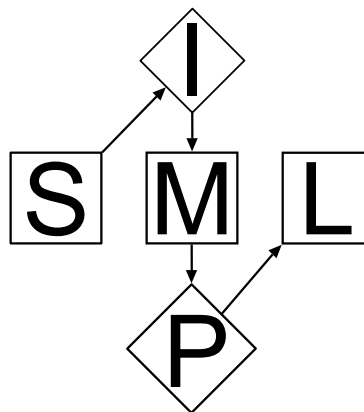


Grobentwurf

Version 1.5

07. Dezember 2009



Inhaltsverzeichnis

1	Einleitung	5
1.1	Zweck des Dokuments	5
1.2	Das SIMPL Rahmenwerk	5
1.3	Anforderungen	5
1.3.1	Einfache Modellierung	5
1.3.2	Große Datenmengen	5
1.3.3	Auditing von Prozessen	6
1.3.4	Late Binding von Datenquellen	6
1.3.5	Authentifizierung und Autorisierung	6
1.3.6	Registry	6
1.3.7	Admin-Konsole	6
1.3.8	Erweiterbarkeit	6
1.3.9	Verwendbarkeit	7
1.4	Entwurfsprinzipien	7
1.4.1	Offenes Rahmenwerk	7
1.4.2	Modularisierung	7
1.4.3	Kopplung und Zusammenhalt	7
1.4.4	Entwicklungsrichtung	7
1.4.5	Plug-In	8
1.4.6	Adapter	8
1.5	Überblick über den Grobentwurf	8
2	Architektur	9
2.1	Eclipse	9
2.2	Apache Tomcat	9
2.3	UDDI Registry	10
3	Komponenten	11
3.1	Eclipse	11
3.1.1	Eclipse Extension Points	12
3.1.2	Eclipse BPEL Designer Plug-In Extension Points	12
3.1.3	BPEL-DM Plug-In	13
3.1.4	SIMPL Core Plug-In	13
3.1.5	RRS Plug-In	13
3.2	Apache ODE	13
3.2.1	Schnittstellen	14
3.2.2	Komponenten	14
3.2.3	Erweiterungen	15
3.3	SIMPL Core	16
3.3.1	Administration Service	17
3.3.2	Datasource Service	17
3.3.3	Security Service	17
3.3.4	Strategy Service	18
3.3.5	Storage Service	18
3.4	Reference Resolution System (RRS)	18
3.4.1	RRS Service	18
3.4.2	RRS Administration Service	18
3.4.3	RRS Transformation Service	18
3.5	Apache jUDDI	19

4 Schnittstellen	20
4.1 SIMPL Core	20
4.1.1 AdministrationService (extern)	20
4.1.2 StorageService (intern und extern)	20
4.1.3 DatasourceService (extern)	20
4.1.4 StrategyService (intern)	20
4.1.5 SecurityService (intern)	20
4.1.6 ReferenceService	20
4.2 RRS	21
4.2.1 RRSRetrievalService	21
4.2.2 RRSAdministrationService	21
4.2.3 RRSTransformationService	21
Literaturverzeichnis	22
Abkürzungsverzeichnis	23
Abbildungsverzeichnis	24

Änderungsgeschichte

Version	Datum	Autor	Änderungen
0.1	29.10.2009	schneimi	Erstellung des Dokuments
0.2	30.10.2009	schneimi	Kapitel 1 erstellt
0.3	04.11.2009	schneimi	Kapitel 2, Kapitel 3.3 erstellt
0.4	07.11.2009	schneimi	Kapitel 4 erstellt
0.5	09.11.2009	rehnre	Kapitel 3.2 erstellt
0.6	09.11.2009	bruenedl	Kapitel 3.1 erstellt
1.0	11.11.2009	schneimi	Korrektur nach internem Review
1.1	20.11.2009	schneimi	Überarbeitung nach Kundenreview
1.2	05.12.2009	schneimi	Neue Schaubilder und Beschreibungen
1.3	07.12.2009	zoabi,bruenedl	Kapitel 3.1 überarbeitet
1.4	07.12.2009	rehnre	Kapitel 3.2 überarbeitet
1.5	07.12.2009	schneimi	Kapitel 4 überarbeitet

1 Einleitung

Dieses Kapitel soll dem Leser einen Überblick über das SIMPL Rahmenwerk geben und dessen Zweck, die damit verbundenen Anforderungen und die eingesetzten Entwurfsprinzipien erläutern. Die Struktur und der Aufbau des Dokuments orientieren sich dabei an der Entwurfsvorlage [1] von Markus Knauß.

1.1 Zweck des Dokuments

Der Grobentwurf beschreibt das Rahmenwerk auf Komponentenebene und bildet die Grundlage für den späteren Feinentwurf. Es werden alle wichtigen Komponenten und ihre Schnittstellen identifiziert sowie ihr Zusammenspiel beschrieben. Damit soll gezeigt werden, dass das resultierende System funktionieren kann und den Anforderungen gerecht wird. Damit die in späteren Iterationen dazukommenden Funktionalitäten schon beim Feinentwurf berücksichtigt werden, wird soweit es möglich ist, das Gesamtsystem beschrieben und inhaltlich nicht zwischen den einzelnen Iterationen unterschieden. Teile, die erst in späteren Iterationen spezifiziert werden können, werden explizit kenntlich gemacht.

1.2 Das SIMPL Rahmenwerk

Das SIMPL Rahmenwerk soll dem Benutzer eine erweiterbare Umgebung bieten, die eine einfache Modellierung von Simulationsworkflows auf Basis von BPEL mit generischem Zugriff auf beliebige Datenquellen ermöglicht. Bei den Datenquellen kann es sich beispielsweise um Datenbanken, Sensornetze oder Dateisysteme handeln. Die Modellierung der Prozesse findet in Eclipse mit dem Eclipse BPEL Designer Plug-In statt, das für diesen Zweck um zusätzliche Aktivitäten für den Zugriff auf Datenquellen und die Definition von **Datamangement-Operationen** innerhalb des Prozesses erweitert wird. Diese Aktivitäten werden im folgenden Verlauf des Dokuments als Data-Management-Aktivitäten bzw. DM-Aktivitäten bezeichnet. Die Ausführung der Prozesse erfolgt durch die Apache ODE Workflow Engine, die für diese Aktivitäten erweitert werden muss. Zusätzlich werden das bestehende Event-Modell und das Auditing der Prozessdaten für die DM-Aktivitäten angepasst. Die Dienste, die für die Ausführung der DM-Aktivitäten von der Workflow Engine benötigt werden, werden in Form von Web Services bereitgestellt. Durch ein weiteres Plug-In **für Eclipse** sollen außerdem Referenzvariablen bei der Modellierung von Prozessen unterstützt werden und diese durch eine entsprechende Modelltransformation auch auf anderen Workflow Engines einsetzbar bleiben.

1.3 Anforderungen


In diesem Abschnitt werden die Anforderungen des SIMPL Rahmenwerks beschrieben.

1.3.1 Einfache Modellierung

Neben der vereinfachten Modellierung der DM-Aktivitäten gibt es eine weitere Anforderung. Während der Modellierung von Prozessen wiederholen sich in der Regel häufig längere Befehle (Statements) oder Befehlsteile in Anfragesprachen wie SQL oder XQuery und auch längere Namen von Datenquellen oder Datencontainern (Tabellen, Dateien, XML-Dokumente, etc). Damit diese vom Prozess-Modellierer nicht jedes mal vollständig angegeben werden müssen, soll es die Möglichkeit geben, diese in BPEL-Variablen zu hinterlegen. Diese BPEL-Variablen können anschließend als Referenzen in anderen Befehlen verwendet werden. Durch die Modularisierung der Befehle sind insbesondere komplexe und geschachtelte Datenbankmanagementoperationen einfacher zu modellieren und die Komplexität wird reduziert.

1.3.2 Große Datenmengen

Der Schwerpunkt des Rahmenwerks liegt in der Modellierung von wissenschaftlichen Prozessen, bei denen überwiegend mit großen Datenmengen gearbeitet wird. Damit diese Datenmengen nicht inner-

halb des Prozesses gehalten werden müssen, wird ein Reference Resolution System (RRS) realisiert. Damit wird es möglich, Daten zu referenzieren,  per Referenz sehr schnell weiterzugeben und nur bei Bedarf aufzulösen und in den Prozess zu laden. ~~Weiterhin können auch~~ innerhalb der DM-Aktivitäten Datenquellen bzw. Datencontainer per Referenz angegeben werden.


1.3.3 Auditing von Prozessen

Beim Auditing von Prozessen handelt es sich um das Protokollieren von Prozessdaten, wie z.B. dem Status einer Variable oder aufgetretene Events, die unter Anderem ein Monitoring der Prozesse ermöglichen. Apache ODE besitzt bereits ein internes Auditing, bei dem Prozessdaten persistent gespeichert werden. Das interne Auditing muss für die zusätzlichen DM-Aktivitäten angepasst werden, und der Benutzer soll außerdem die Möglichkeit bekommen, die Granularität der Protokollierung zu bestimmen. Zusätzlich soll die interne Datenspeicherung auf eine beliebige externe Datenquelle umgeleitet werden können.

1.3.4 Late Binding von Datenquellen

Da bei der Modellierung nicht immer feststeht, auf welche Datenquelle zugegriffen wird, beispielsweise beim Ablegen von Daten, soll das Late Binding von Datenquellen unterstützt werden. Damit kann durch die Formulierung von Anforderungen an die Datenquelle und das Wählen einer Auswahlstrategie durch den Prozess-Modellierer eine passende Datenquelle automatisch zur Laufzeit bestimmt werden.


1.3.5 Authentifizierung und Autorisierung

Datenquellen erfordern in der Regel eine Authentifizierung und Autorisierung des Benutzers bei einem Zugriff. Das Rahmenwerk soll dem Prozess-Modellierer diesen Vorgang vereinfachen, so dass die dafür benötigten Informationen wie z.B. Benutzername und Passwort nicht bei jedem Zugriff erneut angegeben werden müssen. Dazu soll das Konzept des Single Sign On (SSO) angeboten bzw. unterstützt werden. (siehe [5], Kapitel 7.2.1) 


1.3.6 Registry

Für die Verwaltung der für den Prozess-Modellierer zur Verfügung stehenden Datenquellen, ist eine zentrale Registry vorgesehen, in der die Datenquellen von **Datenquellen-Administratoren** definiert werden können. Dort werden auch die entsprechenden Eigenschaften der Datenquellen hinterlegt, die für das Late Binding benötigt werden.

1.3.7 Admin-Konsole

Für alle wichtigen Einstellungen des Rahmenwerks soll eine Admin-Konsole bereitgestellt werden, über die der Workflow-Administrator Einstellungen treffen kann. Dies betrifft vor allem alle Einstellungen, die auch zur Laufzeit getätigt werden können, wie z.B. das Aktualisieren von Referenzen. 

1.3.8 Erweiterbarkeit

Das Rahmenwerk stellt folgende Anforderungen an die Erweiterbarkeit und soll dafür entsprechende Zugriffspunkte bereitstellen: 

- Anbindung weiterer Datenquellentypen und damit weiterer Konzepte für den Zugriff, sowie **Anfragesprachen**
- **Erweiterung der Funktionalität des Rahmenwerks** durch Integration neuer Dienste, die ggf. ~~zur~~ bereits vorhandenen ~~Dienste~~ Dienste benutzen
- **Unterstützung weiterer Anfragesprachen**

- Erweiterung um neue DM-Aktivitätstypen
- Erweiterung um neue Events für das Auditing
- Unterstützung weiterer Authentifizierungs- und Autorisierungsverfahren
- Unterstützung weiterer Registries für Datenquellen
- Erweiterung um zusätzliche Unterpunkte für die Admin-Konsole

1.3.9 Verwendbarkeit

Die Aktivitäten, die für SIMPL entwickelt werden, sollen auch auf anderen BPEL Workflow-Engines ausführbar sein. Dazu müssen die bei der Modellierung verwendeten Referenzen in standard-konformen BPEL-Code transformiert werden. Eine zusätzliche Anforderung ist, dass die von SIMPL bereitgestellten Web Services auf verschiedenen Web Containern lauffähig sein müssen.

1.4 Entwurfsprinzipien

In diesem Abschnitt werden die Prinzipien beschrieben, die für den Entwurf angewendet werden (vgl. [2], Kapitel 17).

1.4.1 Offenes Rahmenwerk

Das SIMPL Rahmenwerk ist größtenteils ein offenes Rahmenwerk. Offen bedeutet, es gibt keine einheitliche Schnittstelle für alle Erweiterungen. Daher können viele Erweiterungen nicht ohne technisches Verständnis und Wissen über die Mechanismen und Abläufe des Rahmenwerks realisiert werden. Die Erweiterungsmöglichkeiten werden daher ausführlich dokumentiert und Beispiele erstellt, mit denen die Umsetzung eigener Erweiterungen erleichtert wird. Bereiche des Rahmenwerks, bei denen im Laufe des Projekts ausreichend Erfahrung gesammelt wurde, werden wenn möglich in geschlossener Form für die Entwicklung bereitgestellt.

1.4.2 Modularisierung

Durch die Modularisierung werden die Komponenten in einfache und leicht verständliche Teile gegliedert. Die Realisierungsdetails eines Moduls werden nach dem Prinzip des Information Hiding versteckt und die Dienste nur über eine Schnittstelle angeboten. Ziel ist es, später Module ändern oder austauschen zu können, möglichst ohne dabei die Schnittstellen ändern zu müssen und damit Auswirkungen auf andere Module zu verursachen.

1.4.3 Kopplung und Zusammenhalt

Beim Entwurf der Module wird darauf geachtet, dass die Kopplung zu anderen Modulen möglichst gering ist, und dass der Zusammenhalt innerhalb eines Moduls möglichst hoch ist. Durch dieses Vorgehen wird eine hohe Lokalität und damit gute Wartbarkeit erreicht, da sich Fehler, die bei Änderungen entstehen, nicht im System fortpflanzen können.

1.4.4 Entwicklungsrichtung

Die Entwicklung wird Top-down durchgeführt. Dabei wird die Aufgabe des Rahmenwerks rekursiv bis zur elementaren Ebene (der Programmiersprache) in Teilaufgaben zerlegt und damit schrittweise verfeinert.

1.4.5 Plug-In

Plug-Ins sind externe Software-Einheiten, durch die das Rahmenwerk um zusätzliche Funktionalität erweitert werden kann. Das Rahmenwerk bietet dafür entsprechende Zugriffspunkte an, an denen die Plug-Ins angeschlossen werden können.

1.4.6 Adapter

Adapter bzw. Konnektoren sind interne Verbindungsstücke. Sie werden dort entwickelt, wo Komponenten angebunden werden sollen, deren Schnittstellen nicht zu den vorhandenen Schnittstellen des Rahmenwerks passen.

1.5 Überblick über den Grobentwurf

Der Grobentwurf gliedert sich in die folgenden weiteren Kapitel.

- Kapitel 2 “Architektur” beschreibt die Architektur des Rahmenwerks. Das Rahmenwerk wird in überschaubare Komponenten gegliedert, die jeweils genau definierte Funktionen erfüllen.
- Kapitel 3 “Komponenten” beschreibt die im Kapitel “Architektur” genannten Komponenten und definiert ihre Schnittstellen.
- Kapitel 4 “Schnittstellen” beschreibt die in Kapitel “Komponenten” definierten Schnittstellen, eingesetzten Protokolle und die übertragenen Daten.

2 Architektur

Abbildung 1 zeigt die Architektur des SIMPL Rahmenwerks. Das Zusammenspiel der übergeordneten Komponenten wird in den folgenden Abschnitten beschrieben, auf die Komponenten selbst und ihre Erweiterungen wird dann in Kapitel 3 eingegangen.

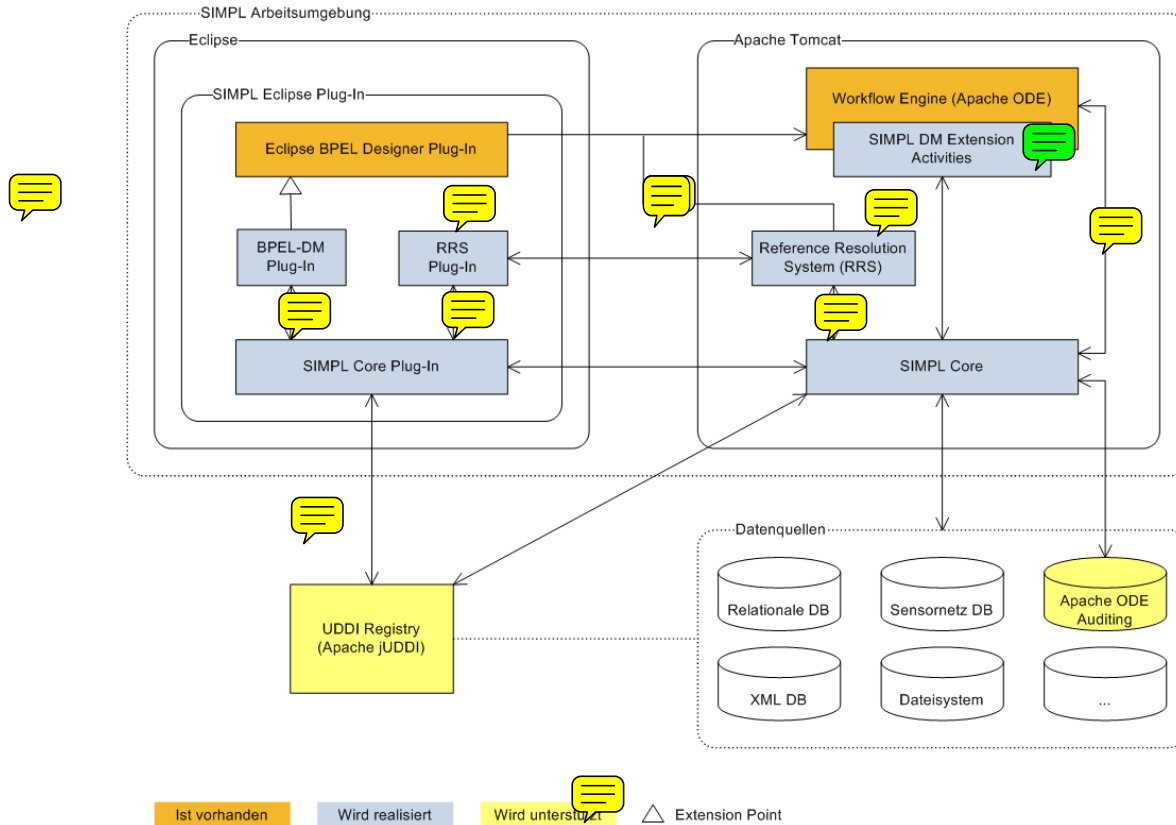


Abbildung 1: Architektur des SIMPL Rahmenwerks

2.1 Eclipse

Die Entwicklungsumgebung Eclipse bildet mit den entsprechenden Plug-Ins sowohl die GUI für den Prozess-Modellierer als auch den Workflow-Administrator. Mit dem Eclipse BPEL Designer Plug-In kann der Prozess-Modellierer bereits BPEL Prozesse erstellen und auf Apache ODE zum Einsatz bringen (deployen). Das SIMPL Eclipse Plug-In erweitert mit dem BPEL-DM Plug-In die bestehenden Aktivitäten des Eclipse BPEL Designer Plug-Ins um die DM-Aktivitäten. Die GUI für den Workflow-Administrator wird vom SIMPL Core Plug-In in Form einer Admin-Konsole bereitgestellt. (siehe [5], Kapitel 4). Mit dem RRS Plug-In wird die Verbindung zum Reference Resolution System (RRS) geschaffen. Dadurch können Referenzen verwaltet werden und falls Referenzen bei der Modellierung verwendet wurden, eine Modelltransformation vor dem Deployment durchgeführt werden, damit die Prozesse auch auf anderen Workflow Engines zum Einsatz kommen können.

2.2 Apache Tomcat

Der Web Container Apache Tomcat ist die Laufzeitumgebung für Apache ODE und den SIMPL Core. Apache ODE ist für die Ausführung der Prozesse nach der Modellierung zuständig und benötigt, wie

bereits in Kapitel 1.2 erwähnt, bestimmte Dienste für die Ausführung der zusätzlichen DM-Aktivitäten in den Prozessen wie z.B. das Ausführen eines Befehls auf einer Datenquelle. Diese Dienste werden vom SIMPL Core bereitgestellt, der sich aus nach Aufgaben eingeteilten Web Services zusammensetzt. Zusätzlich werden für die Umsetzung des Reference Resolution Systems (RRS) weitere Dienste bereitgestellt, die den SIMPL Core ergänzen. Das RRS verwaltet die Referenzen über den SIMPL Core, der seinerseits das RRS für das Late Binding benötigt. Das Auditing von Apache ODE wird über den SIMPL Core auf einer beliebigen externen Datenquelle ermöglicht.

2.3 UDDI Registry

Eine UDDI Registry ermöglicht eine zentrale Verwaltung von Datenquellen, die zur Modellierung bereitgestellt werden. Dort können auch Eigenschaften und Anforderungen zu Datenquellen abgelegt werden, die für das Late Binding benötigt werden.

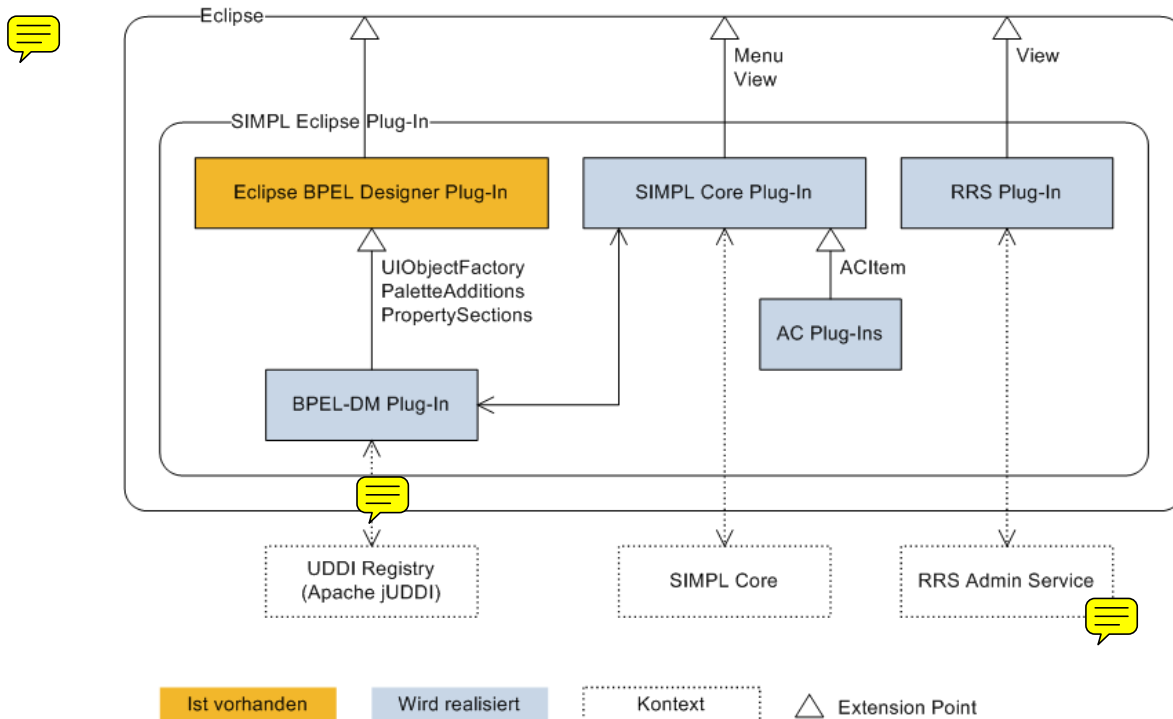


Abbildung 3: Eclipse mit SIMPL Erweiterungen

3.1.1 Eclipse Extension Points

Eclipse bietet über Extension Points die Möglichkeit, die IDE zu erweitern. Für das SIMPL Projekt nutzen wir den “Menu” - und “View” Extension Point, den Eclipse zur Verfügung stellt.

Menu Das Menu stellt die Auswahlmöglichkeiten als Drop-Down Liste dar, die auf die verschiedenen Funktionsbereiche des Plug-Ins referenziert. Der Extension Point “Menu” wird im Rahmen von SIMPL dazu verwendet, um ein SIMPL-Menü in Eclipse und später auch zusätzliche Einträge bereitstellen zu können.

View Die View ist eine Art Fenster zur Darstellung und Eingabe von Daten innerhalb von Eclipse, wie z.B. ein “ErrorLog”, der alle Fehler und Probleme von Eclipse Projekten anzeigt oder eine PropertiesView in der z.B. BPEL-Aktivitäten näher spezifiziert werden. Im Rahmen von SIMPL wird z.B. eine View verwendet, um die Referenzen für das RRS Plug-In anzuzeigen bzw. zu verwalten.

3.1.2 Eclipse BPEL Designer Plug-In Extension Points

Es wird auf die unten genannten drei Extension Points eingegangen, weil nur diese vom BPEL Designer benutzt werden. Es gibt auch noch weitere Extension Points im BPEL Designer Plug-In, diese sind jedoch nicht von Relevanz für SIMPL.

UIObjectFactory Dieser Extension Point ermöglicht das Anbinden neuer UIObjectFactories, mit deren Hilfe Instanzen von BPEL Extension Activities erzeugt werden können. Die so erzeugten Instanzen stehen dann in der BPEL Designer Palette zur Verfügung und können zur Modellierung von BPEL-Prozessen verwendet werden.

PaletteAdditions Die Palette ist ein Auswahlfenster aller zur Verfügung stehenden BPEL-Konstrukte, dies sind z.B. Standard BPEL-Aktivitäten oder eigene Extension Activities. Durch diesen Extension Point können eigene Extension Activities in die Palette eingefügt und so zur Modellierung von Prozessen genutzt werden.

PropertySections Dieser Extension Point ermöglicht das Erweitern von Property-Fenstern, über die in das Prozessmodell eingefügte BPEL-Aktivitäten ausformuliert und mit Werten gefüllt werden können.

3.1.3 BPEL-DM Plug-In

Das BPEL-DM Plug-In beinhaltet die Data-Management-Aktivitäten, die die Standard BPEL-Aktivitäten des BPEL Designer Plug-Ins erweitern. Die Erweiterung der Aktivitäten geschieht über folgende drei Extension Points:

- UIObjectFactory
- PaletteAdditions
- PropertySections

Weiterhin werden durch das BPEL-DM Plug-In PropertySections für alle Data-Management-Aktivitäten bereitgestellt, die Zugriff auf einen Editor zur graphischen Modellierung von Datenquellenbefehlen bieten.

3.1.4 SIMPL Core Plug-In

Das SIMPL Core Plug-In erweitert die Eclipse IDE um das SIMPL-Menü und die SIMPL-Hilfe. Das SIMPL-Menü beinhaltet die Menüpunkte "Admin-Console", "Settings", "Help" und "About", die auf entsprechende Fenster verlinken. Das SIMPL Core Plug-In stellt den Extension Point "Admin Console Item" (ACItem) bereit, mit Hilfe dessen die Admin-Konsole um weitere Elemente erweitert werden kann.

3.1.5 RRS Plug-In

Das RRS Plug-In wird über den View-Extension-Point in Eclipse angebunden. Es wird eine GUI-Schnittstelle für den RRS Administration Service angeboten. So können über Eclipse mit dem RRS Plug-In alle Referenzen des angegebenen RRS verwaltet werden.

3.2 Apache ODE

Hier wird der Aufbau und die einzelnen Bestandteile von ODE, die für das SIMPL Rahmenwerk von Bedeutung sind, erläutert. Zunächst wird kurz Beschreibung über den derzeitigen Zustand von ODE geboten und im Anschluss folgt eine Beschreibung der Erweiterungen, die notwendig sind. In 4 werden alle wichtigen Bestandteile dargestellt.

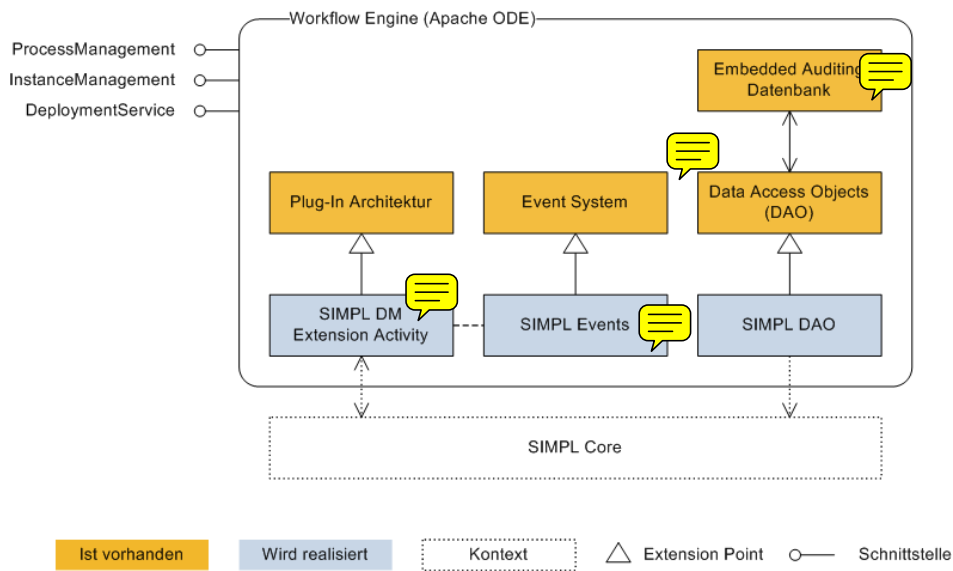


Abbildung 4: Apache ODE mit SIMPL Erweiterungen

3.2.1 Schnittstellen

In diesem Teil folgt eine kurze Beschreibung der Schnittstellen, über die die ODE Engine verfügt. Diese werden hier nur aufgelistet, aber voraussichtlich nicht verwendet.

ProcessManagement und InstanceManagement (Management API) Die Management API von Apache ODE bietet Funktionen auf Prozess- und auf Instanzebene an. Dadurch können z.B. bestimmte Nachrichten an Prozesse geschickt werden, um deren Ausführung zu starten oder zu beenden, und es können Informationen zu den einzelnen Instanzen abgerufen werden.

Die Management API unterteilt sich in Instanz-Management und Prozess-Management. Beide bieten verschiedene Funktionen für Instanzen bzw. Prozesse an. Es besteht auf der einen Seite die Möglichkeit, verschiedene Informationen über die Prozesse und Instanzen zu erhalten, zum Beispiel eine Liste der verschiedenen Prozessmodelle oder Informationen über ein einzelnes Prozessmodell, auf der anderen Seite ist es aber auch möglich, direkt mit Prozessinstanzen zu interagieren. Dadurch ist es beispielsweise möglich, eine Prozessinstanz anzuhalten oder direkt zu terminieren.

DeploymentService Der Deployment Service bietet verschiedene Funktionen für das Deployment von Prozessen an. Dazu gehören neben einer Deploy- und Undeploy-Funktion unter Anderem auch das Auflisten von deployten Prozessen. In den nächsten Iterationen kommen noch weitere Punkte hinzu, die das Deployment betreffen.

3.2.2 Komponenten

Hier werden die relevanten Komponenten von Apache ODE beschrieben.

Plug-In Architektur ODE verfügt über eine Plug-In-Architektur, mit der es unter anderem möglich ist, Extension Activities zu erstellen, die die Standardaktivitäten von BPEL erweitern. Dazu werden spezielle Interfaces zur Verfügung gestellt. Dabei handelt es sich um "AbstractSyncExtensionOperation" bei synchronen Aktivitäten und um "AbstractAsyncExtensionOperation" bei asynchronen Aktivitäten. Um diese Aktivitäten der ODE Engine hinzuzufügen, ist es notwendig, diese als jar-Dateien in

den Classpath zu kopieren. Anschließend ist es nur noch notwendig die Extension in der ODE Engine zu registrieren. Dies geschieht in der ode-axis2.PROPERTIES Datei.

Event System Innerhalb des ODE Event Systems existieren bereits 5 Event-Typen. ~~Diese sind:~~

- Instance Life Cycle Events
- Activity Life Cycle Events
- Scope Handling Events
- Data Handling Events
- Correlation Events

Diese Events sind in einer Hierarchie organisiert.

Data Access Object (DAO) Das DAO ist für die persistente Datenhaltung aller Daten, die innerhalb von ODE erzeugt werden zuständig. Weiterhin werden mit Hilfe des DAOs die Auditing Daten in der Embedded Auditing Datenbank gespeichert. Das DAO bietet außerdem verschiedene Einstellungen für die Auditing Datenbank an. So ist es mit Hilfe des DAOs wie bereits oben erläutert möglich das Auditing auf eine externe Datenbank umzuleiten. Es existiert zum Beispiel die Möglichkeit eine MySQL Datenbank zu nutzen.

Embedded Auditing Datenbank Die Auditingdaten, die ODE erzeugt, werden momentan in einer Embedded Datenbank gespeichert. Dies ist eine Apache Derby Datenbank, die sowohl Hibernate als auch JPA unterstützt. Welche Art der Datenbank genutzt wird, kann durch ein entsprechendes DAO eingestellt werden. Es ist weiterhin möglich statt der embedded Datenbank eine externe Datenbank zu nutzen, in dem dies in einem DAO und der axis2-properties-Datei eingestellt wird.

3.2.3 Erweiterungen

In den folgenden Abschnitten wird beschrieben, welche Erweiterungen für Apache ODE benötigt werden und wie diese realisiert werden.

SIMPL-DM Extension Activities Extension Activities für die Apache ODE Workflow Engine werden als Java Klassen erstellt, welche die Implementierung der neuen Aktivitäten enthalten. Weiterhin ist es notwendig, die unter Plug-in Architektur genannten Interfaces zu implementieren. Bei der Ausführung der DM-Aktivitäten wird der SIMPL Core genutzt, um die Datenmanagement-Operationen zu realisieren.

SIMPL Events Für das Auditing der DM-Aktivitäten ist es notwendig, dass eine Reihe von eigenen Events erstellt wird. Dies geschieht durch Erstellung neuer Event-Klassen, die von den bestehenden Event Klassen abgeleitet und anschließend innerhalb der Engine eingebunden werden. Diese neuen Events werden unterteilt in DM-Events und Connection-Events. DM-Events betreffen hierbei die Ausführung der DM-Aktivitäten, während Connection-Events Rückmeldung über die Verbindung zu einer Datenquelle liefern. Die einzelnen Events sind:

DM-Events:

- DMStarted
- DMFailure
- DMEnd

Connection-Events:

- ConnectionStarted
- ConnectionLost
- ConnectionEnd

Bei der Implementierung werden DM-Events und Connection-Events jeweils als ein neuer Event-Typ implementiert und die einzelnen spezifischen Events werden von diesen abgeleitet. Dies bietet die Möglichkeit die Granularitätseinstellungen innerhalb des Deployment-Deskriptors ebenfalls um diese Typen zu erweitern. Die Events werden direkt in den Extension Activities eingebunden und werden hier auch ausgelöst. Es ist nicht vorgesehen diese Events an einer anderen Stelle in ODE zu nutzen.

SIMPL DAO Das SIMPL DAO implementiert die Schnittstellen des “org.apache.ode.bpel.dao”-Pakets, womit die persistente Datenspeicherung von Apache ODE auf den SIMPL Core umgeleitet werden kann. Mit Hilfe des SIMPL Cores können die Daten auf beliebigen Datenquellen gespeichert werden.



3.3 SIMPL Core

Der SIMPL Core, dargestellt in Abbildung 5, stellt die Funktionalität zur Verfügung, die während der Modellierung und der Ausführung von Prozessen benötigt wird. Er bietet nach Außen verschiedene Web Services zu Verfügung, die jeweils bestimmte Aufgaben innerhalb des SIMPL Rahmenwerks haben. Der SIMPL Core wird als JAR-Datei im Classpath von Apache ODE hinterlegt, dadurch wird eine schnelle direkte Kommunikation zwischen dem SIMPL Core und Apache ODE ermöglicht. Desweiteren kann der SIMPL Core als Singleton betrieben werden und es können damit z.B. Verbindungen zu Datenquellen für Folgezugriffe aufrecht gehalten werden. Die Web Services des SIMPL Cores werden dabei innerhalb der Axis2 Installation von Apache ODE bereitgestellt und zusätzlich mit der Java API JAX-WS annotiert, damit ein Einsatz auch mit anderen Web Containern außer Apache Tomcat möglich ist (siehe Anforderung 1.3.9).

In den folgenden Abschnitten werden zunächst die Web Services des SIMPL Core und ihre Aufgaben näher beschrieben. Die Schnittstellen und ihre Verwendung werden in Kapitel 4 weiter ausgeführt.

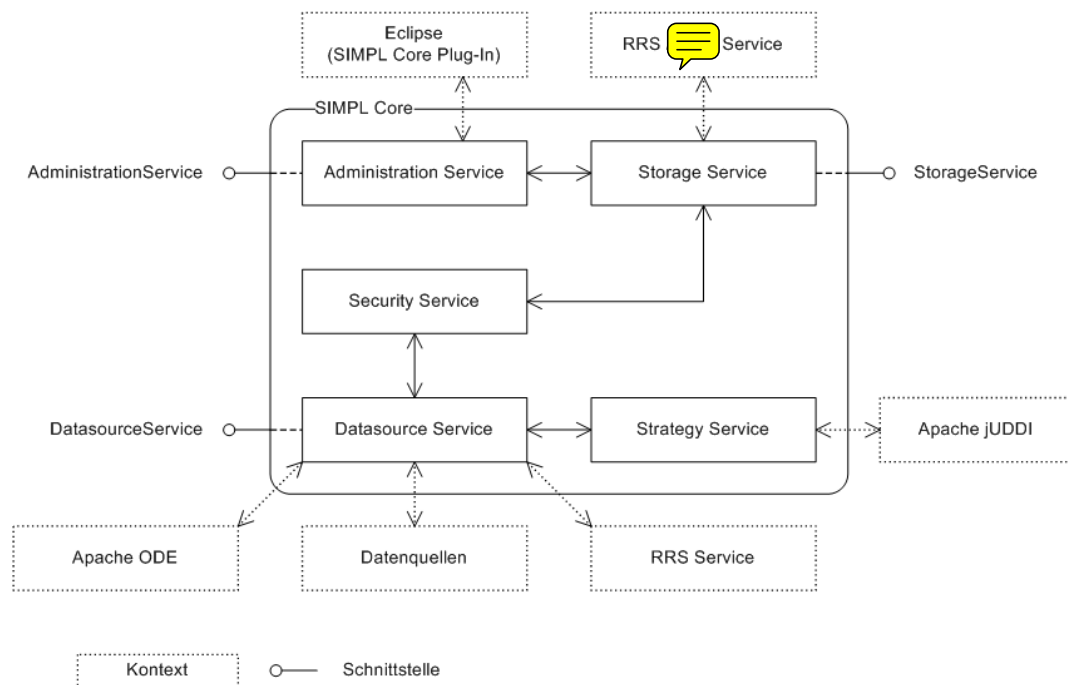


Abbildung 5: SIMPL Core

3.3.1 Administration Service

Über den Administration Service wird die Funktionalität für die Admin-Konsole (siehe Anforderung 1.3.7) in Eclipse bereitgestellt. Die **Einstellungen** der Admin-Konsole werden über den Storage Service verwaltet. Für Einstellungen, mit denen zur Laufzeit Einfluss auf Apache ODE genommen wird, wie z.B. die Granularität des Auditings, **werden die vorhandenen Schnittstellen von Apache ODE verwendet** und, falls nötig, neue Schnittstellen geschaffen. Die geforderte austauschbare GUI (siehe Anforderung 1.3.8) wird durch die **WSDL-Schnittstelle** erreicht.

3.3.2 Datasource Service

Der Datasource Service ist für alle Aufgaben zuständig, die den Zugriff auf die Datenquellen betreffen. Dort werden entsprechende Adapter (siehe 1.4.6) in Form von Direct Access Services (DAS) implementiert, die den Zugriff auf verschiedene Typen von Datenquellen ermöglichen und die Erweiterbarkeit für weitere Typen und Anfragesprachen (siehe Anforderung 1.3.8) garantieren. Des Weiteren werden Plug-In-Schnittstellen (siehe 1.4.5) für zusätzliche funktionale Erweiterungen geschaffen, wie z.B. die Unterstützung verschiedener Dateitypen bei Dateisystemen. ~~Referenzen werden vom RRS Service über den Datasource Service aufgelöst.~~

3.3.3 Security Service

Der Security Service ist zuständig für die Verwaltung von Informationen zur Authentifizierung und Autorisierung gegenüber Datenquellen. Mit diesem Service können ggf. auch Authentifizierungs- und Autorisierungsinformationen in andere Formate transformiert werden, wie beispielsweise Benutzername und Passwort nach SAML. Die Informationen werden über den Storage Service **verwaltet**, um das Konzept des SSO (siehe Anforderung 1.3.5) zu realisieren, bei dem diese Informationen wiederverwendet werden. Die genaue Rolle des Security Service wird erst in einer späteren Iteration **spezifiziert**.

3.3.4 Strategy Service

Der Strategy Service wird vom DataSourceService für das Late Binding (siehe Anforderung 1.3.4) verwendet. Dort stehen verschiedene Auswahlstrategien bzw. -algorithmen zur Verfügung, um mit im Prozess formulierten Anforderungen eine passende Datenquelle ausfindig zu machen. Die Datenquellen zur Auswahl werden von der UDDI Registry (Apache jUDDI) bereitgestellt. In einer späteren Iteration werden die Strategien "Erster Fund", "Bester Fund" und "Passender Fund" realisiert, die erst in einer späteren Iteration näher spezifiziert werden.

3.3.5 Storage Service

Der Storage Service verwaltet in einer embedded Datenbank alle Daten, die innerhalb des Rahmenwerks gespeichert werden müssen. Damit werden beispielsweise vom Administration Service die Einstellungen des SIMPL Rahmenwerks aus der Admin-Konsole gespeichert und können später wieder abgerufen werden. Auch andere Services können diesen Dienst zur Verwaltung von Daten benutzen, wie z.B. der Security Service zur Verwaltung von Authentifizierungs- und Autorisierungsinformationen.

3.4 Reference Resolution System (RRS)

Das RRS erfüllt die Anforderung, große Datenmengen in BPEL referenzieren zu können (siehe Anforderung 1.3.2). Wie in Abbildung 6 zu sehen ist, besteht es aus drei Web Services, die in den folgenden Abschnitten beschrieben werden.

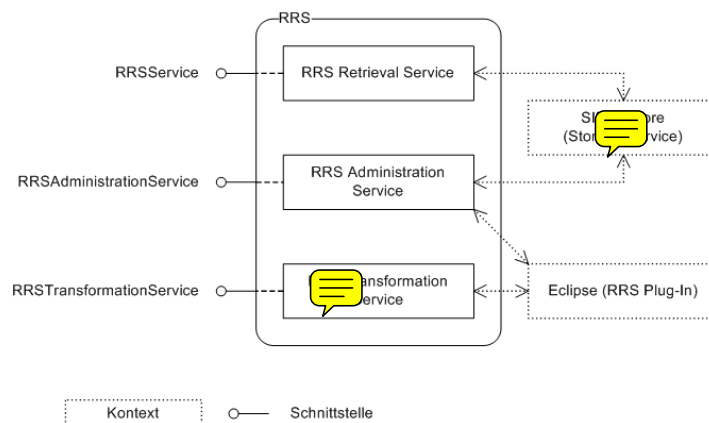


Abbildung 6: Reference Resolution System (RRS)

3.4.1 RRS Service

Über den RRS Service lassen sich Referenzen auflösen und die referenzierten Daten bei Bedarf in einen Prozess holen. Die Referenzen werden mit dem DataSourceService des SIMPL Core aufgelöst.

3.4.2 RRS Administration Service

Mit dem RRS Administration Service können Referenzen über das Eclipse RRS Plug-In verwaltet werden. Verwaltung geschieht dabei über den Storage Service vom SIMPL Core.

3.4.3 RRS Transformation Service

Damit modellierte Prozesse auch auf anderen Workflow Engines ausgeführt werden können (siehe Anforderung 1.3.9), wird der Transformation Service bereitgestellt, mit dem der um Referenzen erweiterte

BPEL Code in standard-konformen BPEL-Code transformiert werden kann. Die Transformation wird über das RRS-Plug-In vor dem Deployment mit dem Eclipse BPEL Designer Plug-In durchgeführt.

3.5 Apache jUDDI

Das SIMPL Rahmenwerk stellt mit Apache jUDDI bereits eine UDDI-Registry zur Verfügung, in der Datenquellen von Datenquellen-Administratoren verwaltet werden können. Das BPEL-DM Plug-In kann über die Registry Datenquellen für den Modellierer zur Verfügung stellen. Durch den UDDI Standard können auch beliebige andere UDDI-Registries angebunden werden. (siehe Anforderung 1.3.8)

4 Schnittstellen

In diesem Kapitel werden die Schnittstellen der Komponenten beschrieben, sowie eingesetzte Protokolle und übertragene Objekte erläutert. Bei allen Schnittstellen handelt es sich um WSDL-Schnittstellen, mit denen über ein Übertragungsprotokoll wie z.B. HTTP SOAP-Nachrichten ausgetauscht werden können. Die UDDI Schnittstelle der Registry Apache jUDDI wird direkt genutzt und bietet entsprechende Funktionen, die in [7] spezifiziert sind.

4.1 SIMPL Core

Hier werden die externen und wichtige interne Schnittstellen des SIMPL Rahmenwerks beschrieben. (siehe 5)

4.1.1 AdministrationService (extern)

Die AdministrationService-Schnittstelle bietet Funktionen für das Speichern, Löschen und Laden von Einstellungen der Admin-Konsole. Die Einstellungen werden dabei als SDO übertragen.

4.1.2 StorageService (intern und extern)

Die StorageService-Schnittstelle bietet Funktionen zur Verwaltung von Daten innerhalb des Rahmenwerks. Die Daten werden als SDO übertragen.

4.1.3 DatasourceService (extern)

Die DatasourceService-Schnittstelle wird extern von Apache ODE für den Zugriff auf Datenquellen und das Speichern der Auditing-Daten verwendet. Sie bietet Funktionen zum Abrufen der Daten von Datenquellen und Senden von Befehlen in unterstützten Anfragesprachen. Die Daten werden dabei als Service Data Object (SDO) [4] übertragen.

4.1.4 StrategyService (intern)

Die StrategyService-Schnittstelle wird vom Datasource Service für das Late Binding genutzt und bietet Funktionen für die verschiedenen Auswahlstrategien (siehe 3.3.4). Die nichtfunktionalen Anforderungen an die Datenquellen werden dabei als WS-Policy-Objekte übertragen, andere funktionale Eigenschaften als XML-Struktur.

4.1.5 SecurityService (intern)

Die SecurityService-Schnittstelle bietet Funktionen für die Verwaltung von Authentifizierungs- und Autorisierungsinformationen. Sie wird vom Datasource Service benötigt, um entsprechende Informationen für einen Zugriff auf eine Datenquelle abzurufen. Die Informationen werden dabei hauptsächlich in der Security Assertion Markup Language (SAML) und der eXtensible Access Control Markup Language (XACML) übertragen.

4.1.6 ReferenceService

Die ReferenceService-Schnittstelle wird bei der Modellierung in Eclipse zur Verwaltung von Referenzen genutzt und von Apache ODE zur Auflösung der Referenzen zur Laufzeit verwendet. Sie bietet Funktionen zur Verwaltung und Auflösung von Referenzen, wobei die Daten der aufgelösten Referenzen als SDO zurückgeliefert werden.

4.2 RRS

In den folgenden Abschnitten werden die Schnittstellen des RRS beschrieben. (siehe 6) Die nachfolgenden Beschreibungen beziehen sich auf ~~Quelle~~ [6] und werden hier nicht im Detail erläutert.

4.2.1 RRSRetrievalService

Die RRSRetrievalService-Schnittstelle bietet eine Funktion, mit der Referenzen aufgelöst werden können. Dazu wird eine Endpoint Reference (EPR) übergeben und der damit verbundene Wert aus einer Datenquelle ausgelesen und als SDO zurückgeliefert.

4.2.2 RRSAdministrationService

Diese Schnittstelle bietet Funktionen zum Speichern, Löschen und Ändern von Referenzen in Form von EPRs.

4.2.3 RRSTransformationService

Die RRSTransformationService-Schnittstelle bietet eine Funktion zur Umwandlung des durch Referenzen erweiterten BPEL-Codes in standard-konformen BPEL-Code. Dazu werden die entsprechenden BPEL-Dateien übergeben und transformierte Kopien der Prozessmodelle zurückgeliefert. Während der Transformation wird das Prozessmodell entsprechend den Beschreibungen in [6] erweitert.

Literatur

- [1] Knauß, Markus: *Entwurfsvorlage*. Institut für Softwaretechnologie, März 2008. <http://www.iste.uni-stuttgart.de/se/>, Abruf: 06.12.2009
- [2] Ludewig, Jochen; Lichter, Horst: *Software Engineering*. Grundlagen, Menschen, Prozesse und Techniken. dpunkt.verlag GmbH, 2007.
- [3] Alves, Alexandre; Arkin, Assaf; Askary, Sid; Barreto, Charlton; Bloch, Ben; Curbera, Francisco; Ford, Mark; Goand, Yaron; Guízar, Alejandro; Kartha, Neelakantan; Liu, Canyang Kevin; Khalaf, Rania; König, Dieter; Marin, Mike; Mehta, Vinkesh; Thatte, Satish; van der Rijn, Danny; Yendluri, Prasad; Yiu, Alex: *Web Services Business Process Execution Language Version 2.0*. Organization for the Advancement of Structured Information Standards OASIS, 11. April 2007. <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>, Abruf: 11.11.2009
- [4] Adams, Matthew; Andrei, Cezar; Barack, Ron; Blohm, Henning; Boutard, Christophe; Brodsky, Stephen; Budinsky, Frank; Bünnig, Stefan; Carey, Michael; Doughan, Blaise; Grove, Andy; Halaseh, Omar; Harris, Larry; von Mersewsky, Ulf; Moe, Shawn; Nally, Martin; Preotiuc-Pietro, Radu; Rowley, Mike; Samson, Eric; Taylor, James; Thiefaine, Arnaud: *Service Data Objects For Java*. Version: 2.1.0, November 2006. <http://www.osoa.org/download/attachments/36/Java-SDO-Spec-v2.1.0-FINAL.pdf>, Abruf: 11.11.2009
- [5] *Spezifikation v1.1*. Stupro-A SIMPL (2009).
- [6] 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)
- [7] Clement, Luc; Hately, Andrew; von Riegen, Claus; Rogers, Tony: *UDDI Version 3 Specification-OASIS Standard*. Organization for the Advancement of Structured Information Standards OASIS, 19. Oktober 2004. <http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3>, Abruf: 07.12.2009

Abkürzungsverzeichnis

API	Application Programming Interface
BPEL	Business Process Execution Language
DAO	Data Access Object
DAS	Data Access Service
DM	Data-Management
EPR	Endpoint Reference
GEF	Graphical Editing Framework
GUI	Graphical User Interface
JAX-WS	Java API for XML - Web Services
ODE	Orchestration Director Engine
RRS	Reference Resolution System
SAML	Security Assertion Markup Language
SDO	Service Data Object
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
WS	Web Service
WSDL	Web Service Description Language
XACML	eXtensible Access Control Markup Language
XQuery	XML Query Language

Abbildungsverzeichnis

1	Architektur des SIMPL Rahmenwerks	9
2	Komponenten des SIMPL Rahmenwerks	11
3	Eclipse mit SIMPL Erweiterungen	12
4	Apache ODE mit SIMPL Erweiterungen	14
5	SIMPL Core	17
6	Reference Resolution System (RRS)	18