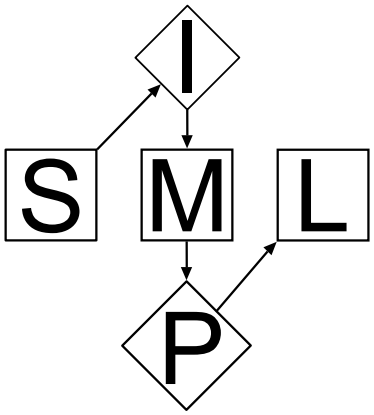
Studienprojekt SIMPL

Grobentwurf

Version 2.3

11. Juli 2010



Inhaltsverzeichnis

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	bruededl	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,bruededl	Kapitel 3.1 überarbeitet
1.4	07.12.2009	rehnre	Kapitel 3.2 überarbeitet
1.5	07.12.2009	schneimi	Kapitel 4 überarbeitet
1.6	21.02.2010	schneimi	Kapitel 1, 2, 3.0, 3.3, 3.4, 4 überarbeitet (091207-Grobentwurf-comments_PR_MS.pdf)
1.7	22.02.2010	bruededl	Kapitel 3.1 überarbeitet (091207-Grobentwurf-comments_PR_MS.pdf)
1.8	22.02.2010	rehnre	Kapitel 3.2 überarbeitet (091207-Grobentwurf-comments_PR_MS.pdf)
1.9	22.02.2010	huettiwg	Kapitel 3.2.2 und 3.2.3 überarbeitet (091207-Grobentwurf-comments_PR_MS.pdf)
2.0	14.06.2010	schneimi	Kapitel 1 und 2 überarbeitet (100222-Grobentwurf-v03-comments_PR_MS.pdf)
2.1	15.06.2010	schneimi	Kapitel 3 Einleitung überarbeitet (100222-Grobentwurf-v03-comments_PR_MS.pdf)
2.2	28.06.2010	bruededl	Kapitel 3 Überarbeitet (100222-Grobentwurf-v03-comments_PR_MS.pdf)
2.3	11.07.2010	schneimi	Kapitel 3.3 überarbeitet (100222-Grobentwurf-v03-comments_PR_MS.pdf)

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. Die Teile, die erst in späteren Iterationen entworfen 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 Datenmangementoperationen innerhalb des Prozesses erweitert wird. Diese Aktivitäten werden im folgenden Verlauf des Dokuments als Datenmanagement-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 das SIMPL Rahmenwerk sollen außerdem Referenzvariablen bei der Modellierung von Prozessen unterstützt werden und diese durch eine entsprechende Modelltransformation [6] 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 (siehe [5] Kapitel 7.3) 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 solch eine Modularisierung der Befehle sind insbesondere komplexe und geschachtelte Datenmanagementoperationen 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 innerhalb

des Prozesses gehalten werden müssen, wird ein Reference Resolution System (RRS) nach [6] realisiert. Damit wird es möglich, Daten zu referenzieren, sie per Referenz sehr schnell weiterzugeben und nur bei Bedarf aufzulösen und in den Prozess zu laden.

Weiterhin kann es einen weiteren Typ von Referenzen innerhalb der DM-Aktivitäten geben. In diesem Kontext können 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, das Auditing zu aktivieren und zu deaktivieren. Zusätzlich soll die interne Datenspeicherung auf eine beliebige externe Datenbank 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 eine Datenquelle und der Angabe einer Auswahlstrategie durch den Prozess-Modellierer, eine passende Datenquelle erst 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, sodass 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 7.4.1).

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 registriert 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 Aktivieren und Deaktivieren des Auditings.

1.3.8 Erweiterbarkeit

Für das gesamte SIMPL Rahmenwerk und insbesondere die DM-Aktivitäten werden folgende Anforderungen an die Erweiterbarkeit gestellt, die über die Bereitstellung entsprechender Zugriffspunkte gewährleistet werden sollen:

SIMPL Rahmenwerk

- Anbindung weiterer Frameworks für den Datenzugriff wie z.B. JDBC, SDO, EJB
- Erweiterung der Funktionalität des Rahmenwerks durch Integration neuer Dienste, die ggf. bereits vorhandene Dienste benutzen
- Unterstützung weiterer Authentifizierungs- und Autorisierungsverfahren
- Unterstützung weiterer Registries für Datenquellen
- Erweiterung um zusätzliche Unterpunkte für die Admin-Konsole

DM-Aktivitäten

- Anbindung weiterer Datenquellentypen bzw. Anfragesprachen
- Erweiterung/Anpassung der DM-Aktivitätstypen, insbesondere Erweiterung um neue Datenmanagement-Patterns
- Erweiterung um neue Events für das Auditing

1.3.9 Verwendbarkeit

BPEL-Prozesse, die Referenzvariablen ([6]) nutzen, 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, die eingesetzten Protokolle und die übertragenen Daten.

2 Architektur

Abbildung 1 zeigt die Architektur des SIMPL Rahmenwerks. Es zeigt die bereits vorhanden Komponenten, Komponenten, die realisiert werden und externe Komponenten, die eingebunden bzw. unterstützt werden. Die eingezeichneten Pfeile beschreiben den Datenfluß zwischen den Komponenten, machen aber keine Aussage über die Abhängigkeit bzw. welche Komponente dabei auf welche zugreift. Das Zusammenspiel der Komponenten wird in den folgenden Abschnitten näher beschrieben, auf die Komponenten selbst und ihre Erweiterungen wird in Kapitel 3 eingegangen.

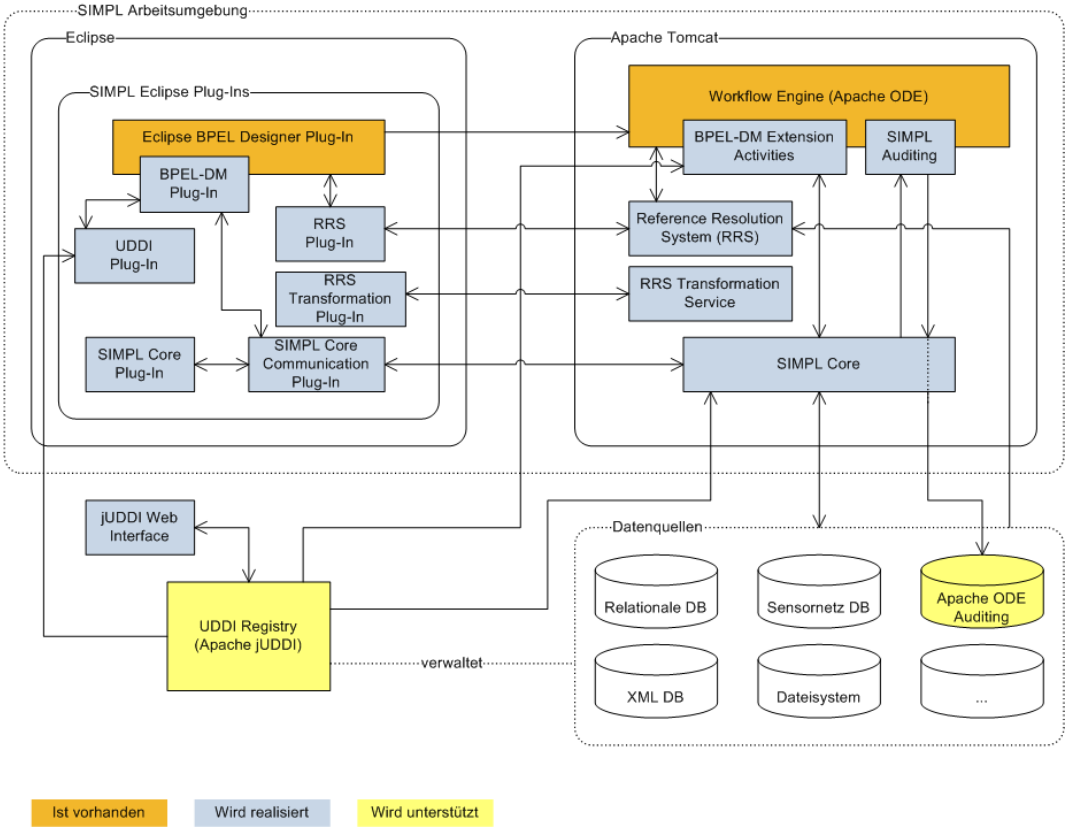


Abbildung 1: Architektur des SIMPL Rahmenwerks

2.1 Eclipse

Die Entwicklungsumgebung Eclipse bildet mit den SIMPL Eclipse Plug-Ins sowohl die GUI für den Prozess-Modellierer als auch für 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). Mit dem BPEL-DM Plug-In werden die bestehenden Aktivitäten des Eclipse BPEL Designer Plug-Ins um die DM-Aktivitäten über einen vorhandenen Extension Point erweitert und um weitere Modellierungshilfen wie z.B. Statement-Editoren und die Bereistellung von Datenquellen-Metadaten erweitert. Das SIMPL Core Plug-In stellt das SIMPL Eclipse-Menü und die GUI der Admin-Konsole für den Workflow-Administrator (siehe [5], Kapitel 5.2) zur Verfügung. Beide Plug-Ins nutzen das SIMPL Communication Plug-In, um die Verbindung zu den SIMPL Core Web Services (siehe Kapitel 3.3) herzustellen.

Mit dem RRS Plug-In wird die Modellierung und Verwaltung von Referenzvariablen im Eclipse BPEL Designer ermöglicht und über das RRS Transformation Plug-In der Kontakt zum RRS Trans-

formation Service hergestellt, über den ggf. vor dem Deployment eine Modelltransformation durchgeführt werden kann (siehe Anforderung 1.3.9). Eine UDDI Datenquellenregistry (siehe Anforderung 1.3.6) kann über das UDDI Plug-In integriert werden und stellt die verwendbaren Datenquellen für die Modellierung im Eclipse BPEL Designer zur Verfügung.

2.2 Apache Tomcat

Der Web Container Apache Tomcat ist die Laufzeitumgebung für Apache ODE, den SIMPL Core und das RRS, die aber grundsätzlich auch in unterschiedlichen Laufzeitumgebungen installiert sein können. Apache ODE ist für die Ausführung der Prozesse nach der Prozessmodellierung zuständig und wird über einen vorhandenen Extension Point für die zusätzliche Ausführung von DM-Aktivitäten um die BPEL-DM Extension Activities erweitert. Diese neuen Extension Activities können über den SIMPL Core mit den Datenquellen kommunizieren und z.B. Daten abrufen. Wie bereits in Kapitel 1.2 erwähnt, werden dazu bestimmte Dienste benötigt, die vom SIMPL Core bereitgestellt werden, der sich aus nach Aufgaben eingeteilten Web Services zusammensetzt (siehe Kapitel 3.3).

Das Reference Resolution System bietet Web Services für die Verwaltung der Referenzen in einer Embedded Datenbank über das RRS Plug-In, sowie für das Abrufen und Auflösen von Referenzen zur Laufzeit von der Workflow Engine (siehe Kapitel 3.4.1). Der eigenständige RRS Transformation Service realisiert den Web Service für die BPEL Transformation über das RRS Transformation Plug-In.

Das Auditing auf einer externen Datenbank (siehe Anforderung 1.3.3) wird über die SIMPL Auditing Erweiterung in der Workflow Engine realisiert, die alle relevanten anfallenden Daten von der Workflow Engine an den SIMPL Core weiterleitet, über den die Daten in einer Auditing Datenbank (Apache ODE Auditing) gespeichert werden. Das Ein- und Ausschalten des Auditings zur Laufzeit, wird vom SIMPL Core direkt über die SIMPL Auditing Erweiterung vorgenommen.

2.3 UDDI Registry

Eine UDDI Registry ermöglicht die zentrale Verwaltung von Datenquellen und wird über das jUDDI Web Interface ermöglicht. In der Registry werden auch die Eigenschaften der Datenquellen erfasst, die z.B. für das Late Binding (siehe Anforderung 1.3.4) benötigt werden oder über die der Prozess-Modellierer eine geeignete Datenquelle identifizieren kann. Das SIMPL Rahmenwerk wird mit Apache jUDDI als Registry ausgeliefert, unterstützt aber auch andere UDDI-Registries, welche die Version 3 der UDDI API Spezifikation [7] unterstützen. Die Registry wird zur Laufzeit vom SIMPL Core für das Auffinden einer Datenquelle beim Late Binding verwendet, sowie vom UDDI Plug-In um die Datenquellen in Eclipse bereitzustellen und von Apache ODE um die bei der Modellierung verwendeten logischen Deskriptoren wie z.B. Namen von Datenquellen zu einer Datenbankaddresse aufzulösen, mit der über den SIMPL Core dann darauf zugegriffen werden kann.

3 Komponenten

In Abbildung 2 werden die für das SIMPL Rahmenwerk relevanten Komponenten und ihre Abhängigkeiten gezeigt, die in den folgenden Abschnitten näher beschrieben werden. Die Richtung der Pfeile zu den Komponenten beschreibt die Richtung, in der die Zugriffe stattfinden, der Datenfluß dabei wird durch gestrichelten Pfeile verdeutlicht.

Eclipse greift auf die Registry Apache jUDDI zu, um die zur Verfügung stehenden Datenquellen abzurufen und für die Modellierung zur Verfügung zu stellen. Der SIMLCore greift ebenfalls auf Apache jUDDI zu, um beim Late Binding eine passende Datenquelle zu finden. Die bei der Modellierung verwendeten logischen Deskriptoren von Datenquellen werden von Apache ODE zur Laufzeit über Apache jUDDI aufgelöst. Eclipse bietet die Möglichkeit, den SIMPL Core über eine GUI (Admin-Konsole) zu administrieren, Einstellungen werden im SIMPL Core gespeichert und von dort wieder geladen. Zusätzlich können vom SIMPL Core Metadaten von den Datenquellen abgerufen werden, die in Eclipse zur Modellierung verwendet werden können.

Für Einstellungen zur Laufzeit, wie das An- und Ausschalten des Auditings, muss vom SIMPL Core direkt auf Apache ODE zugegriffen werden. Apache ODE verwendet den SIMPL Core für die Ausführung der DM-Aktivitäten und für die Speicherung der Auditing Daten, sowie das RRS, um Referenzen zur Laufzeit aufzulösen. Die Referenzen des RRS werden ebenfalls über eine GUI in Eclipse verwaltet und innerhalb des RRS gespeichert. Mit Hilfe des RRS Transformation Service kann in Eclipse ggf. eine Modelltransformation durchgeführt werden, bevor ein Prozess in Apache ODE deployt wird.

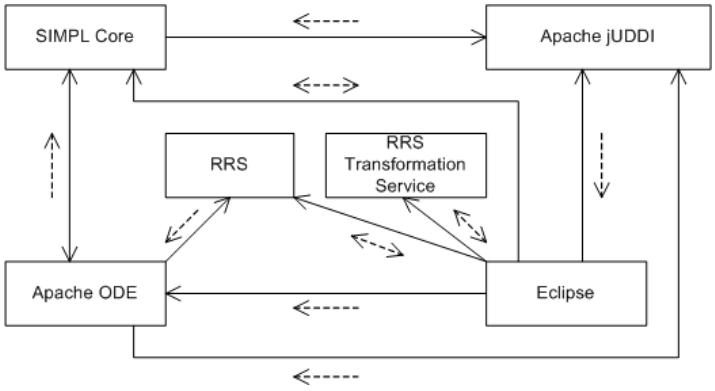


Abbildung 2: Komponenten des SIMPL Rahmenwerks

3.1 Eclipse

Im Rahmen des SIMPL Projektes sind die relevanten Komponenten der Enwicklungsumgebung Eclipse in Abbildung 3 dargestellt. Dies sind die Eclipse Basis IDE und das SIMPL Eclipse Plug-In. Das SIMPL Eclipse Plug-In setzt sich aus verschiedenen Plug-Ins zusammen, die über bestehende Extension Points realisiert werden und ggf. selbst Extension Points anbieten. In den folgenden Abschnitten werden die Plug-Ins näher beschrieben.

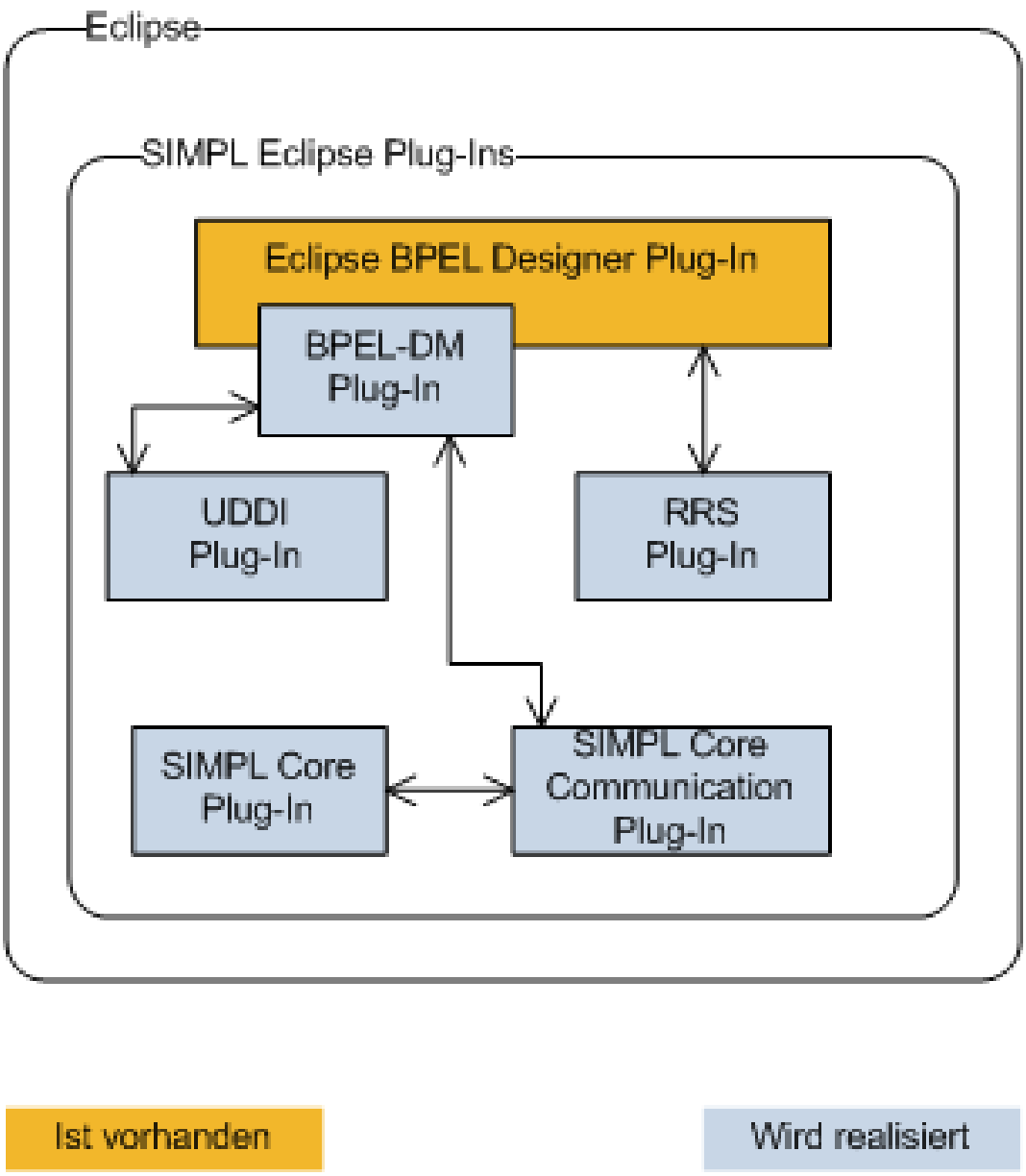


Abbildung 3: Eclipse mit SIMPL Erweiterungen

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 Auswahlmöglicheiten als Liste dar, die auf die verschiedenen Funktionsbereiche des Plug-Ins verweist. 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 können. 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 Plug-In benutzt werden. Es gibt auch noch weitere Extension Points im BPEL Designer Plug-In, diese sind jedoch nicht für SIMPL relevant.

UIObjectFactory Dieser Extension Point ermöglicht das Anbinden neuer UIObjectFactories, mit deren Hilfe Instanzen von BPEL Extension Activities erzeugt werden können. Beim Einfügen einer Aktivität der BPEL Designer Palette in einen Prozess wird ein neues Objekt durch die UIObjectFactory erzeugt.

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 neue Items und Kategorien der Palette hinzugefügt 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 Datenmanagement-Aktiviäten, die die Standard BPEL-Aktivitäten des BPEL Designer Plug-Ins erweitern. Die Erweiterung der Aktivitäten geschieht über folgende drei Extension Points:

- UIObjectFactory (- Erzeugt neue Objekte für die Modellierung).
- PaletteAdditions (- Ermöglicht grafische Auswahlmöglichkeit in der Palette).
- PropertySections (- Ermöglicht Erzeugung und Konfiguration von Werten der Aktivitäten).

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

3.1.4 SIMPL Core Plug-In

Das SIMPL Core Plug-In erweitert die Eclipse IDE um das SIMPL-Menü. Das SIMPL-Menü beinhaltet die Menüpunkte "Admin-Console", "Settings", "Help" und "About" und "Reload Plug-In Data". Mit Ausnahme des letzten Punktes verlinken alle Punkte auf die entsprechenden Fenster. Die "Admin-Console" ermöglicht die De- un Aktivierung des Auditings sowie dessen Konfiguration. "Settings" bietet die Möglichkeit, Einstellungen für die SIMPL Eclipse Plug-Ins vorzunehmen. Dazu gehört die Angabe der Adresse des Reference Resolution Systems, des SIMPL Cores, des Transformation Services und der UDDI Registry. Der Menüpunkt Help leitet den Benutzer auf die Eclipse Hilfe weiter. Dort stehen dem Benutzer über die Punkte BPEL-DM Plug-In und SIMPL Core Plug-In die entsprechenden Hilfe-Dokumente der SIMPL Eclipse Plug-Ins zur Verfügung. "About" bietet rechtliche Informationen sowie Angaben zum Entwicklerteam von SIMPL. Über "Reload Plug-In Data" können die Daten der Plug-Ins neu geladen werden. Damit erspart man sich einen Neustart des Rahmenwerkes, wenn bei der Initialisierung der SIMPL Core nicht zu erreichen war. 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 SIMPL Core Communication Plug-In

Das SIMPL Core Communication Plug-In ist für das Laden und Speichern von Rahmenwerkseinstellungen zuständig. Durch diese Realisierung als eigenständiges Plug-In ist das Rahmenwerk bei Modifikationen flexibler.

3.1.6 UDDI Plug-In

Das UDDI Plug-In ist in eine Eclipse View integriert. Die Datenquellen-Informationen werden aus der Datenquellen-Registry gelesen und in einer Tabelle in der View angezeigt. Die Informationen können nur angezeigt, und nicht verändert oder gelöscht werden. Dies kann nur in der Datenquellen-Registry direkt über das Web Interface gemacht werden.

3.1.7 RRS Plug-In

Das RRS Plug-In wird über den "View"-Extension-Point in Eclipse angebunden. Damit wird eine GUI-Schnittstelle für den RRS Management Service angeboten. Nähere Informationen dazu finden sich im Kapitel 3.4. So können über Eclipse mit dem RRS Plug-In alle Referenzen des angebundenen RRS verwaltet werden. Hier kann auch die Adresse des gewünschten RRS eingestellt werden, wenn mehrere RRS vorhanden sind und eines ausgewählt werden sollte. Die Modellierung der Referenzvariablen erfolgt in der RRS-View. Dort kann durch Betätigen des "New-Buttons oder durch New im Kontextmenü eine neue Referenzvariable erstellt und in den Pim Zuerst muss dazu in der Toolbar oder im Kontextmenü "New" angeklickt werden. In einem Pop-up Fenster können nun alle Parameter der Referenz eingestellt werden. Für die Transformation wird dann die Referenzvariable vom RRS-Transformation Plug-In an den an das RRS-Transformation-Service gesendet und dort aufgelöst um die "echte" Adresse zu erhalten. Nach der Auflösung können die Daten über das RRS geholt werden.

3.2 Apache ODE

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

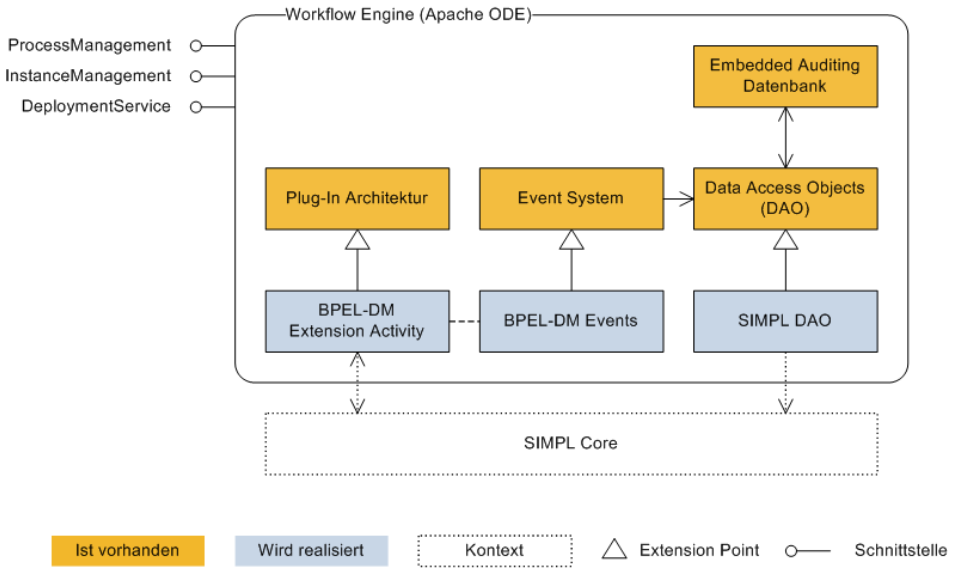


Abbildung 4: Apache ODE mit SIMPL Erweiterungen

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 Prozessmodell, und auf Prozessinstanzebene 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 Prozessinstanzen abgerufen werden.

Die Management API unterteilt sich in Instanz-Management und Prozess-Management. Beide bieten verschiedene Funktionen für Prozessinstanzen bzw. Prozessmodelle an. Es besteht auf der einen Seite die Möglichkeit, verschiedene Informationen über die Prozessemodelle und Prozessinstanzen zu erhalten, zum Beispiel eine Liste der verschiedene Prozessmodelle oder Informationen über ein einzelnes Prozessmodell. Auf der anderen Seite ist es 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 Deploy- und Undeploy-Funktionen.

3.2.2 Komponenten

Hier werden die bereits vorhandenen 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 BPEL um zusätzliche Aktivitäten 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 von ODE zu kopieren (ode\WEB-INF\lib). 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:

- Instance Life Cycle Events
  - Diese Events sind für alle Ereignisse zuständig, die während der Ausführung einer Prozessinstanz auftreten können. Dies sind beispielsweise das Starten der Ausführung einer Prozessinstanz oder das Terminieren einer Prozessinstanz
- Activity Life Cycle Events
  - Diese Events sind für alle Ereignisse zuständig, die während der Ausführung von Aktivitäten auftreten können. Dies sind beispielsweise das Starten der Ausführung einer Aktivität oder Rückmeldungen über das Fehlschlagen einer Aktivität.
- Scope Handling Events
  - Diese Events sind für alle Ereignisse zuständig, die während der Ausführung von Scopes auftreten können. Dies sind peispielsweise das Registrieren von Compensation Handlern für einen Scope und Rückmeldungen über Fehler in einem Scope.
- Data Handling Events
  - Diese Events sind für alle Ereignisse die beim Umgang mit Daten auftreten können. Dies betrifft beispielsweise das Lesen und Modifizieren von Variablen.
- Correlation Events
  - Diese Events sind für Ereignisse "Correlation sets" betreffend zuständig. "Correlation sets" sind dabei Partner die Nachrichten untereinander austauschen. Hier gibt es nur die Events "Correlation match" und "Correlation no match". "Correlation match" wird ausgelöst wenn das "Correlation set" für das die Nachricht bestimmt ist gefunden wurde und "Correlation no match", falls es nicht gefunden wurde.

Diese Events sind in einer Hierarchie organisiert. Diese ensteht durch die Vererbungsverhältnisse unter den Events. Dabei haben alle verschiedenen Events die Klasse "BPEL-Event" als Oberklasse. Anschließend ist die weitere Hierarchie durch die von den einzelnen Events betroffenen Konstrukten begründet. Das bedeutet, dass "Activity Life Cycle Events" eine Unterklasse von "Scope Handling Event" ist. "Scope Handling Events" wiederum sind eine Unterklasse von "Instanz Life Cycle Events", welche eine direkte Unterklasse von "BPELEvent" sind. Bei "BPELEvent" handelt es sich um die Vaterklasse, von der alle anderen Events innerhalb der Hierarchie abgeleitet werden.

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 die Auditing-Daten auf eine externe Datenbank umzuleiten. Es existiert zum Beispiel die Möglichkeit eine MySQL Datenbank zu nutzen. Dies ist jedoch nur über die ODE-Config-Dateien möglich und wird erst nach dem Neustart von ODE übernommen. Mit dem SIMPL DAO ist es möglich die Auditing-Daten, auch wärend der Laufzeit des Prozesses auf eine andere Datenbank umzuleiten. Näheres zum SIMPL DAO siehe (3.2.3) .

Embedded Auditing Datenbank Die Embedded Auditing Datenbank ist eine Apache Derby Datenbank und wird als Standard bei Apache-Ode mitgeliefert. Die Auditingdaten, die ODE erzeugt, werden dort standardmäßig gespeichert.

3.2.3 Erweiterungen

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

BPEL-DM Extension Activities Diese Extension Activities sind für die Ausführung der unterschiedlichen DM-Aktivitäten durch Apache ODE zuständig. Sie werden über den Extension Point "Extension Activity" realisiert. Extension Acitivities 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 ("AbstractAsyncExtensionOperation" und "AbstractSyncExtensionOperation"). Bei der Ausführung der DM-Aktivitäten wird der SIMPL Core genutzt, um die Datenmanagement-Operationen zur Verarbeitung an die jeweilige Datenquelle zu schicken.

SIMPL DAO Das SIMPL DAO nutzt die JPA-DAO Implementierung von Apache ODE und erweitert diese, um die Möglichkeit Auditing-Daten über SDOs an den SIMPL Core zu senden, über den diese in beliebigen Datenbanken gespeichert werden können. Die ursprüngliche JPA-DAO Funktionalität wird dadurch nicht beeinflusst. Der Vorteil dieser Methode ist, dass die Auditing-Daten unabhängig vom JPA-DAO, in einer eigenen und übersichtlicheren Struktur angeordnet werden können.

3.3 SIMPL Core

Der SIMPL Core, dargestellt in Abbildung 5, stellt die Funktionalität zur Verfügung, die während der Prozessmodellierung mit DM-Aktivitäten und der Ausführung dieser über Extension Activities benötigt wird. Er bietet nach außen verschiedene Web Services an, die jeweils bestimmte Aufgaben innerhalb des SIMPL Rahmenwerks übernehmen. Der SIMPL Core wird zudem als JAR-Datei im Classpath von Apache ODE hinterlegt, dadurch wird eine schnelle direkte Kommunikation zwischen dem SIMPL Core und Apache ODE ermöglicht. 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). Des Weiteren kann der SIMPL Core als Singleton betrieben werden, und es können damit z.B. Verbindungen zu Datenquellen für Folgezugriffe aufrecht gehalten werden.

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

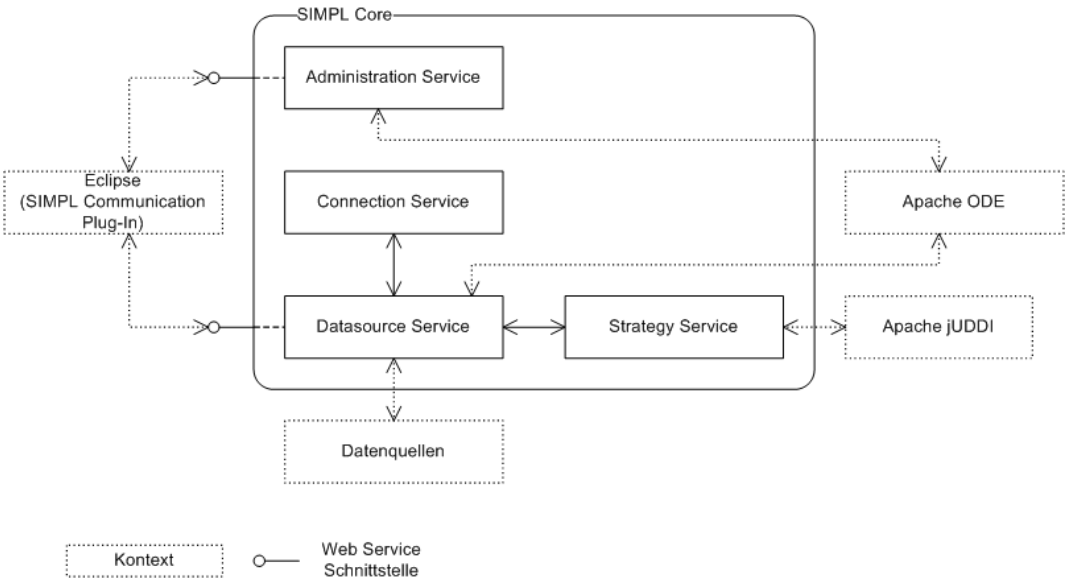


Abbildung 5: SIMPL Core

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 Einstellungsinformationen der Admin-Konsole werden über den Storage Service verwaltet. Für Einstellungen, mit denen zur Laufzeit Einfluss auf Apache ODE genommen wird, wie z.B. das Ein- und Ausschalten 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, durch die eine lose Kopplung zwischen Administration Service und der GUI geschaffen wird.

3.3.2 Datasource Service

Der Datasource Service ist für alle Aufgaben zuständig, die den Zugriff auf die Datenquellen und Informationen zu den Datenquellen (z.B. Metadaten) betreffen. Er wird auf Klassenebene von Apache ODE zur Ausführung der DM-Aktivitäten und die Speicherung der Auditing-Daten verwendet, sowie von Eclipse für den Abruf von Informationen zu den Datenquellen. Dazu 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 Datenformate bei Datenquellen (z.B. CSV-Daten bei Dateisystemen) und die Konvertierung der Daten zwischen den Datenformaten.

Die Authentifizierungdaten für den Verbindungsaufbau zu einer Datenquelle, werden in der Datenquellenregistry (siehe 1.3.6) hinterlegt und beim Zugriff an den Datasource Service übermittelt, wo sie vom Connection Service zwischengespeichert werden.

3.3.3 Connection Service

Der Connection Service ist für die Verwaltung von Datenquellenverbindungen zuständig und ermöglicht das Pooling von Verbindungen, die wiederaufgenommen werden können. Damit entfällt bei nachfolgenden Verbindungen, der meist lange Verbindungsaufbau und die Authentifzierung. Im Rahmen des

Studienprojekts und der damit unterstützten Typen von Datenquellen, wird der Connection Service ausschließlich für JDBC-Datenbankverbindungen realisiert.

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 funktionalen und nicht-funktionalen Anforderungen eine passende Datenquelle ausfindig zu machen. Im Rahmen des Studienprojekts wird nur die Auswahlstrategie "Erster Fund" realisiert. Die Datenquellen zur Auswahl für den Strategy Service, werden von der UDDI Registry (Apache jUDDI) bereitgestellt.

3.4 Reference Resolution System (RRS)

Mit Hilfe des RRS wird das Referenzieren von großen Datenmengen in BPEL ermöglicht (siehe Anforderung 1.3.2). Wie in Abbildung 6 zu sehen ist, besteht es aus zwei Web Services und internen Datenquellen-Adaptern (Datasource Adapter), die in den folgenden Abschnitten beschrieben werden.

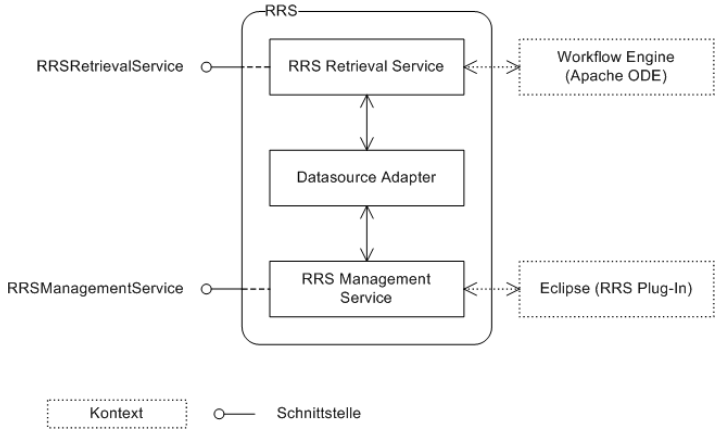


Abbildung 6: Reference Resolution System (RRS)

3.4.1 RRS Retrieval Service

Über den RRS Retrieval Service lassen sich Referenzen auflösen und die referenzierten Daten bei Bedarf von der Workflow Engine in einen Prozess holen.

3.4.2 Datasource Adapter

Das RRS bietet über Datasource Adapter eine erweiterbare Möglichkeit, verschiedenste Datenquellen anzubinden und darüber auch mit dem RRS Retrieval Service und RRS Management Service Referenzen und referenzierte Daten zu verwalten.

3.4.3 RRS Management Service

Mit dem RRS Management Service können Referenzen z.B. über das Eclipse RRS Plug-In verwaltet werden. Die Verwaltung der Referenzen geschieht dabei vorerst über eine eingebettete Datenbank, es ist allerdings später auch eine externe Verwaltung durch die Verwendung der entsprechenden Datenquellen-Adapter denkbar.

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 UDDI Plug-In kann über die Registry Datenquellen für den Prozeß-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 Kapitel 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 Listen bzw. HashMaps übertragen.

4.1.2 StorageService (intern und extern)

Die StorageService-Schnittstelle bietet Funktionen zur Verwaltung von Daten innerhalb des Rahmenwerks. Die Daten werden als Service Data Object (SDO) [4] übertragen.

4.1.3 DatasourceService (extern)

Die DatasourceService-Schnittstelle wird von Apache ODE für den Zugriff auf Datenquellen und die Speicherung 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 SDO 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 Kapitel 3.3.4). Die nichtfunktionalen Anforderungen an die Datenquellen werden dabei als WS-Policy-Objekte übertragen, andere funktionale Eigenschaften als proprietäre 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.

4.2 RRS

In den folgenden Abschnitten werden die Schnittstellen des RRS beschrieben (siehe Seite 19 6). Die nachfolgenden Beschreibungen beziehen sich auf [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 anschließend wird eine SOAP Nachricht zurückgeschickt in der die entsprechenden Daten stehen.

4.2.2 RRSManagementService

Diese Schnittstelle bietet Funktionen zum Speichern, Löschen und Ändern von Referenzen an und gibt im Falle eines Fehlers eine entsprechende Rückmeldung die darüber informiert was den Fehler verursacht hat.

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 v2.5.* Stupro-A SIMPL (2010).

[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	Datenmanagement
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