und ggf. Zwischenzuspeichern. Ziel des Rahmenwerks ist es, eine Vielzahl an Authentifizierungs- und Autorisierungs-Verfahren zu unterstützen und für weitere Verfahren erweiterbar zu sein.

7.3.1 Authentifizierung

Bei der Authentifizierung wird vor dem eigentlichen Zugriff zunächst ein Vertrauensverhältnis zwischen Benutzer und Datenquelle hergestellt. Je nach Verfahren wird dabei das Vertrauen einseitig oder auch beidseitig hergestellt. Für das Rahmenwerk soll dabei das Konzept des Single Sign On (SSO) angewendet werden, das nach einer erfolgreich durchgeführten Authentifizierung verhindert, dass bei weiteren Zugriffen eine erneute Authentifizierung durchgeführt werden muss. Unabhängig von der Realisierung, soll das für den Benutzer auch bedeuten, dass die Authentifizierungsinformationen nicht in jeder DM-Aktivität zu einer Datenquelle redundant angegeben werden müssen.

Benutzername und Passwort

Benutzername und Passwort können in der ersten Iteration zunächst nur in den Globalen Einstellungen (siehe 4.1.1) für alle Datenquellen festgelegt werden.

7.3.2 Autorisierung

Eine Autorisierung findet meist nach einer erfolgreichen Authentifizierung statt und überprüft, ob die Operation eines Zugriffs berechtigt ist. Dazu werden Zugriffsrechte überprüft, die als Regeln formuliert auf Seite der Datenquellen verwaltet werden. In der ersten Iteration wird lediglich die Autorisierung über den Benutzernamen unterstützt.

Benutzername

Wird durch die Authentifizierung durch Benutzername und Passwort unterstützt.

7.4 Transaktionen

7.5 Auditing

In diesem Abschnitt wird eine Beschreibung des geplanten Auditing von SIMPL gegeben. Dazu wird zunächst auf das bestehende Monitoring von ODE eingegangen.

7.5.1 Momentane Situation - Monitoring von ODE

Das Monitoring von ODE wird durch sogenannte Events und Event Listener sowie die Management API realisiert.

Management API

Die Management API der ODE Engine macht es möglich zahlreiche Informationen abzurufen. So ist es beispielsweise möglich, zu überprüfen, welche Prozesse gerade eingesetzt werden und welche Prozessinstanzen ausgeführt werden oder beendet sind. Dies ist besonders für das Monitoring wichtig, da es hierdurch möglich ist spezifische Informationen zu einem bestimmten Prozess oder einer bestimmten

Instanz abzurufen. Dies kann zum Beispiel das Erstellungsdatum sein, eine Liste der Events, die für diesen Prozess oder diese Instanz generiert wurden und vieles mehr.

Events

Events sind Ereignisse, die auftreten, wenn bestimmte Aktionen innerhalb der ODE Engine durchgeführt werden. Zum Beispiel das Aktivieren eines Prozesses oder das Erzeugen einer neuen Prozessinstanz. Diese Events machen es möglich, zu verfolgen, was innerhalb der ODE Engine passiert und produzieren detaillierte Informationen über die Prozessausführung. Sie können mit Hilfe der Management API abgefragt werden oder man nutzt Event Listener.

Event Listener

Event Listener sind bestimmte Konstrukte, die es ermöglichen bei Auftreten bestimmter Events eine direkte Rückmeldung zu geben oder auf verschiedene Events zu reagieren und event-spezifische Aktionen durchzuführen.

7.5.2 Auditing von SIMPL

Im BPEL Designer ist der Zugriff auf das Auditing und die Auditingeinstellungen unter dem entsprechenden Menüpunkt möglich. Hier kommt man nun zum entsprechenden Interface, in dem sich verschiedene Einstellungen für das Auditing tätigen lassen (siehe Abbildung 5). Es ist möglich das Auditing zu aktivieren oder zu deaktivieren. Weiterhin ist möglich die Datenbank für das Speichern der Auditing Daten festzulegen. Das Auditing ist standardmäßig aktiviert, kann allerdings über die Adminkonsole deaktiviert und auch erneut aktiviert werden.

Für das Auditing von SIMPL werden die von ODE erzeugten Daten direkt an den SIMPL Core übergeben und weiterverarbeitet, anstatt diese wie bisher in der virtuellen Datenbank von ODE abzuliegen. Die so erzeugten Daten über die Prozesse, Instanzen und Events werden anschließend in einer zuvor in der Adminkonsole festgelegten Auditing-Datenbank abgelegt.

7.5.3 Event Modell

In diesem Unterabschnitt wird das Event Modell für das Auditing von SIMPL aufgeführt und erläutert. In Abbildung 19 ist das Event Modell für die Ausführung von DM-Aktivitäten zu sehen. Darin enthalten sind die verschiedenen WS-BPEL Events. Auf diesem Modell aufbauend wird analysiert welche Events in ODE hinzugefügt werden müssen.

Aus den Events im BPEL Event Modell der Ausführung folgt, dass zwei neue Event Klassen hinzugefügt werden müssen. Dies sind die Event Klassen `Connection_Events` und `DM_Events`. Jede dieser beiden Event Klassen verfügt über eine Anzahl von einzelnen Events, die nachfolgend genauer erläutert werden.

`Connection_Events`

- `Connection_Failure_Event`
- `Connection_Active_Event`
- `Connection_Closed_Event`
- `Connection_TimeOut_Event`

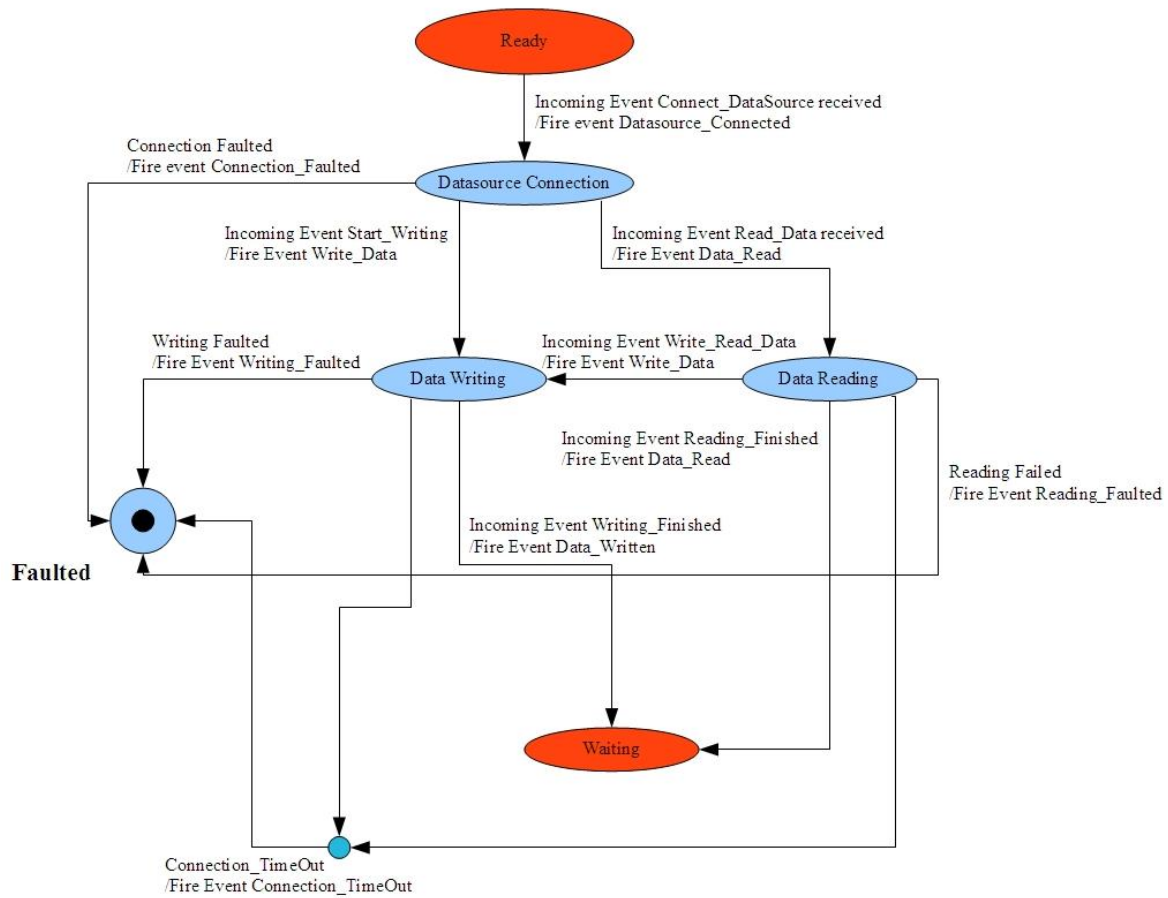


Abbildung 19: Event-Modell für die Ausführung von DM-Aktivitäten

DM_Events

- DM_Finished
- DM_Started
- DM_Failure

Dabei sind Connection_Events für Ereignisse bei der Verbindung zu einer Datenquelle zuständig. Wie die Namen der Events bereits zeigen, gibt es ein Event für:

- einen fehlgeschlagenen Verbindungsversuch
- einen erfolgreichen Verbindungsaufbau
- das Beenden einer Verbindung zu einer Datenquelle
- einen unerwarteten Abbruch der Verbindung

Die DM_Events sind für die Ereignisse einer DM-Aktivität zuständig. Hier wird nicht getrennt, um welche Art von DM-Aktivität es sich handelt. Für alle DM-Aktivitäten werden die selben Events erzeugt. Diese Events enthalten jeweils Informationen über den Namen und den Typ der Aktivität. Auch hier zeigen bereits die Namen an welche Events es gibt:

- die Ausführung einer DM-Aktivität wurde gestartet
- die Ausführung einer DM-Aktivität wurde erfolgreich beendet
- bei der Ausführung einer DM-Aktivität trat ein Fehler auf

8 Technologien und Werkzeuge

In diesem Kapitel folgt ein Überblick über die im Projekt verwendeten Technologien und Werkzeuge.

8.1 Technologien

In der Entwicklung von SIMPL werden die folgenden Technologien zum Einsatz kommen:

Apache Axis2 [6]

Axis2 ist eine SOAP-Engine zur Konstruktion von darauf basierenden Web Services und Client-Anwendungen.

Apache ODE [7]

Apache ODE ist eine BPEL-Workflow-Engine. Es wird sowohl WS-BPEL 2.0 als auch BPEL4WS 1.1 unterstützt, und die Ausführung von Prozessen in SOA ist möglich. Zudem ist ein Deployment von Prozessen zur Laufzeit (Hot Deployment) möglich, als auch die Analyse und Validierung von Prozessen.

Apache Tomcat [8]

Apache Tomcat ist ein Web Container für Web Services, der aber auch einen kompletten Webserver beinhaltet.

Apache Tuscany DAS [9]

Data Access Services ermöglichen den einheitlichen Zugriff auf Datenquellen, indem eine zentrale Schnittstelle für das Lesen und Schreiben von heterogenen Daten bereitgestellt wird. Die Daten werden dabei in Service Data Objects gekapselt und so von ihrer Quelle unabhängig gemacht.

Apache Tuscany SDO [10]

Service Data Objects stellen eine einheitliche API zur Verfügung, mit der die Handhabung verschiedener heterogener Daten und Datenquellen vereinfacht wird.

Eclipse BPEL Designer [11]

Der Eclipse BPEL Designer ist ein Eclipse Plugin für die Modellierung von BPEL-Prozessen. Er verfügt über eine leicht verständliche GUI und über eine Syntaxprüfung für BPEL-Prozesse. Zudem bietet der Eclipse BPEL Designer, in Kombination mit Apache ODE, eine Schritt für Schritt Ausführung von Prozessen.

IBM-DB2 [12]

Die IBM-DB2 ist ein Hybriddatenserver mit dem sowohl die Verwaltung von XML-Daten als auch von relationalen Daten möglich ist.

Java 6 [13]

Java ist eine objektorientierte Programmiersprache der Firma Sun Microsystems, mit der sich plattformunabhängige Programme entwickeln lassen.

JAX-WS [14]

JAX-WS ist eine Java API zur Erstellung von Webservices, die für das Deployment Java Annotationen verwendet.

TinyDB [15]

TinyDB ist ein Datenbanksystem für Sensornetze und bietet eine SQL-ähnliche Schnittstelle.

Webservices

Webservices sind abgeschlossene Software-Einheiten, die sich selbst beschreiben und die ihre Dienste über ein Netzwerk, wie beispielsweise das Internet, bereitstellen. Damit werden verteilte Anwendungen möglich, die flexibel auf sich ändernde Anforderungen angepasst werden können. Die Schnittstellen von Webservices werden mit WSDL beschrieben und können damit unabhängig von Betriebssystem, Plattform und Programmiersprache verwendet werden.

8.2 Werkzeuge

Es werden folgende Werkzeuge eingesetzt, um den Entwicklungsvorgang und die Dokumentation zu unterstützen:

Apache Maven [16]

Maven ist ein Projekt-Management-Tool zur standardisierten Erstellung und Verwaltung von Java-Programmen. Mit Maven ist es möglich viele Schritte, die die Entwickler normalerweise von Hand erledigen müssen, zu automatisieren.

Eclipse [17]

Eclipse ist eine IDE für Java und auch andere Programmiersprachen. Für SIMPL wird Eclipse in der Version 3.5 (Galileo) verwendet.

Hudson [18]

Hudson ist ein webbasiertes System zur kontinuierlichen Integration von Softwareprojekten.

L^AT_EX [19]

L^AT_EX ist ein Textverarbeitungsprogramm, mit dem es möglich ist, auf einfache Art und Weise L^AT_EX-Dokumente zu erstellen.

PDF-XChange Viewer [20]

Mit dem PDF-XChange Viewer lassen sich PDF-Dateien nicht nur öffnen, lesen und drucken, sondern zusätzlich Kommentare, Notizen und Markierungen vornehmen.

Subversion [21]

Subversion ist eine Software zur Versionskontrolle und eine Weiterentwicklung von CVS. Es wird genutzt um mehreren Nutzern den gleichzeitigen Zugriff und ein gleichzeitiges Bearbeiten von verschiedenen Dateien und Dokumenten zu ermöglichen.

Literatur

- [1] SIMPL Angebot, Version 1.2, 25.September 2009
- [2] Vrhovnik, M.; Schwarz, H.; Radeschütz, S.; Mitschang, B.: An Overview of SQL Support in Workflow Products. In: Proc. of the 24th International Conference on Data Engineering (ICDE 2008), Cancún, México, April 7-12, 2008
- [3] Wieland, M.; Görlach, K.; Schumm, D.; Leymann, F.: Towards Reference Passing in Web Service and Workflow-based Applications. In: Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009)
- [4] W3C. Web Services Addressing 1.0 - Core, W3C Recommendation. <http://www.w3.org/TR/ws-addr-core/>, 2006.
- [5] Web Services Business Process Execution Language Version 2.0, <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>, 15.10.2009
- [6] Axis2, <http://ws.apache.org/axis2/>, 17.10.2009
- [7] Apache ODE, <http://ode.apache.org/>, 17.10.2009
- [8] Apache Tomcat, <http://tomcat.apache.org/>, 17.10.2009
- [9] Apache Tuscany DAS, <http://tuscany.apache.org/das-overview.html>, 17.10.2009
- [10] Apache Tuscany SDO, <http://tuscany.apache.org/sdo-overview.html>, 17.10.2009
- [11] Eclipse BPEL Designer, <http://www.eclipse.org/bpel/>, 17.10.2009
- [12] IBM-DB2, <http://www.ibm.com/db2/>, 17.10.2009

- [13] Java 6, <http://java.sun.com/>, 17.10.2009
- [14] JAX-WS, <https://jax-ws.dev.java.net/>, 17.10.2009
- [15] TinyDB, <http://telegraph.cs.berkeley.edu/tinydb/>, 17.10.2009
- [16] Apache Maven, <http://maven.apache.org/>, 18.10.2009
- [17] Eclipse, <http://www.eclipse.org/>, 18.10.2009
- [18] Hudson, <https://hudson.dev.java.net/>, 18.10.2009
- [19] L^AT_EX, <http://www.lyx.org/>, 18.10.2009
- [20] PDF-XChange Viewer, <http://pdf-xchange-viewer.softonic.de/>, 18.10.2009
- [21] Subversion, <http://subversion.tigris.org/>, 18.10.2009

Abkürzungsverzeichnis

API	Application Programming Interface
BPEL	Business Process Execution Language
DAS	Data Access Service(s)
DDL	Data Definition Language
DM	Data Management
FLWOR	FOR, LET, WHERE, ORDER, RETURN
IDE	Integrated Development Environment
IUD	INSERT, UPDATE, DELETE
ODE	Orchestration Director Engine
RDB	Relational Data Base
RRS	Reference Resolution System
SDO	Service Data Object(s)
SIMPL	SimTech: Information Management, Processes and Languages
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSO	Single Sign On
UDDI	Universal Description, Discovery and Integration
URI	Unified Resource Identifier
URL	Unified Resource Locator
WS	Web Service
WSDL	Web Service Description Language
XML	Extensible Markup Language
XQUERY	XML Query Language

Abbildungsverzeichnis

1	Übersicht über die Systemumgebung von SIMPL	7
2	SIMPL Menü und Eclipse BPEL Designer mit einigen Data-Management-Aktivitäten	10
3	Admin-Konsole	11
4	Dialog der globalen Eigenschaften	11

5	Dialog für die Auditing Einstellungen	12
6	Dialog für allgemeine Referenzen Einstellungen	13
7	Dialog für die Verwaltung von Referenzen	13
8	Dialog für das Anlegen neuer Referenzen	14
9	Dialog zur Bearbeitung von Referenzen	15
10	Eigenschaftsfenster einer Data-Management-Aktivität am Beispiel einer Query Activity	16
11	Anwendungsfall-Diagramm des gesamten Softwaresystems	17
12	Anwendungsfall-Diagramm für den Prozess-Modellierer	18
13	Anwendungsfall-Diagramm für den Workflow-Administrator	20
14	Anwendungsfall-Diagramm für die ODE Workflow-Engine	24
15	Anwendungsfall-Diagramm für den Eclipse BPEL Designer	26
16	Anwendungsfall-Diagramm für die Prozess-Modellierer	26
17	Die Architektur des Reference Resolution Systems (RRS).	29
18	Alternativen zur Realisierung von Referenzen in BPEL	31
19	Event-Modell für die Ausführung von DM-Aktivitäten	48