Studienprojekt SIMPL

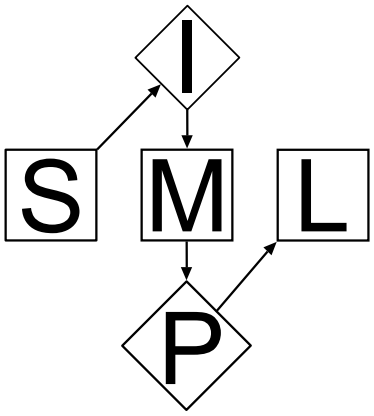
Spezifikation

Version 0.1

11. September 2009

Verfasser:

Wolfgang Huettig, Michael Hahn, Firas Zoabi, Michael Schneidt



Dok-Status: neu

QS-Status: nicht QS-geprüft

Prüf-Status: nicht geprüft

Review-Status: kein Review durchgeführt

End-Status: -

Inhaltsverzeichnis

1 Einleitung

In diesem Abschnitt 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 Richtschnur 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, 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 beruhen. Über das Rahmenwerk sollen beliebige Datenmanagement-Funktionen in einen BPEL-Prozess eingebunden werden können. Dafür werden bereits vorhandene Konzepte evaluiert und falls nötig erweitert oder angepasst. Es wird untersucht, inwiefern die Sprache BPEL ebenfalls erweitert werden muss. Für eine möglichst hohe Flexibilität soll ein dynamischer Ansatz gewählt werden, so dass erst während der 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-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-Engine ausgeführt werden können.

1.3 Definitionen

Die in der Spezifikation verwendeten Begriffe, Definitionen und Abkürzungen werden in einem separaten Begriffslexikon definiert und eindeutig erklärt. Durch das Begriffslexikon werden somit alle technischen und fachlichen Begriffe, Definitionen und Abkürzungen, die zu Missverständnissen innerhalb des Projektteams oder zwischen Projektteam und Kunde führen könnten, eindeutig definiert.

Auf alle in Abschnitt 6.2.3 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 6.2.3 beschriebenen Aktivitäten, realisiert werden.

1.4 Überblick

In diesem Dokument soll die zu entwickelnde Software spezifiziert werden. Dazu werden in Abschnitt 2 die spätere Systemumgebung, die Kernfunktionen, die Sprache und weitere Aspekte der Software beschrieben. So erhält der Leser einen Überblick über die Software, deren Verwendung und die Ziele

und Aufgaben, die für die Realisierung bestehen. Anschließend werden die vom Kunden genannten Anforderungen durch die Abschnitte 3 und 5 aufgezeigt. Dabei werden durch die nichtfunktionalen Anforderungen qualitative (Robustheit, Portabilität, usw.) und quantitative (Mengengerüst) Anforderungen an die Software spezifiziert. Die Anwendungfälle beschreiben im Anschluss die funktionalen Anforderungen, also konkret die Funktionen, die z.T. schon in der Übersicht der Kernfunktionalität weiter oben aufgeführt sind, die die Software enthalten soll. Im Anschluss folgen dann in Abschnitt 6 die Beschreibungen und Definitionen einiger Konzepte bzw. Ansätze, die zur Umsetzung der gewünschten Funktionalität benötigt werden. Am Ende des Dokuments werden dann noch die verwendeten Materialien, Werkzeuge und Technologien, die für die Erstellung der Software benötigt werden, in Abschnitt 7 vorgestellt.

2 Allgemeine Beschreibung

Dieser Abschnitt liefert allgemeine Informationen über die zu entwickelnde Software. Dazu gehöhren beispielsweise die Beschreibung der späteren Systemumgebung, die wichtigsten Funktionen, die verwendete Sprache und Informationen über den Benutzerkreis der Software.

2.1 Einbettung

Das SIMPL Rahmenwerk soll in die, in Abbildung 1 dargestellte, Systemumgebung eingebettet werden. Die Systemumgebung besteht dabei aus den unterschiedlichen Datenquellen, Eclipse mit dem BPEL-Designer Plug-In und einem Web-Server, wie z.B. dem Apache Tomcat, in dem das Rahmenwerk und eine Workflow-Engine (z.B. Apache ODE) ausgeführt werden. Dabei läuft die benötigte Software auf dem lokalen Rechner des Benutzers, die Datenquellen können auf verschiedene Server verteilt sein.

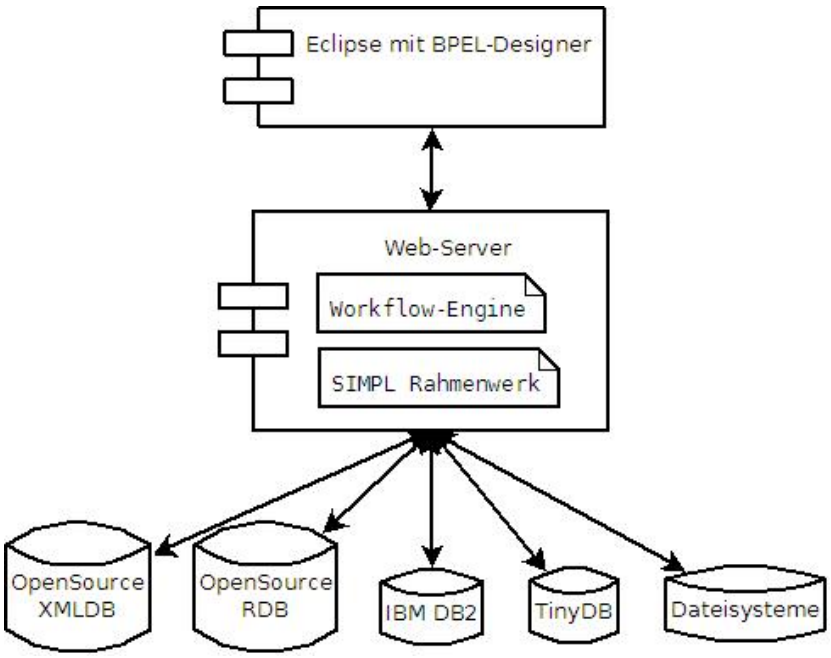


Abbildung 1: Übersicht über die Systemumgebung von SIMPL

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 als Laufzeitumgebung integriert sein. Ebenso soll die Verarbeitung von großen, heterogenen Datenmengen im Rahmen eines wissenschaftlichen Workflows möglich sein.
Um die Fehlertoleranz des Rahmenwerks zu realisieren, sollen Fehler der neu definierten Aktivitäten erkannt und durch entsprechende FaultHandler abgefangen werden können. Abgefangene Fehler werden dann durch entsprechende CompensationHandler kompensiert oder die vorher ausgeführten Aktionen rückgängig gemacht.

BPEL

Über den Eclipse BPEL Designer sollen innerhalb von BPEL-Prozessen Referenzen unterstützt werden, d.h. es soll möglich sein Referenzen (z.B. auf eine Tabelle) innerhalb eines SQL-Befehls über eine BPEL Variable anzugeben oder auch die Ergebnisse eines Queries per Referenz im Prozess bereit zu halten. Dazu ist es erforderlich, dass ein Reference Resolution System mit verschiedenen Adaptern implementiert wird. Dieses System dient dazu Referenzen/Pointer innerhalb eines BPEL Prozesses aufzulösen und so z.B. auch Daten in den Prozess-Cache zu laden. Bei Daten, die nicht für die Prozesslogik relevant sind, soll dabei die Referenz/der Pointer vom Service, zur Entlastung der Workflow-Engine, aufgelöst werden. Mithilfe eines Transformers übersetzt der BPEL Designer den BPEL-Code mit Referenzen in Standard BPEL-Code, der dann auf der Workflow-Engine ausgeführt werden kann.

BPEL Aktivitäten

Alle BPEL-Aktivitäten, die für den Umgang mit Datenquellen benötigt werden, müssen erstellt werden und sollen erweiterbar sein. Als Anhaltspunkt welche Aktivitäten benötigt werden, gelten die folgenden Datenmanagement Patterns (siehe Abschnitt 6.2 bzw. Quelle [2]: 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 extern gespeichert oder direkt im Prozessspeicher gehalten werden. Das *Set UID Pattern* beschreibt die Möglichkeit mengenorientiertes Einfügen (insert), Aktualisieren (update) und Löschen (delete) auf externen Daten durchführen zu können. 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. Da die Verarbeitung von komplexen Daten meist durch Stored Procedures erfolgt, ist es bei der Verarbeitung von externen Daten unbedingt erforderlich, Stored Procedures auch aus einem Prozess heraus aufrufen zu können (*Stored Procedure Pattern*). Manchmal ist es nötig Daten innerhalb des Prozessspeichers verarbeiten zu können. Das *Set Retrieval Pattern* liefert dafür eine mengenorientierte Datenstruktur, in die man die angefragten externen Daten innerhalb des Prozessspeichers abgelegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im Prozessspeicher, der keine Verbindung zur originalen Datenquelle besitzt. Das *Set Access Pattern* beschreibt die Notwendigkeit auf den erzeugten Datencache sequentiell und direkt (random) zugreifen zu können. Das *Tuple IUD Pattern* beinhaltet einfügen (insert), aktualisieren (update) und löschen (delete) von Daten im Datencache. Das *Synchronization Pattern* realisiert die Synchronisation eines lokalen Datencaches mit der originalen Datenquelle.

Anbindung von Datenquellen

Die Anbindung von Datenquellen soll generisch sein und es sollen so viele Anbindungen wie möglich realisiert werden. Als Minimum gilt dabei die IBM DB2, mindestens zwei OpenSourceDB's (eine RDB und eine XMLDB), mindestens eine SensorDB (z.B. TinyDB) und mindestens ein Dateisystem (z.B. ext3). Es sollen auch Zugriffe auf mehrere Datenquellen innerhalb eines Prozesses möglich sein, dazu

besitzt jede Aktivität eine Variable, in der der logische Namen der Datenquelle auf die zugegriffen werden soll, hinterlegt ist. Alle Anfragesprachen für den Datenbankzugriff müssen unterstützt werden, mindestens SQL und XQuery. Zusätzlich sollen auch alle SQL-Dialekte unterstützt werden. Transaktionen sollen innerhalb von DB und BPEL-Prozessen unterstützt werden. Die Konzepte sind nach Absprache mit dem Kunden frei wählbar. Es soll auch möglich sein, dass Daten aus Datenquellen aus einem Prozess auch in lokale Dateien exportiert werden können und umgekehrt soll es auch möglich sein lokale Daten zu importieren. Dabei gilt XML als Standardformat.

Die Datenbanken, die verwendet werden, existieren bereits und müssen nicht durch das Rahmenwerk erstellt werden. Weiterhin sollen aber Schema-Definitionen möglich sein und das Erstellen, Ändern und Löschen von Tabellen innerhalb der Schemas.

Das Late-binding von Datenquellen soll unterstützt werden. Dazu müssen Kriterien zur Beschreibung von Datenquellen erstellt werden. Diese Kriterien werden dann als Anforderungen durch den Benutzer modelliert und als Annotationen übertragen. Die Spezifizierung der Anforderungen, die sinnvoll an eine Datenquelle gestellt werden können sollten, wird über die Definition eines Anforderungskataloges, der nach Fertigstellung mit dem Kunden abgesprochen werden muss, realisiert. Die Annotationen dienen dazu, dass Datenquellen durch definierte Strategien automatisch ausgewählt werden können. Diese Strategien, die zur Auswahl einer Datenquelle anhand der vom Benutzer angegebenen Anforderungen verwendet werden, müssen von uns definiert und anschließend mit dem Kunden abgesprochen werden. Die Auswertung der Annotationen und die Verarbeitung der darin enthaltenen Informationen soll, ebenfalls selbst definiert und anschließend mit dem Kunden abgesprochen werden. Die Adressierung der Datenbanken soll statisch über konkrete Adressen oder auch dynamisch über logische Namen mithilfe der JNDI API möglich sein.

Admin-Konsole

Die Admin-Konsole ist ein Werkzeug mit dem verschiedene Einstellungen des Rahmenwerks auch während der Laufzeit noch geändert werden können. Dazu gehört beispielsweise das An- und Abschalten des Auditings und Einstellen der Granularität. Eine solche Komponente soll für die Administration des Rahmenwerks erstellt werden.

Autorisierung und Authentifizierung

Die Autorisierung und Authentifizierung soll momentan nur für Datenquellen bereitgestellt werden, allerdings soll eine spätere Erweiterung einfach realisiert werden können. Für die Authentifizierung und Autorisierung sind verschiedene Verfahren gewünscht, sodass in jeder Situation das Bestmögliche verwendet wird. Dazu soll es möglich sein, verschiedene konkrete Verfahren flexibel anbinden zu können. Die Autorisierung und die Authentifizierung soll dabei über einen Single-Sign-On in der Instanz beim Zugriff auf Datenquellen durchgeführt werden. Autorisierungs- und Authentifizierungsparameter für einen Prozess sollen auf Datenquellenebene spezifiziert werden, d.h. bei Datenbanken über Schema und bei Dateisystemen z.B. über extra Dateien. Die Autorisierung und Authentifizierung ist bei allen Datenquellen notwendig und soll auf verschiedene Arten möglich sein.Angaben über Autorisierung und Authentifizierung sollen extra im Eclipse BPEL Designer abgefragt und als Nachricht an die Datenquelle geschickt werden.

Auditing und Monitoring

Es soll ein Auditing der Prozessausführung geben, dafür soll die Ausführungshistorie in einer angebundenen Datenbank und nicht im lokalen Speicher der Engine gespeichert werden können. Die Datenbank für das Auditing soll über das Rahmenwerk frei wählbar sein. Die Auditing-Daten werden dabei auf einer Datenbank gespeichert und nicht über mehrere verteilt. Das vorhandene Auditing von Apache ODE muss dafür um das Auditing unserer Funktionalitäten und die Möglichkeit, eine variable DB als Auditing-DB anzugeben, erweitert werden. Die Speicherungsdauer der Auditing-Daten soll variabel als Parameter übergeben werden können. Das Auditing soll standardmäßig aktiv sein. Auf Benutzerwunsch

soll die Granularität des Auditings veränderbar sein oder das Auditing auch komplett abschaltbar. Dies soll auch während der Laufzeit über die Admin-Konsole steuerbar sein.

Für unsere Zwecke muss ein komplettes Monitoring erstellt werden, d.h. ein Monitoring für alle Datenquellenaktionen und die Anpassung und Erweiterung des bestehenden Monitorings von Apache ODE. Dazu soll das Event-Modell von Apache ODE für das Monitoring, d.h. welche Informationen angezeigt werden können, für unsere Erweiterungen modifiziert und mit den Kunden abgesprochen werden.

2.3 Sprache

Generell gilt, dass alle Dokumente auf Deutsch und jeder Quellcode einschließlich Kommentaren auf Englisch verfasst und ausgeliefert werden. Eine Ausnahme bilden das Handbuch und die verschiedenen Dokumentationen der durchzuführenden 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 Installationpakets 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 erleichern. Das Installationspaket wird dabei als RAR-Archiv zusammen mit allen wichtigen Dokumenten auf einer CD/DVD ausgeliefert. Zu den Dokumenten zählen die Spezifikation, das Handbuch mit Installationsanleitung auf Deutsch und Englisch, die vollständige Dokumentation aller Erweiterungen auf Deutsch und Englisch und der gesammte Quellcode des Rahmenwerks. 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.

2.5 Benutzerprofile

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

2.6 Einschränkungen

2.7 Annahmen und Abhängigkeiten

3 Nichtfunktionale Anforderungen

In diesem Abschnitt 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

3.2 Benutzbarkeit

Die Benutzbarkeit soll sich vor allem an Nutzer mit wenig Kenntnissen im Umgang mit Workflows und BPEL richten und dafür die größtmögliche Transparenz liefern, d.h. dass die interne Prozesslogik der

Software bestmöglich vom Benutzer abgeschirmt wird und er eine möglichst einfache und schnell verständliche Schnittstelle zur Software erhält, um die Verwendung von SIMPL für alle Benutzergruppen zu ermöglichen.

3.3 Robustheit

Unter Robustheit versteht man hier, zwar kommt die ungünstigen Bedingungen, aber SIMPL kann noch stabil zuverlässig angewanden ,richtigen Aktivitäten durchgeführt und korrekte Ergebniss liefert werden,oder sicher beendet werden.

3.4 Sicherheit

3.5 Portabilität

Portabilität bedeutet hier, wenn man bei verschiedenen Softwareumgebungen bzw. verschiedenen Operationsystems (z.B.nicht nur bei windows sonder auch bei linux)die SIMPL durchlaufen wollte, zwischen der verschiedenen Varianten braucht man nur weniger Anpassungen einzustellen.

3.6 Erweiterbarkeit

Die Erweiterbarkeit des Systems spielt 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, wird ein modularer Aufbau zugrunde gelegt und entsprechende Schnittstellen geschaffen.

3.7 Wartbarkeit

3.8 Skalierbarkeit

Die Skalierbarkeit des Systems muss eine sehr flexible Infrastruktur erlauben, da die Computersysteme, auf denen SIMPL später ausgeführt wird, in ihrer Leistung sehr weit auseinander gehen können, d.h. vom normalen Desktop-Computer bis zum Supercomputer kann und soll alles möglich sein.

3.9 Ausfallsicherheit

4 Akteure

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

4.1 Prozess-Modellierer

Ein Prozess-Modellierer erstellt mit dem Eclipse BPEL Designer BPEL Prozesse. Dazu kann er BPEL-Aktivitäten erstellen, bearbeiten und auch löschen. Weiterhin kann er für die dynamische Datenquellenauswahl zur Laufzeit, die von ihm benötigten Datenquellen-Eigenschaften angeben und eine geeignete Auswahl-Strategie auswählen. Hat der Prozess-Modellierer den BPEL-Prozess fertig modelliert, kann er diesen im Anschluss auf einer Workflow-Engine ausführen lassen.

4.2 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.

Er kann beispielsweise über die Admin-Konsole während der Prozesslaufzeit das Auditing und Monitoring an- und abschalten. Ebenso legt er die Datenbank für das Speichern der Auditing-Daten fest und administriert das Reference Resolution System, d.h. er kann Referenzen anlegen, löschen und bearbeiten.

4.3 Datenquellen-Administrator

Ein Datenquellen-Administrator verwaltet Datenquellen und stellt diese über die Datenquellen-Registry anderen Nutzern zu Verfügung. Dazu kann er Datenquellen in der Datenquellen-Registry registrieren, die Eigenschaften bereits registrierter Datenquellen bearbeiten und registrierte Datenquellen auch wieder aus der Datenquellen-Registry löschen.

4.4 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. Zu ihren Aufgaben gehöhrt das Ausführen, Kompensieren und rückgängig Machen von Daten-Management-Aktivitäten.

4.5 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 kann Datenquellen aus der Datenquellen-Registry abrufen und die erhaltene Liste anhand der vom Prozess-Modellierer hinterlegten Eigenschaften filtern. Weiterhin sorgt er für die automatische Datenquellenauswahl per Strategie, das Laden der Einstellungen der Admin-Konsole und das Kopieren aller Prozess-Artefakte auf die Workflow-Engine.

4.6 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 Ausflösen von Referenzen.

5 Anwendungsfälle (Use-Cases)

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

5.1 Diagramm aller Anwendungsfälle

Abbildung 2 zeigt das Diagramm aller Anwendungsfälle der gesamten Software. Dadurch soll die Funktionalität und die Akteure des späteren Gesamtsystems sichtbar werden.

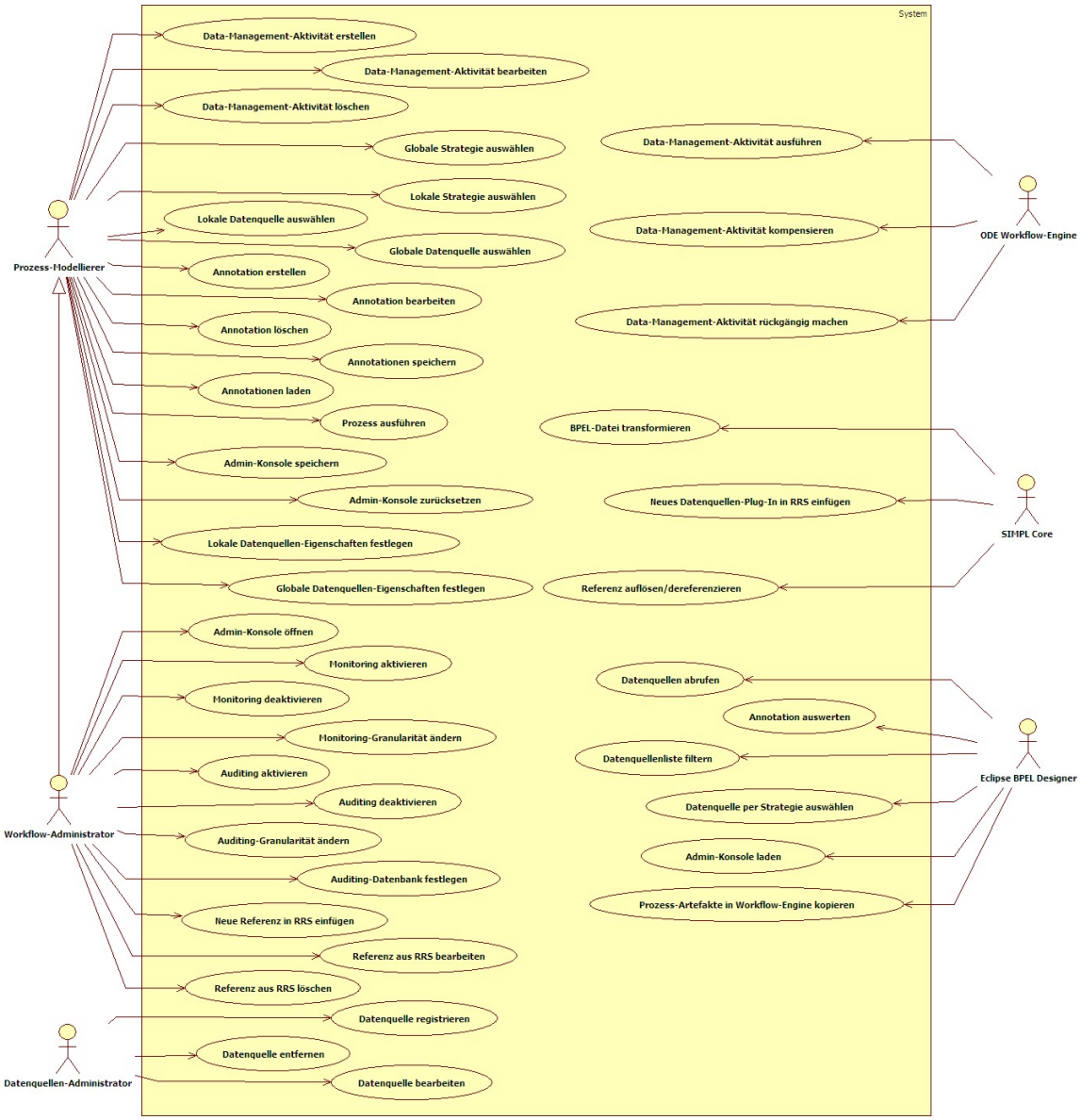


Abbildung 2: Anwendungsfall-Diagramm des gesamten Softwaresystems

5.2 Anwendungsfälle der Prozess-Modellierer

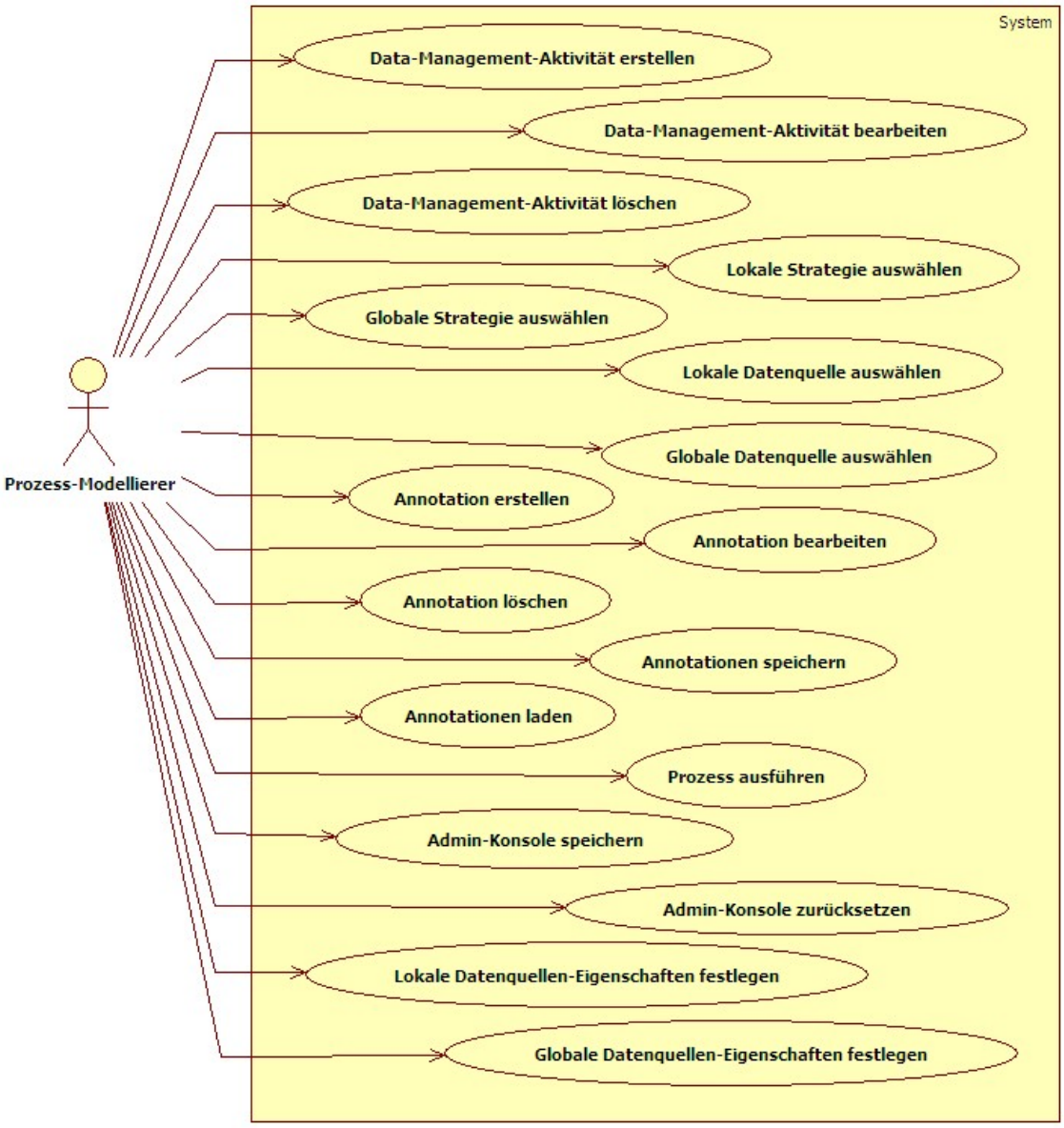


Abbildung 3: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.2.1 Data-Management-Aktivität erstellen

Ziel	Erstellung einer neue Daten-Management-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess muss im Eclipse BPEL Designer geöffnet und die BPEL Designer-Palette muss angezeigt werden.
Nachbedingung	Die erstellte Aktivität wurde an der selektierten Position in den Prozess eingefügt und der eingegebene Namen wird angezeigt.
Nachbedingung im Sonderfall	Die erstellte Aktivität wurde an der selektierten Position in den Prozess eingefügt und der vorgeschlagene Namen wird angezeigt.
Normalablauf	1. Auswahl einer Aktivität, durch Markierung mit der Maus, aus der Palette 2. Selektion der Stelle des Prozesses an der die ausgewählte Aktivität eingefügt werden soll 3. Eingabe eines Aktivitätsnamens
Sonderfälle	3a. Der angezeigte Namensvorschlag wird vom Benutzer mit [Enter] bestätigt

5.2.2 Data-Management-Aktivität bearbeiten

Ziel	Bearbeitung der Eigenschaften einer vorhandenen Daten-Management-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess muss im Eclipse BPEL Designer geöffnet, die zu bearbeitende Aktivität muss ausgewählt sein und der “Properties-View” von Eclipse muss angezeigt werden.
Nachbedingung	Alle durchgeführten Änderungen der Eigenschaften wurden korrekt übernommen und werden im “Properties-View” angezeigt.
Nachbedingung im Sonderfall	
Normalablauf	1. Auswahl einer Aktivität, durch Markierung mit der Maus, aus dem Prozess 2. Eigenschaften der Aktivität werden in der “Properties-View” angezeigt 3. Änderungen der Eigenschaften
Sonderfälle	

5.2.3 Data-Management-Aktivität löschen

Ziel	Löschen der ausgewählten Daten-Management-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess muss im Eclipse BPEL Designer geöffnet und die zu löschende Aktivität muss ausgewählt sein.
Nachbedingung	Die ausgewählte Aktivität wurde vollständig und korrekt aus dem Prozess gelöscht.
Nachbedingung im Sonderfall	
Normalablauf	1. Auswahl einer Aktivität, durch Markierung mit der Maus, aus dem Prozess 2. Löschen der Aktivität
Sonderfälle	

5.2.4 Lokale Strategie auswählen

Ziel	Auswahl einer lokalen Strategie zur Datenquellenauswahl, d.h. Auswahl einer Strategie, die nur für die Datenquellenauswahl der ausgewählten Aktivität gilt.
Vorbedingung	Das Eigenschaftsfenster einer Data-Management-Aktivität wird angezeigt.
Nachbedingung	Die ausgewählte Stategie wird intern gespeichert und für das Late-binding der Datenquelle der ausgewählten Aktivität verwendet.
Nachbedingung im Sonderfall	Die in den globalen Einstellungen ausgewählte Strategie wird für das Late-binding von Datenquellen für diese Aktivität verwendet.
Normalablauf	1. Auswahl einer Strategie aus der Liste
Sonderfälle	1a. Auswahl der globalen Strategie (Standardwert)

5.2.5 Globale Strategie auswählen

Ziel	Auswahl einer globalen Strategie zur Datenquellenauswahl, d.h. Auswahl einer Standard-Strategie, die für alle Aktivitäten als Standardwert festgelegt ist.
Vorbedingung	Der Dialog zur Strategieauswahl in der Admin-Konsole muss angezeigt werden.
Nachbedingung	Die ausgewählte Stategie wird gespeichert und für das Late-binding der Datenquellen verwendet.
Nachbedingung im Sonderfall	
Normalablauf	1. Auswahl einer Strategie aus der Liste 2. Speichern der Eingaben
Sonderfälle	

5.2.6 Lokale Datenquellen-Eigenschaften festlegen

Ziel	Lokales Festlegen der benötigten Datenquellen-Eigenschaften in der ausgewählten Aktivität.
Vorbedingung	Das Eigenschaftsfenster einer Data-Management-Aktivität wird angezeigt.
Nachbedingung	Die angegebenen Eigenschaftswerte werden intern gespeichert und für das Late-binding der Datenquelle der ausgewählten Aktivität verwendet.
Nachbedingung im Sonderfall	Die in den globalen Einstellungen hinterlegten Eigenschaftswerte werden für das Late-binding der Datenquelle der ausgewählten Aktivität verwendet.
Normalablauf	1. Eingabe der gewünschten Werte in den entsprechenden Eigenschaftsfeldern
Sonderfälle	1a. Auswahl der Checkbox zur Nutzung der global hinterlegten Eigenschaften in den globalen Einstellungen

5.2.7 Globale Datenquellen-Eigenschaften festlegen

Ziel	Globales Festlegen der benötigten Datenquellen-Eigenschaften in der Admin-Konsole.
Vorbedingung	Der Dialog zur Angabe der benötigten Datenquellen-Eigenschaften in der Admin-Konsole muss angezeigt werden.
Nachbedingung	Die angegebenen Eigenschaftswerte werden gespeichert und für das Late-binding der Datenquellen verwendet.
Nachbedingung im Sonderfall	
Normalablauf	1. Eingabe der gewünschten Werte in den entsprechenden Eigenschaftsfeldern 2. Speichern der Eingaben
Sonderfälle	

5.2.8 Lokale Datenquelle auswählen

Ziel	Auswählen einer lokalen Datenquelle für die ausgewählte Aktivität zur Modellierungszeit.
Vorbedingung	Bestehende Verbindung zur Datenquellen-Registry.
Nachbedingung	Die ausgewählte Datenquelle wurde intern als zugrundeliegende Datenquelle der Aktivität hinterlegt und ein erfolgreicher Verbindungstest mit den in den lokalen Eigenschaften hinterlegten Login-Daten ausgeführt.
Nachbedingung im Sonderfall	Es wird eine entsprechende Fehlermeldung angezeigt, die den Benutzer darüber informiert, warum der Verbindungstest nicht erfolgreich war. Ob z.B. die Login-Daten falsch waren oder die Datenquelle nicht erreichbar ist.
Normalablauf	1. Auswahl einer Datenquelle aus der Liste
Sonderfälle	1a. Der Verbindungstest schlägt fehl

5.2.9 Globale Datenquelle auswählen

Ziel	Auswählen einer globalen Datenquelle, die als Standard-Datenquelle für alle Aktivitäten genutzt werden kann, zur Modellierungszeit.
Vorbedingung	Der Dialog zur Auswahl einer Datenquelle in der Admin-Konsole muss angezeigt werden und es muss eine Verbindung zur Datenquellen-Registry bestehen.
Nachbedingung	Die ausgewählte Datenquelle wurde als globale Datenquelle hinterlegt, die Eingaben der Admin-Konsole erfolgreich gespeichert und ein erfolgreicher Verbindungstest ausgeführt.
Nachbedingung im Sonderfall	Es wird eine entsprechende Fehlermeldung angezeigt, die den Benutzer darüber informiert, warum der Verbindungstest nicht erfolgreich war. Ob z.B. die Login-Daten falsch waren oder die Datenquelle nicht erreichbar ist.
Normalablauf	1. Auswahl der gewünschten Datenquelle aus der Liste 2. Speichern der Eingaben
Sonderfälle	1a. Der Verbindungstest schlägt fehl

5.2.10 Annotation erstellen

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.2.11 Annotation bearbeiten

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.2.12 Annotation löschen

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.2.13 Annotationen speichern

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.2.14 Annotationen laden

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.2.15 Prozess ausführen

Ziel	Ausführen des Prozesses auf der Apache ODE Workflow-Engine.
Vorbedingung	Ein Prozess muss im Eclipse BPEL Designer geöffnet und die Apache ODE Workflow-Engine korrekt in Eclipse eingebunden sein.
Nachbedingung	Die Prozess-Dateien wurden auf die Apache ODE Workflow-Engine kopiert, der Prozess wurde erfolgreich deployed und wird ausgeführt.
Nachbedingung im Sonderfall	
Normalablauf	1. Erstellung eines ODE Deployment-Deskriptors 2. Kopieren der Prozess-Artefakte in Workflow-Engine 3.
Sonderfälle	

5.2.16 Admin-Konsole speichern

Ziel	Speicherung des Inhalts der Admin-Konsole.
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	Alle geänderten Werte der Admin-Konsole bleiben durch den Speichervorgang unverändert und es kann ein erneuter Versuch durchgeführt werden. Die Werte bleiben dabei solange erhalten, bis Eclipse beendet wird. Nach der Beendigung von Eclipse werden die Werte dann verworfen und beim nächsten Start die zuletzt gespeicherten Einstellungen geladen.
Normalablauf	1. Klick auf den Button [Speichern]
Sonderfälle	1a. Beim Speichern der Werte tritt ein Fehler auf

5.2.17 Admin-Konsole zurücksetzen

Ziel	Zurücksetzen des Inhalts der Admin-Konsole.
Vorbedingung	Die Admin-Konsole wird angezeigt.
Nachbedingung	Alle geänderten Werte der Admin-Konsole wurden auf die gespeicherten Werte zurückgesetzt.
Nachbedingung im Sonderfall	Alle geänderten Werte der Admin-Konsole bleiben durch den Zurücksetz-Vorgang unverändert und es kann ein erneuter Versuch durchgeführt werden. Die Werte bleiben dabei solange erhalten, bis Eclipse beendet wird.
Normalablauf	1. Klick auf den Button [Zurücksetzen]
Sonderfälle	1a. Beim Laden der Werte tritt ein Fehler auf

5.3 Anwendungsfälle der Workflow-Administratoren

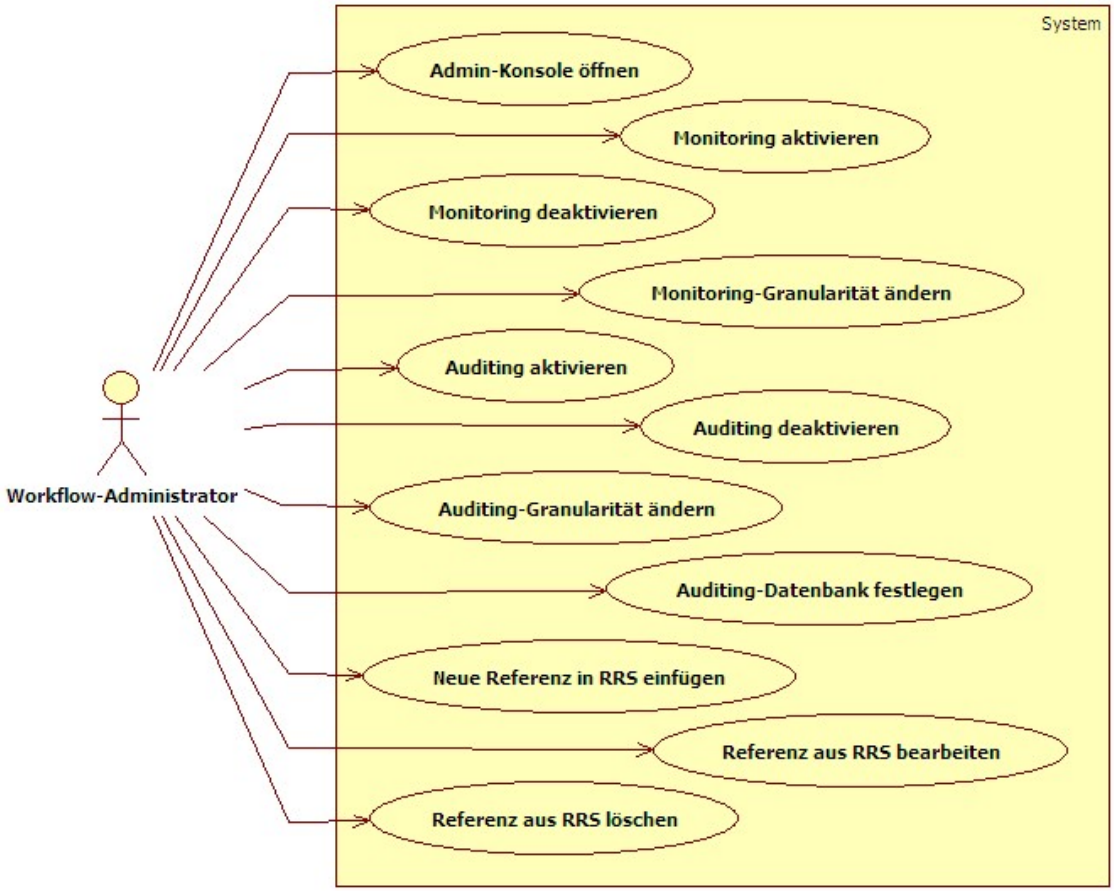


Abbildung 4: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.3.1 Admin-Konsole öffnen

Ziel	Öffnen der Admin-Konsole.
Vorbedingung	Eclipse muss geöffnet sein.
Nachbedingung	Die Admin-Konsole wird angezeigt und kann verwendet werden.
Nachbedingung im Sonderfall	
Normalablauf	1. Klick auf das SIMPL Menü 2. Klick auf den Menüeintrag [Admin-Konsole]
Sonderfälle	

5.3.2 Monitoring aktivieren

Ziel	Monitoring von SIMPL aktivieren.
Vorbedingung	Admin-Konsole muss angezeigt werden und das Monitoring nicht aktiv sein.
Nachbedingung	Das Monitoring ist aktiviert und kann über das SIMPL Menü angezeigt werden.
Nachbedingung im Sonderfall	
Normalablauf	1. Monitoring aktivieren
Sonderfälle	

5.3.3 Monitoring deaktivieren

Ziel	Monitoring von SIMPL deaktivieren.
Vorbedingung	Admin-Konsole muss angezeigt werden und das Monitoring ist aktiv.
Nachbedingung	Das Monitoring ist deaktiviert.
Nachbedingung im Sonderfall	
Normalablauf	1. Monitoring deaktivieren
Sonderfälle	

5.3.4 Monitoring-Granularität ändern (optional)

Zu Beginn wird nur ein statisches Monitoring unterstützt, d.h. es wird nicht möglich sein die Granularität zu ändern. Sofern zeitlich machbar wird diese Funktionalität noch nachgeliefert.

5.3.5 Auditing aktivieren

Ziel	Auditing von SIMPL aktivieren.
Vorbedingung	Admin-Konsole muss angezeigt werden und das Auditing ist nicht aktiv.
Nachbedingung	Das Auditing ist aktiviert.
Nachbedingung im Sonderfall	
Normalablauf	1. Auditing aktivieren
Sonderfälle	

5.3.6 Auditing deaktivieren

Ziel	Auditing von SIMPL deaktivieren.
Vorbedingung	Admin-Konsole muss angezeigt werden und das Auditing ist aktiv.
Nachbedingung	Das Auditing ist deaktiviert.
Nachbedingung im Sonderfall	
Normalablauf	1. Auditing deaktivieren
Sonderfälle	

5.3.7 Auditing-Granularität ändern (optional)

Zu Beginn wird nur ein statisches Auditing unterstützt, d.h. es wird nicht möglich sein die Granularität zu ändern. Sofern zeitlich machbar wird diese Funktionalität noch nachgeliefert.

5.3.8 Auditing-Datenbank festlegen

Ziel	Festlegung einer Datenbank für das Auditing.
Vorbedingung	Die Admin-Console ist geöffnet.
Nachbedingung	Die Datenbank für das Auditing wurde festgelegt und kann verwendet werden.
Nachbedingung im Sonderfall	Der Administrator wird durch eine entsprechende Fehlermeldung darauf hingewiesen, dass die angegebene Datenbank nicht erreichbar ist.
Normalablauf	1. Angabe der Datenbank-Adresse (optional Auswahl über Datenbank-Registry)
Sonderfälle	1a. Auf die angegebene Datenbank kann nicht zugegriffen werden

5.3.9 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 wurd nicht in das RRS eingefügt und es wird ein entsprechender Fehlerdialog angezeigt, der den Benutzer über die aufgetretenen Fehler informiert.
Normalablauf	1. Erstellung einer neuer Referenz 2. Eingeben aller benötigten Parameter 3. Referenz einfügen
Sonderfälle	3a. Fehler beim Einfügen der Referenz in das RRS

5.3.10 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

5.3.11 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

5.4 Anwendungsfälle der Datenquellen-Administratoren

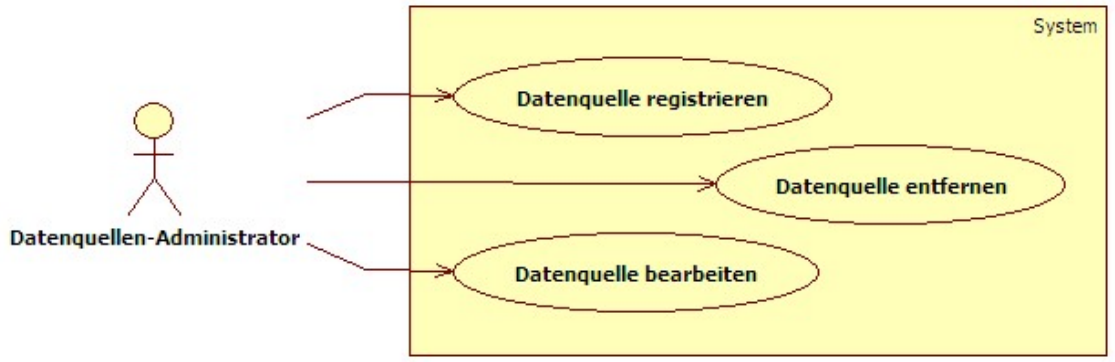


Abbildung 5: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.4.1 Datenquelle registrieren

Ziel	Registrierung einer Datenquelle in der Datenquellen-Registry.
Vorbedingung	Die Datenquellen-Registry ist erreichbar und der Benutzer hat die erforderlichen Rechte.
Nachbedingung	Eine neue Datenquelle wurde anhand der Benutzerangaben in der Datenquellen-Registry korrekt registriert und kann nun in Workflows verwendet werden.
Nachbedingung im Sonderfall	
Normalablauf	1. Anlegen eines neuen Datenquellenobjekts 2. Registrierung des Datenquellenobjekts in der Datenquellen-Registry
Sonderfälle	

5.4.2 Datenquelle entfernen

Ziel	Entfernen einer vorhandenen Datenquelle aus der Datenquellen-Registry
Vorbedingung	Die Datenquellen-Registry ist erreichbar und der Benutzer hat die erforderlichen Rechte.
Nachbedingung	Die vom Benutzer ausgewählte Datenquelle wurde korrekt und vollständig aus der Datenquellen-Registry entfernt und kann nun nicht mehr verwendet werden.
Nachbedingung im Sonderfall	
Normalablauf	1. Auswahl der Datenquelle 2. Entfernen der Datenquelle aus der Datenquellen-Registry
Sonderfälle	

5.4.3 Datenquelle bearbeiten

Ziel	Entfernen einer vorhandenen Datenquelle aus der Datenquellen-Registry
Vorbedingung	Die Datenquellen-Registry ist erreichbar und der Benutzer hat die erforderlichen Rechte.
Nachbedingung	Die vom Benutzer ausgewählte Datenquelle wurde korrekt und vollständig anhand der vom Benutzer durchgeführten Änderungen (Änderung der Eigenschaften, der Adresse, usw.) aktualisiert.
Nachbedingung im Sonderfall	
Normalablauf	1. Auswahl der Datenquelle 2. Bearbeiten der Werte der Datenquelle
Sonderfälle	

5.5 Anwendungsfälle der ODE Workflow-Engine

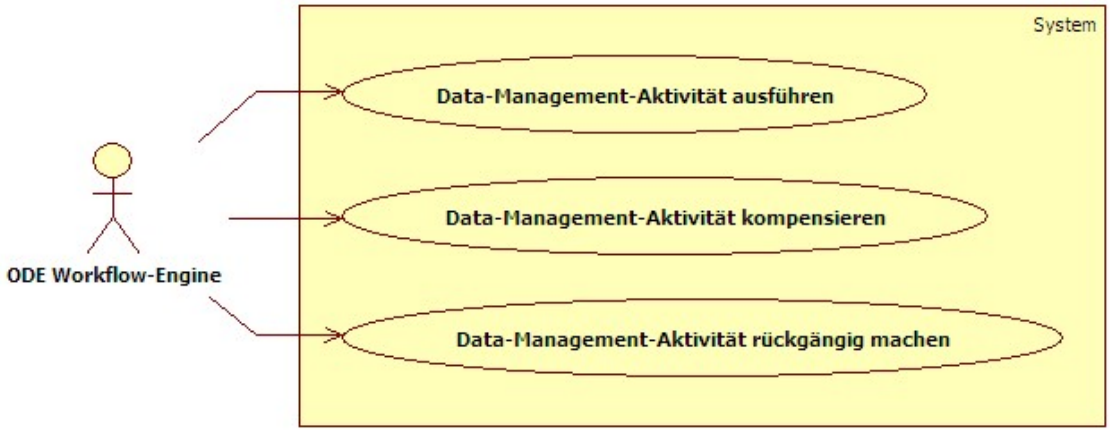


Abbildung 6: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.5.1 Data-Management-Aktivität ausführen

Ziel	Ausführen einer Data-Management-Aktivität.
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.5.2 Data-Management-Aktivität kompensieren

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.5.3 Data-Management-Aktivität rückgängig machen (rollback)

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.6 Anwendungsfälle des Eclipse BPEL Designers

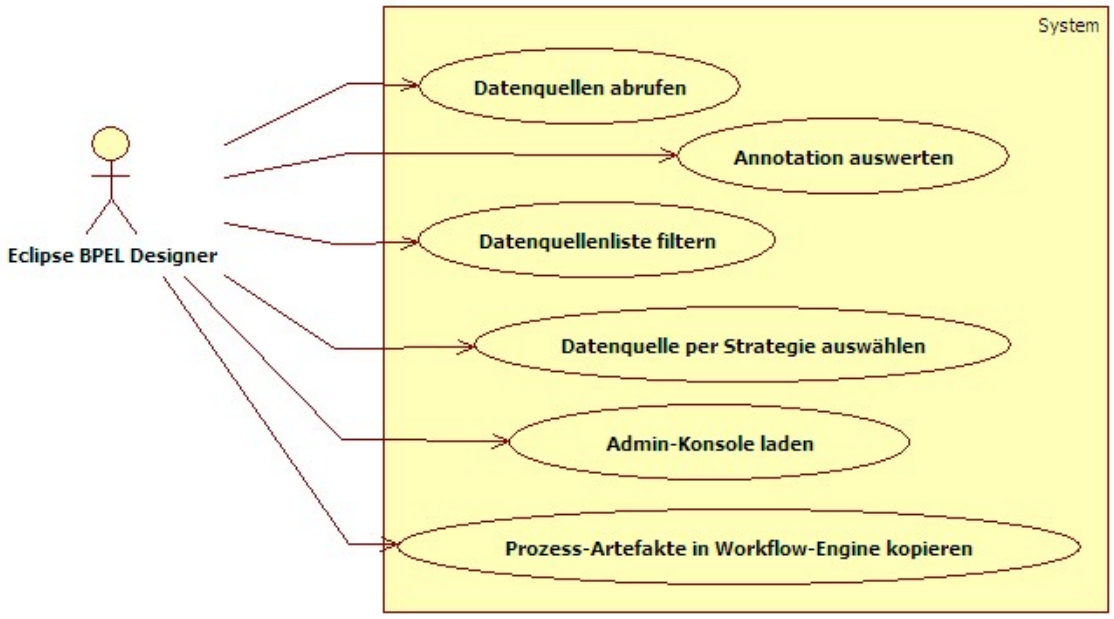


Abbildung 7: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.6.1 Prozess-Artefakte in Workflow-Engine kopieren

Ziel	Kopieren aller Prozess-Artefakte (Deployment Deskriptor, BPEL-Dateien, WSDL-Dateien) eines Projekts in die Worflow-Engine.
Vorbedingung	
Nachbedingung	Alle Prozess-Artefakte befinden sich nun im Verzeichnis der Workflow-Engine und eine .deployed Datei wurde erstellt.
Nachbedingung im Sonderfall	Die kopierten Dateien sind unverändert, es wurde keine .deployed Datei erstellt und die aufgetretenen Fehler werden in der ODE-Konsole im Eclipse BPEL Designer angezeigt.
Normalablauf	1. Kopieren der Prozess-Artefakte 2. Prozess wird automatisch deployed
Sonderfälle	2a. Fehler beim Deployment des Prozesses

5.6.2 Datenquellen abrufen

Ziel	Abrufen aller verfügbaren Datenquellen.
Vorbedingung	Die Datenquellen-Registry ist erreichbar.
Nachbedingung	Eine Liste mit allen verfügbaren Datenquellen steht zur Verfügung.
Nachbedingung im Sonderfall	
Normalablauf	1. Mit Datenquellen-Registry verbinden 2. Verfügbare Datenquellen abfragen
Sonderfälle	

5.6.3 Annotation auswerten

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.6.4 Datenquellenliste filtern

Ziel	Filterung der Datenquellenliste anhand von Eigenschaften.
Vorbedingung	Der Benutzer hat Werte für Datenquellen-Eigenschaften, die erfüllt werden sollen, angegeben.
Nachbedingung	Die durch die Filterung resultierende Datenquellenliste enthält genau die Datenquellen, die die angegebenen Eigenschaften erfüllen und sonst keine weiteren.
Nachbedingung im Sonderfall	Die Liste aller verfügbarer Datenquellen bleibt ungefiltert und unverändert, damit hat der Benutzer noch die Möglichkeit eine Datenquelle "per Hand" auszuwählen.
Normalablauf	1. Eigenschaftswerte werden gelesen 2. Liste aller verfügbarer Datenquellen wird anhand der Eigenschaften gefiltert
Sonderfälle	1a. Eigenschaftswerte können nicht gelesen werden 2a. Keine der verfügbaren Datenquellen erfüllt die Eigenschaften

5.6.5 Datenquelle per Strategie auswählen

Ziel	Auswahl einer Datenquelle per Strategie.
Vorbedingung	Der Benutzer hat Eigenschaften angegeben anhand derer die Liste der verfügbaren Datenquellen gefiltert wurde und eine korrekte und vollständige Datenquellenliste steht zur Verfügung.
Nachbedingung	Eine der ausgewählten Strategie entsprechende Datenquelle wurde ausgewählt und kann nun im Prozess verwendet werden.
Nachbedingung im Sonderfall	Es wurde eine Datenquelle nach der Standard-Strategie (siehe 6.5) aus der Liste ausgewählt und kann nun im Prozess verwendet werden.
Normalablauf	1. Filterung der Datenquellenliste 2. Lesen der ausgewählten Strategie aus den globalen Einstellungen 3. Auswahl einer Datenquelle aus der Liste anhand der gegebenen Strategie
Sonderfälle	2a. Beim Lesen der ausgewählten Strategie tritt ein Fehler auf

5.6.6 Admin-Konsole laden

Ziel	Laden der Inhalte der Admin-Konsole.
Vorbedingung	
Nachbedingung	Alle Werte der Datei mit den Einstellungen/Eingaben (Inhalten) der Admin-Konsole wurden geladen und im Fenster “Admin-Konsole” und dessen Unterfenstern angezeigt.
Nachbedingung im Sonderfall	Es wurden Standard-Einstellungen, die im Quellcode hinterlegt sind, geladen und im Fenster “Admin-Konsole” und dessen Unterfenstern angezeigt.
Normalablauf	1. Laden der Werte aus der Datei 2. Füllen der Felder der Admin-Konsole
Sonderfälle	1a. Beim Laden der Werte tritt ein Fehler auf

5.7 Anwendungsfälle des SIMPL Core

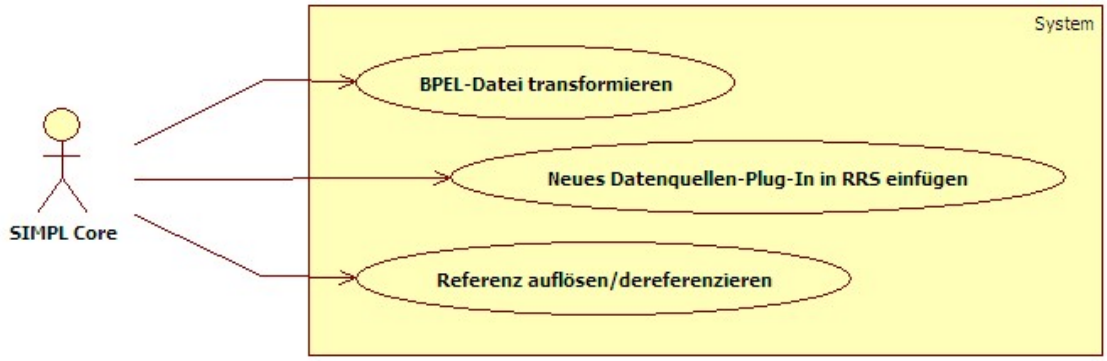


Abbildung 8: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.7.1 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 6.1.2) 3. Generierung des neuen BPEL-standard-konformen erweiterten Prozesses
Sonderfälle	3a. Bei der Generierung tritt ein Fehler auf

5.7.2 Neues Datenquellen-Plug-In in RRS einfügen

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.7.3 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	

6 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 Datenmanagement Aktivitäten zur Realisierung einer SQL-inline Unterstützung für BPEL. Ebenso werden die benötigten Eclipse BPEL Designer Erweiterungen, sowie die Realisierung des Datenquellen-Monitorings und das dafür zugrunde liegende Event-Modell beschrieben.

6.1 Referenzen in BPEL

Da bereits ein Konzept und umfassende Erkentnisse zur Bereitstellung und Verwaltung von Referenzen in BPEL, durch Quelle 3 *("Towards Reference Passing in Web Service and Workflow-based Applications")* vorlag, wurden an dieser Stelle nur die Ergebnisse des Dokuments zusammengefasst und auf unsere Bedürfnisse reduziert.
In BPEL werden Daten immer in Variblen gespeichert und *by value* (die Werte) innerhalb des Workflows weitergegeben. Dies hat zur Folge, dass gerade bei großen Datenmengen, wie sie in wissentschaftlichen 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 Quelle 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 wissentschaftliche 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 firmenweites 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, die in Abschnitt 6.1.2 beschrieben werden, realisiert. Das externe Verwaltungssystem der Referenzen, das sogenannte Reference Resolution System, wird in Abschnitt 6.1.1 näher erläutert.

6.1.1 Reference Resolution System

Abbildung **??** 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.

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 *enpoint reference* (EPR) als Eingabe. Das Konzept der endpoint references wird im Web Services Addressing Standard [4] 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 **??** 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.

6.1.2 Referenzen und Endpoint References

Nachdem nun die Architektur des Systems beschrieben wurde, folgt jetzt 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) beschrieben [4]. 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, 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 wärend 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"*.

6.1.3 Realisierung von Referenzen in BPEL

In [3] werden zwei verschiedene Ansätze (siehe Abbildung **??**) 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 entscheidenen 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>`s 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 Prozessmodel sichtbar sein, dies wird durch die Generierung eines entsprechenden `<partnerLink>`'s 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.

6.2 BPEL-SQL

In diesem Abschnitt werden die zu realisierenden Datenmanagement-Patterns zur Realisierung einer SQL-inline Unterstützung vorgestellt und im weiteren Verlauf deren Umsetzung für die verschiedenen Datenquellen, d.h. für Dateisysteme, Datenbanken und Sensornetze, erläutert. Aus den in Abschnitt 6.2.2 identifizierten Methoden und Funktionen werden dann in Abschnitt 6.2.3 neue benötigte BPEL-Aktivitäten abstrahiert, die dann später die entpsrechenden Methoden und Funktionen realisieren werden.

6.2.1 Datenmanagement Patterns

In diesem Kapitel werden die Datenmanagement Patterns, die zur Realisierung einer SQL-inline Unterstützung benötigt werden, aufgeführt und beschrieben (siehe Quelle [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 extern gespeichert oder direkt im Prozessspeicher gehalten werden.

Set IUD Pattern

Das Set IUD Pattern beschreibt die Möglichkeit mengenorientiertes Einfügen (insert), Aktualisieren (update) und Löschen (delete) auf externen Daten durchführen zu können.

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 Verarbeitung von komplexen Daten meist durch Stored Procedures erfolgt, ist es bei der 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 Prozessspeichers verarbeiten zu können. Das Set Retrieval Pattern liefert dafür eine mengenorientierte Datenstruktur, in die man die angefragten externen Daten innerhalb des Prozessspeichers ablegen kann. Diese Datenstruktur verhält sich dabei wie ein Cache im Prozessspeicher, 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 beinhaltet einfügen (insert), aktualisieren (update) und löschen (delete) von Daten im Datencache.

Synchronization Pattern

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

6.2.2 Umsetzung der Datenmanagement Patterns

In diesem Kapitel wird die Realisierung der acht Datenmanagement Patterns aus Abschnitt 6.2.1 im Zusammenhang mit den jeweiligen Datenquellentypen beschrieben. Es wird versucht eine einheitliche Realisierung zu erreichen, um die Usability möglichst hoch zu halten. Dafür sollen alle Befehle, die der Benutzer angeben kann, eine SQL bzw. XQuery konforme Befehlsform haben. D.h. sollte es im Moment nicht möglich sein einen benötigten Befehl in SQL oder XQuery anzugeben, so werden im Rahmen unserer Erweiterungen neue Befehlsformen erstellt, die dann auch Zugriffe auf Dateien oder Ähnliches realisieren und trotzdem nicht stark von der SQL- bzw. XQuery-Syntax abweichen.

Dateisysteme

Hier wird die Umsetzung der in Abschnitt 6.2.1 beschriebenen Patterns im Bereich der Dateisysteme durch SQL oder XQuery ähnliche Befehle, die intern durch bestehende Java-Methoden oder falls erforderlich durch die Definition neuer zu implementierender Funktionen realisiert werden, beschrieben.

- Query Pattern: Realisierung durch eine Select-Methode, die intern java.io Methoden verwendet, um Inhalte aus Dateien oder komplette Dateien aus einem Dateisystem zu lesen. Dafür muss noch ein Konzept erarbeitet werden, wie entsprechende Daten gezielt über ein SQL-Select ähnlichen Befehl oder XQuery (bei XML-Dateien oder eventuell auch Anderen) ausgewählt und abgefragt werden können.
  - Beispiel für eine CSV-Datei: SELECT * FROM *DATEINAME* WHERE *SPALTE1* LIKE 'abc', wobei Spalte1 der erste Wert aus der comma-separated-values-Liste ist.
- Set IUD Pattern: Realisierung durch entsprechende INSERT, UPDATE und DELETE-Methoden, 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 SQL ähnliche Befehle oder XQuery (bei XML-Dateien oder eventuell auch Anderen) eingefügt und gelöscht werden können.
  - Beispiel für eine CSV-Datei:

    - INSERT INTO *DATEINAME* VALUES (hans, meyer, 12345, Musterstadt) [AT END, AT BEGINING, AS *3*, SORTED, ...]
    - UPDATE *DATEINAME* SET *SPALTE3* = 'Chef' WHERE *SPALTE1* = '000290'
    - DELETE FROM *DATEINAME* WHERE *SPALTE1* = *'hans'* AND *SPALTE2* = *'meyer'*

- Data Setup Pattern: Realisierung durch die Verwendung entsprechender CREATE-Methoden, die intern durch java.io-Methoden realisiert sind. Hier sollte es auf jeden Fall möglich sein, Dateien und Ordner in einem Dateisystem erzeugen zu können. Ebenso sollte optional das Löschen auf der gleichen Ebene, d.h. von ganzen Dateien und Ordnern, möglich sein.

  – Beispiele:
    - CREATE FILE 'datei.csv' INTO 'uri (z.B. C:/)'
    - CREATE FOLDER 'test' INTO 'uri (z.B. C:/)'
    - DROP FILE 'datei.csv'
    - DROP FOLDER 'test'

- Stored Procedure Pattern: Findet im Rahmen der Dateisysteme keine Verwendung.

- Set Retrieval Pattern: Realisierung durch die Definition und Bereitstellung einer entsprechenden Activity, die Daten kapselt und im Prozessspeicher halten kann, für BPEL. Eine solche Activity liefert dann die Möglichkeit, dass in ihren Ausprägungen Ergebnisse von Queries (siehe Query Pattern), die über die SDO API abstrahiert wurden, innerhalb des Prozessspeichers abgelegt werden können. Die Firma IBM hat mit ihrer *Business Integration Suite* und dem darin enthaltenen *WebSphere Integration Developer* bereits eine Umsetzung dieses Patterns als Activity mit der Bezeichnung *"Retrieve Set Activity"* realisiert. Dabei werden in einer Retrieve Set Activity externe Daten in eine XML-Struktur innerhalb des Prozessspeichers geladen, eine etwas ausführlichere Beschreibung liefert [2].

- Set Access Pattern: Um das Set Access Pattern zu realisieren, müssen Methoden, die eine sequentielle und direkte Bearbeitung des Datencache einer RetrieveSet Activity realisieren, bereitgestellt werden. D.h. konkret, dass es möglich sein muss, mit der definierten Activity innerhalb von BPEL zu arbeiten. Die SDO API liefert dafür bereits einige Methoden die verwendet werden können. Es muss weiterhin möglich sein, dass eine entsprechende Aktivität auch in Containern, wie z.B. einer ForEach Activity, korrekt ausgeführt wird.

- Tuple IUD Pattern: Erweitert die Set Access Pattern Methoden um die Möglichkeiten Werte innerhalb der mengenorientierten Datenstruktur im Prozessspeicher 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 DAS API liefern dafür bereits einige Methoden die verwendet werden können.

Datenbanken

Hier wird die Umsetzung der in Abschnitt 6.2.1 beschriebenen Patterns im Bereich der Datenbanken durch bestehende SQL und XQuery-Befehle oder falls erforderlich durch die Definition neuer zu implementierender Funktionen beschrieben.

- Query Pattern: Realisierung durch SQL-SELECT oder entsprechende XQuery-Befehle.

  - Beispiele:
    * SELECT Info FROM Customer
    * XQUERY db2-fn:xmlcolumn ('CUSTOMER.INFO')

- Set IUD Pattern: Realisierung durch SQL-INSERT, SQL-UPDATE, SQL-DELETE oder entsprechende XQuery-Befehle.
  - Beispiele:
    * INSERT INTO DEPARTMENT VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
    * UPDATE EMPLOYEE SET JOB = 'LABORER' WHERE EMPNO = '000290'
    * DELETE FROM DEPARTMENT WHERE DEPTNO = 'D11'
    * xquery transform copy $mycust := db2-fn:sqlquery('select info from customer where cid = 1004') modify do insert <billto country="Canada"> <street>4441 Wagner</street> <city>Aurora</city> <prov-state>Ontario</prov-state> <pcode-zip> N8X 7F8</pcode-zip> </billto> after $mycust/customerinfo/phone[last()] return $mycust
    * UPDATE Customer SET info = XMLQUERY('declare default element namespace "http://posample.org"; transform copy $newinfo := $info modify do delete ($newinfo/customerinfo/phone) return $newinfo' passing info as "info") WHERE cid = 1002
    * xquery transform copy $mycust := db2-fn:sqlquery('select INFO from CUSTOMER where Cid = 1003') modify do delete $mycust/customerinfo/phone[@type!="home"] return $mycust

- Data Setup Pattern: Realisierung durch SQL-CREATE SCHEMA, SQL-CREATE TABLE, SQL-CREATE VIEW und weitere SQL-CREATE Befehle. Falls optional auch das Löschen dieser Objekte möglich sein soll, wird dies 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 DEPARTX
    * CREATE VIEW ADMINISTRATOR.KUND_WITH_ADR AS SELECT KUNDEN.KUNDEN_NR, KUNDEN.VORNAME, KUNDEN.NACHNAME, ADRESSEN.STRASSE, ADRESSEN.POSTLEITZAHL, ADRESSEN.STADT FROM "SYSTEM".KUNDEN AS KUNDEN, "SYSTEM".ADRESSEN AS ADRESSEN WHERE KUNDEN.KUNDEN_NR = ADRESSEN.KUNDEN_NR
    * DROP SCHEMA INTERNAL
    * DROP TABLE TDEPT
    * DROP VIEW ADMINISTRATOR.KUND_WITH_ADR

- Stored Procedure Pattern: Realisierung durch [(optional) SQL-CREATE PROCEDURE und] SQL-CALL (Aufruf über JDBC API möglich).
  - Beispiele:
    * CALL PARTS_ON_HAND (?,?,?) *(? = Parameter der Prozedur)*

- Set Retrieval Pattern: Realisierung durch die Definition und Bereitstellung eines mengenorientierten Datentyps oder einer entsprechenden Activity, die solch einen Datentyp kapselt, für BPEL.

Ein solcher Datentyp/Activity liefert dann die Möglichkeit, dass in Ausprägungen dieses Datentyps/Activity, Ergebnisse von Queries (siehe Query Pattern), die über die SDO API abstrahiert wurden, innerhalb des Prozessspeichers abgelegt werden können. Die Firma IBM hat mit ihrer *Business Integration Suite* und dem darin enthaltenen *WebSphere Integration Developer* bereits eine Umsetzung dieses Patterns als Activity mit der Bezeichnung *"Retrieve Set Activity"* realisiert. Dabei werden in einer Retrieve Set Activity externe Daten in eine XML-Struktur innerhalb des Prozessspeichers geladen, eine etwas ausführlichere Beschreibung liefert [2].

- Set Access Pattern: Um das Set Access Pattern zu realisieren, müssen Methoden, die eine sequentielle und direkte Bearbeitung der mengenorientierten Datenstruktur realisieren, bereitgestellt werden. D.h. konkret, dass es möglich sein muss, mit dem definierten mengenorientierten Datentyp innerhalb von BPEL zu arbeiten. Die SDO API liefert dafür bereits einige Methoden die verwendet werden können. Bei der Kapselung als Aktivität muss es auch möglich sein, dass die entsprechende Aktivität auch in Containern, wie z.B. einer ForEach Activity, entsprechend korrekt ausgeführt wird.

- Tuple IUD Pattern: Erweitert die Set Access Pattern Methoden um die Möglichkeiten Werte innerhalb der mengenorientierten Datenstruktur im Prozessspeicher 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 und ebenso SQL-Befehle um die Daten aus dem Prozesscache auf die Datenquelle zu übertragen. Die SDO API und die DAS API liefern dafür bereits einige Methoden die verwendet werden können.

Sensornetze

Hier wird die Umsetzung der in Abschnitt 6.2.1 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-Befehls Beispiele beziehen sich hier auf den SQL-Dialekt der Sensornetz-Datenbank TinyDB.

- Query Pattern: Realisierung durch ein entsprechendes SQL-SELECT der Sensornetz-Datenbank.
  - Befehlsstruktur: SELECT select-list [FROM sensors] WHERE where-clause [GROUP BY gb-list [HAVING having-list]][TRIGGER ACTION command-name[(param)]] [EPOCH DURATION integer]
  - 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*)

- Set IUD Pattern: Dieses Pattern wird von Sensornetz-Datenbanken nicht unterstützt, da Sensoren nur ausgelesen und nicht beschrieben werden können. Darum ist das Einfügen, Aktualisieren und Löschen von externen Daten auf einer Sensornetz-Datenbank nicht möglich. Es besteht lediglich die Möglichkeit sensornetzintern in einem Buffer Werte zwischenzuspeichern und den Buffer zu löschen. 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:
    * Werte einfügen in Buffer-Tabelle: SELECT field1, field2, ... FROM sensors SAMPLE PERIOD x INTO 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, uint16, int8, int16, int32} und* ***field1*** *,* ***field2*** *, usw. sind Spaltennamen, die wie der Tabellenname jeweils 8 Zeichen lang sein dürfen.*
  - Alle Buffer-Tabellen löschen: DROP ALL

- Stored Procedure Pattern: Es gibt für die TinyDB einen *stored procedures* 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 Datenbanken beschrieben.
- Set Access Pattern: Realisierung ebenso wie bei Datenbanken beschrieben.
- Tuple IUD Pattern: Realisierung ebenso wie bei Datenbanken beschrieben.
- Synchronization Pattern: Hier gilt dasselbe wie für das Set IUD Pattern. Da es nicht möglich ist, Daten von außen in die Sensornetz-Datenbank einzubringen, wird die Realisierung dieses Patterns nicht unterstützt bzw. benötigt.

6.2.3 Resultierende BPEL-Aktivitäten

In diesem Abschnitt werden die durch Abschnitt 6.2.2 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 *kind*, *dqAddress* und *statement* vom Typ String und *dqReference* vom Typ EPR (endpoint reference) besitzen. Die Variable *kind* dient zur Angabe des Datenquellentyps für den die Aktivität ausgeführt wird, also ob es sich um ein Dateisystem, eine Datenbank oder ein Sensornetz handelt. Diese Auswahl ist wichtig um intern den richtigen SQL-Dialekt für die entsprechende Datenquelle auszuwählen. Die Variable *dqAddress* dient zur Angabe der Datenquellenadresse und die Variable *statement* zur Haltung des entsprechenden SQL- oder XQuery-Befehls der ausgeführt werden soll. Die Variable *dqReference* ist eine *endpoint reference* auf eine Datenquelle und dient als Alternative zur Datenquellenadresse. Diese vier Variablen besitzt jede der definierten Aktivitäten. Darum werden diese nachfolgend

nicht jedesmal explizit 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 vorallem 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 auch alle Befehle in einer SQL-ähnlichen Syntax angegeben werden, um genau diese allgemeine Handhabung realisieren zu können. Wie diese Befehle nun intern verarbeitet werden, braucht der Benutzer nicht zu wissen und er kann dadurch schnellstmöglich mit nur "einer" Anfragesprache alle zur Verfügung gestellten Daten- und Datenquellenbefehle für alle unterstützten Datenquellen ausführen.

Select Activity

Diese Aktivität ermöglicht es aus jeder beliebigen Datenquelle Daten zu lesen. Weitere spezifische Attribute werden dafür nicht benötigt. Die Select Activity ist für alle Datenquellen gleich strukturiert und nur die entsprechenden SQL-Befehle unterscheiden sich.

Insert Activity

Diese Aktivität ermöglicht es Daten in jede Datenquelle einzufügen. Da dies für Sensornetze nicht relevant ist, wird diese Aktivität für Sensornetze für das sensornetzinterne Einfügen von Sensornetzdaten in Buffer-Tabellen genutzt (wie auch in Abschnitt 6.2.2 beschrieben).

Update Activity

Diese Aktivität ermöglicht es Daten aus Datenquellen mit externen Daten zu aktualisieren. Diese Aktivität existiert nicht für Sensornetze.

Delete Activity

Diese Aktivität ermöglicht es Daten auf beliebigen Datenquellen zu löschen. Diese Aktivität existiert nicht für Sensornetze.

Create Activity

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

- Dateisysteme
  - CREATE FOLDER
  - CREATE FILE
- Datenbanken
  - CREATE TABLE
  - CREATE SCHEMA
  - CREATE VIEW
- Sensornetze
  - CREATE BUFFER

Call Activity

Diese Aktivität ermöglicht es auf der Datenquelle hinterlegte Prozeduren auszuführen. Diese Aktivität existiert nur für Datenbanken.

RetrieveSet Activity

Diese Aktivität ermöglicht es Daten von Datenquellen in den Prozessspeicher zu laden.

WriteBack Activity

Diese Aktivität ermöglicht es Daten aus dem Prozessspeicher auf die originale Datenquelle zurückzuschreiben. Diese Aktivität existiert nicht für Sensornetze.

Optionale Aktivitäten:

- **Import Activity**: Diese Aktivität ermöglicht es lokale Daten (z.B. Simulationsparameter) des Benutzers in einen Prozess einzubinden und in diesem zu verwenden.
- **Export Activity**: Diese Aktivität ermöglicht es Daten eines Prozesses (z.B. Simulationsergebnisse) lokal auf den Benutzer-Rechner zu exportieren.
- **Move Activity**: 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.
- **Drop Activity**: Diese Aktivität ermöglicht es Zuordnungseinheiten auf beliebigen Datenquellen zu löschen. Dazu werden die folgenden Befehle benötigt:
  - Dateisysteme
    - DROP FOLDER
    - DROP FILE
  - Datenbanken
    - DROP TABLE
    - DROP SCHEMA
    - DROP VIEW
  - Sensornetze
    - DROP ALL

6.3 Eclipse BPEL Designer

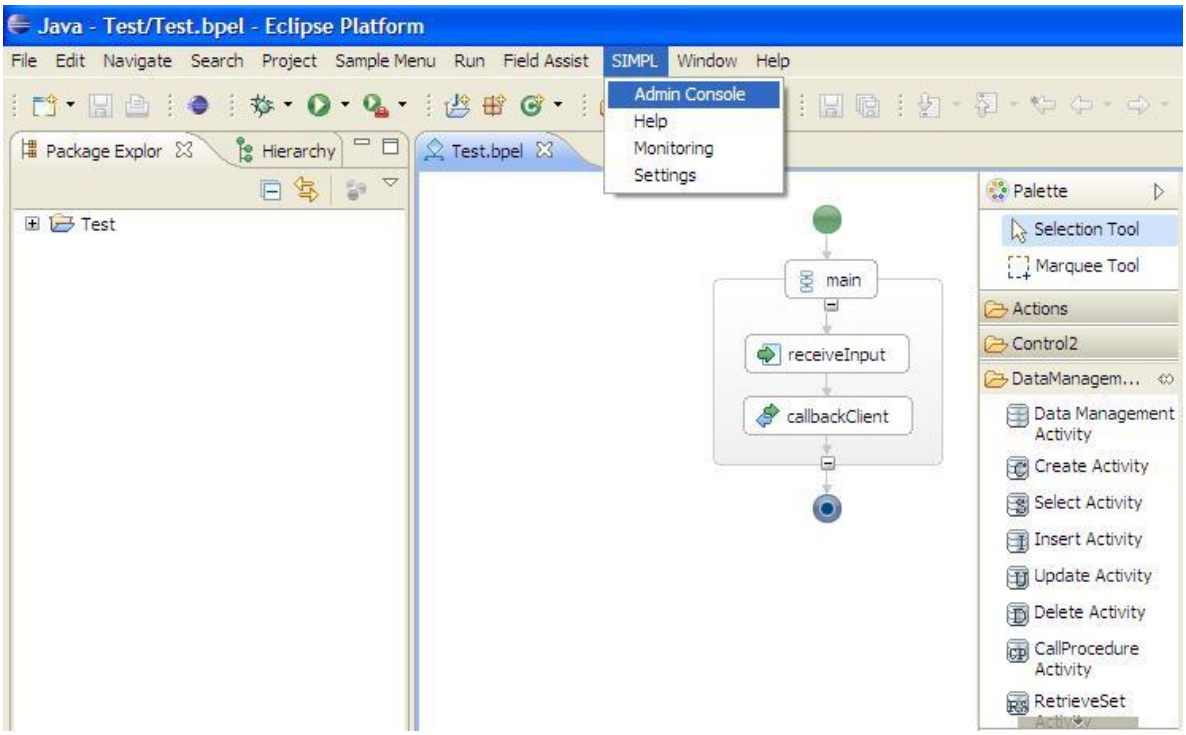


Abbildung 9: SIMPL Menü und Eclipse BPEL Designer mit einigen Data-Management-Aktivitäten

6.3.1 Monitoring

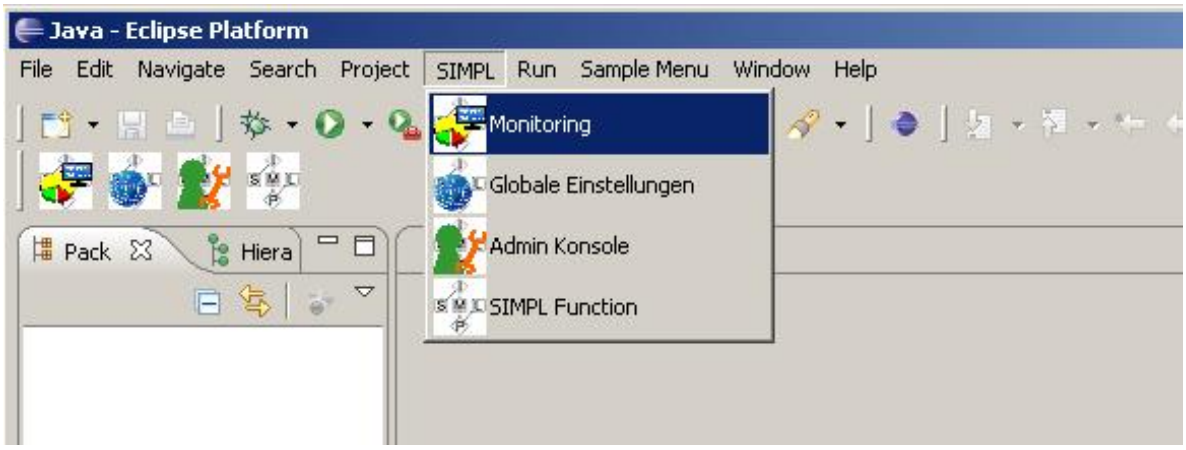


Abbildung 10: Monitoring im Menu

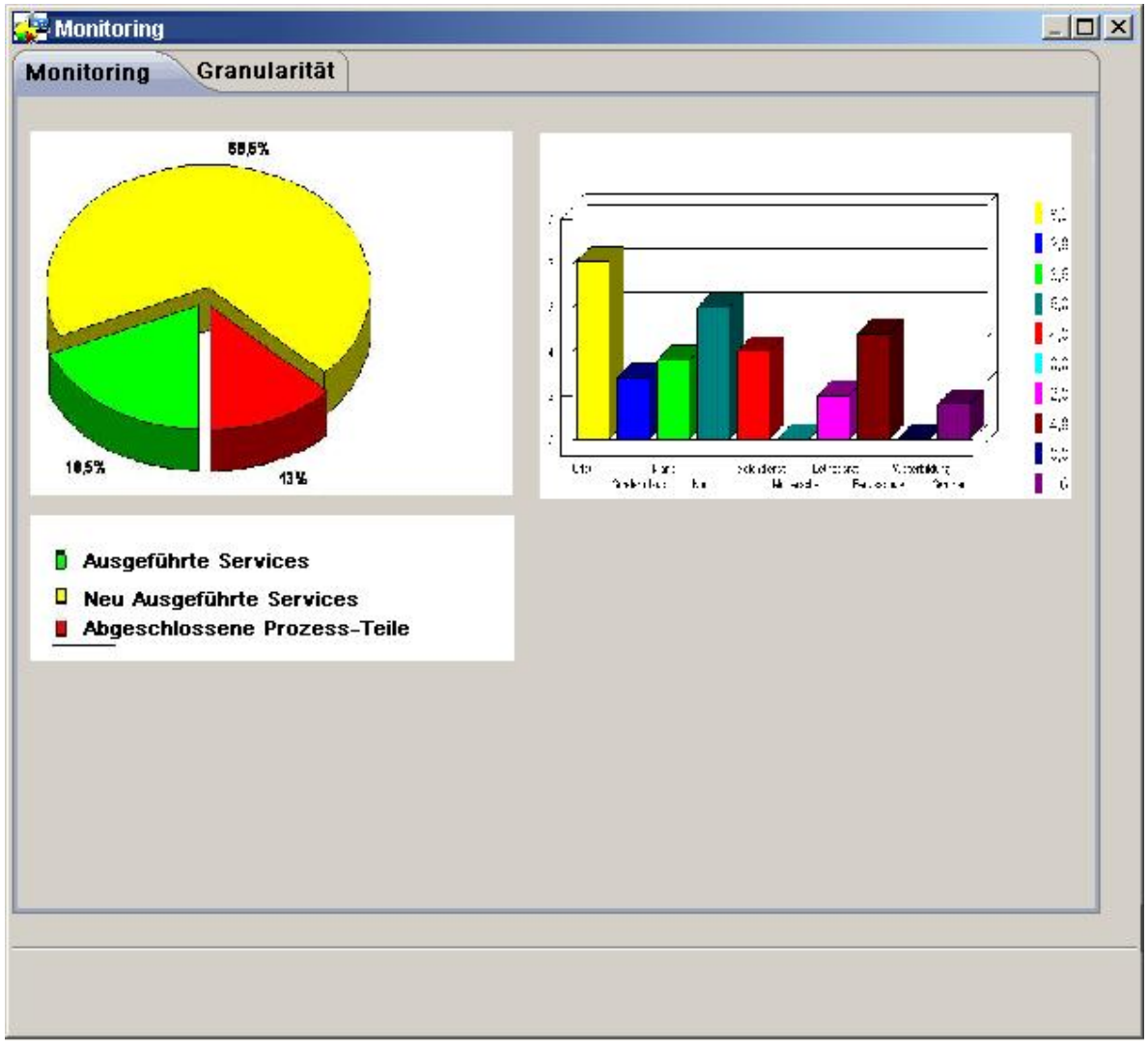


Abbildung 11: Monitoring Dialog

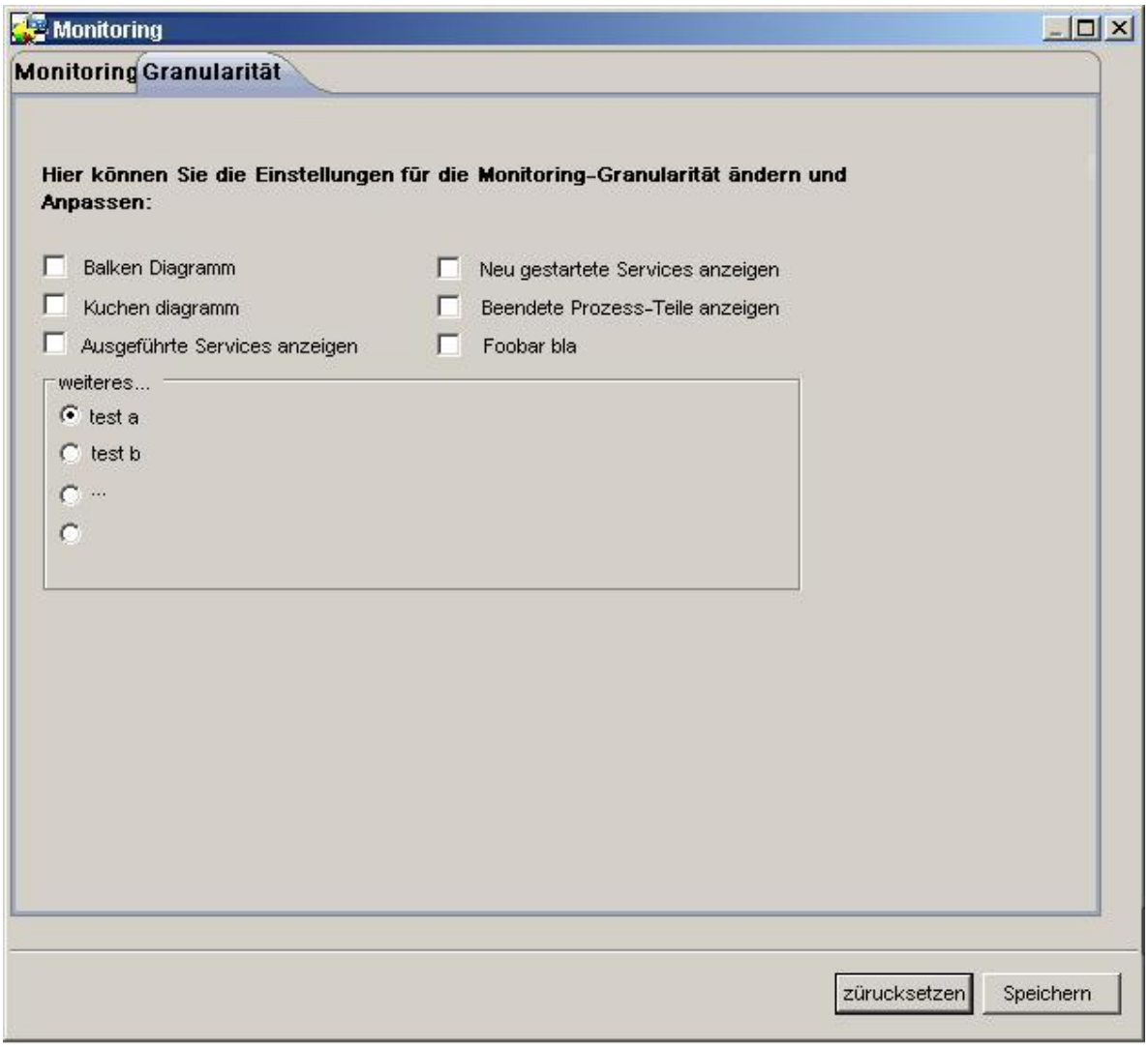


Abbildung 12: Monitoring-Granularität Dialog

6.3.2 Admin-Konsole

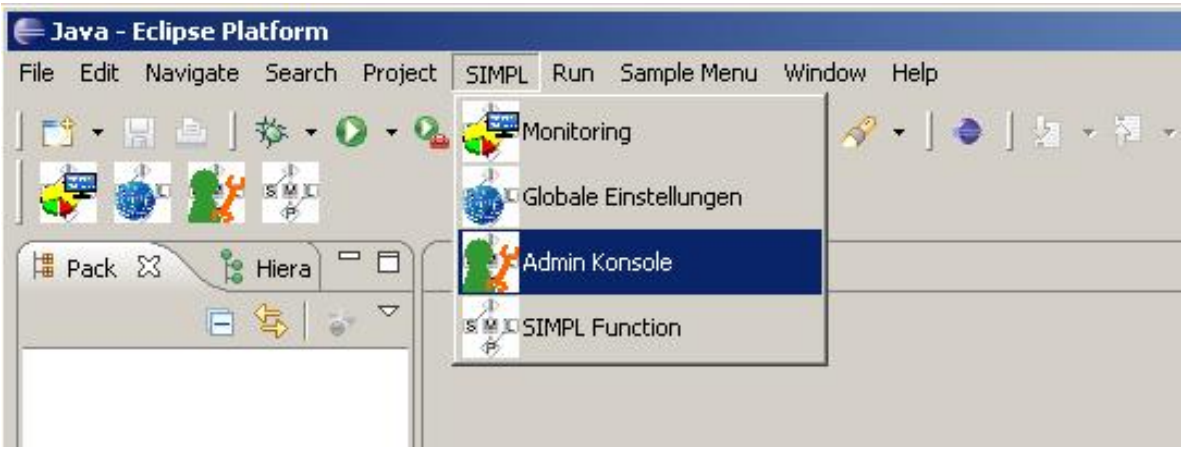


Abbildung 13: Admin-Konsole im Menu

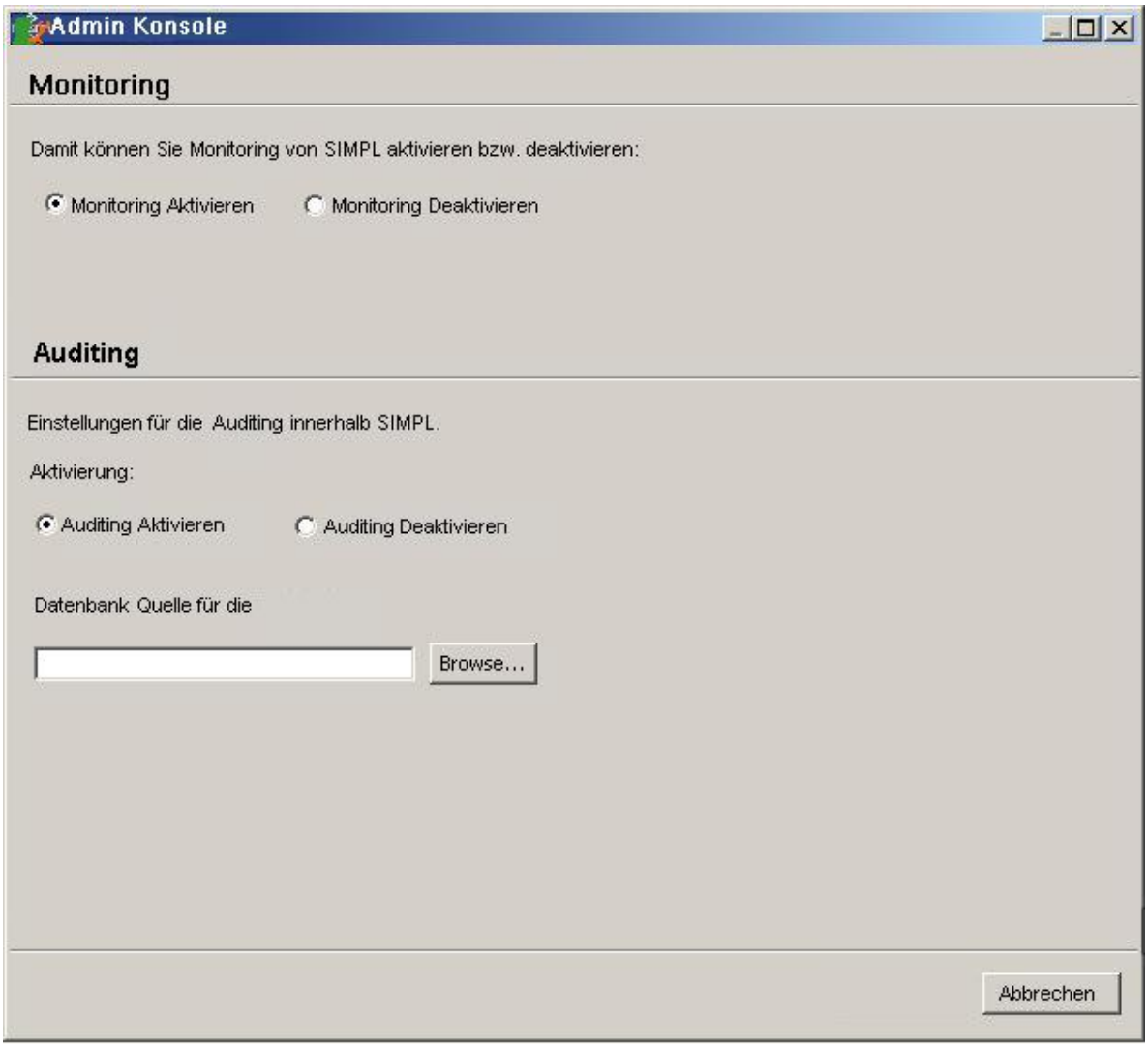


Abbildung 14: Admin-Konsole

6.3.3 Globale Eigenschaften

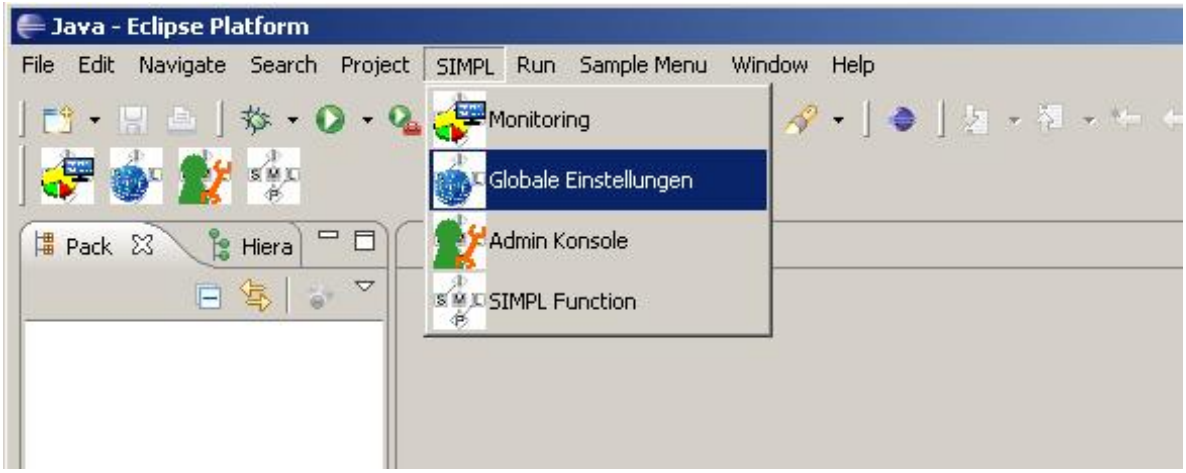


Abbildung 15: Globale Eigenschaften im Menu

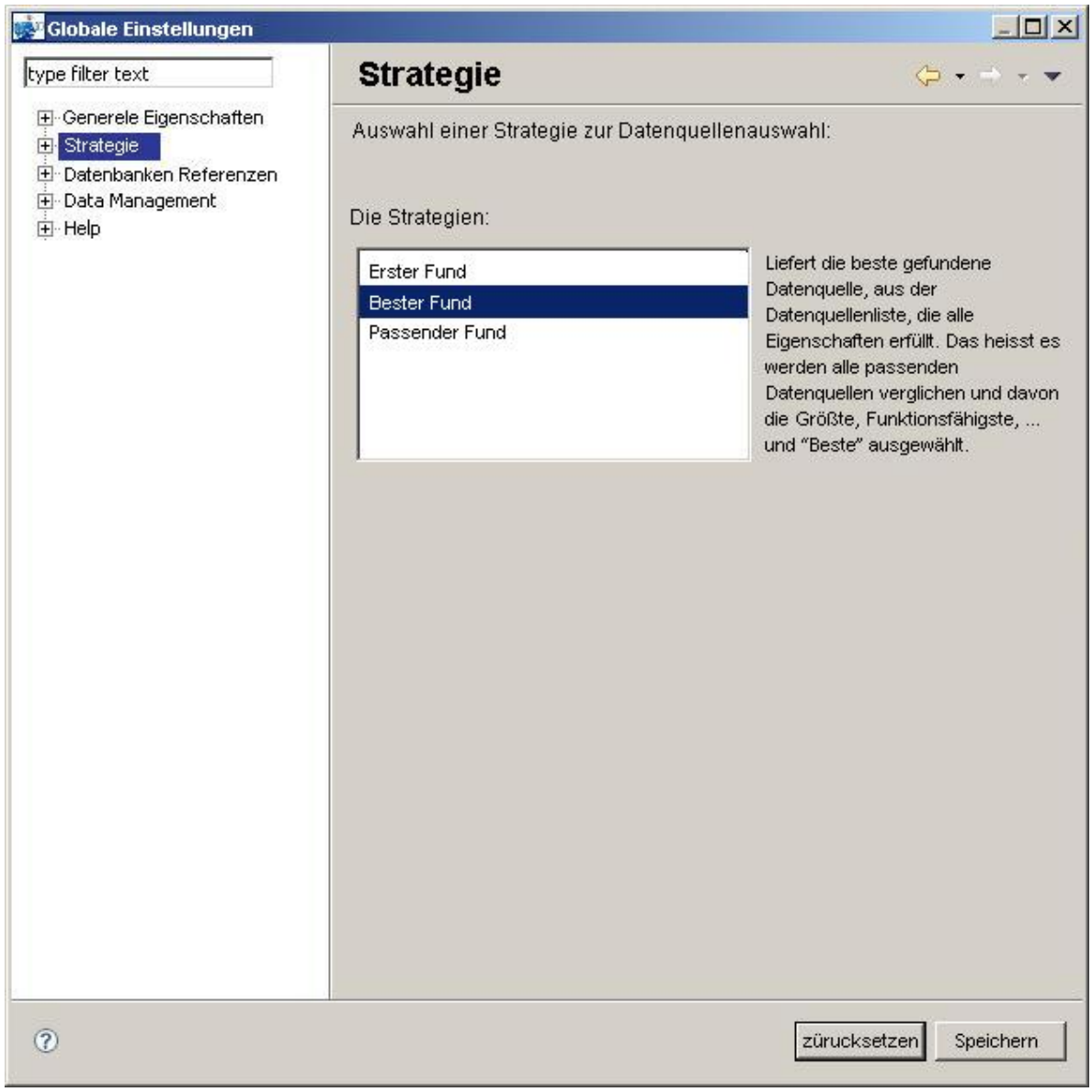


Abbildung 16: Globale Eigenschaften Dialog

6.3.4 Data-Management-Aktivitäten

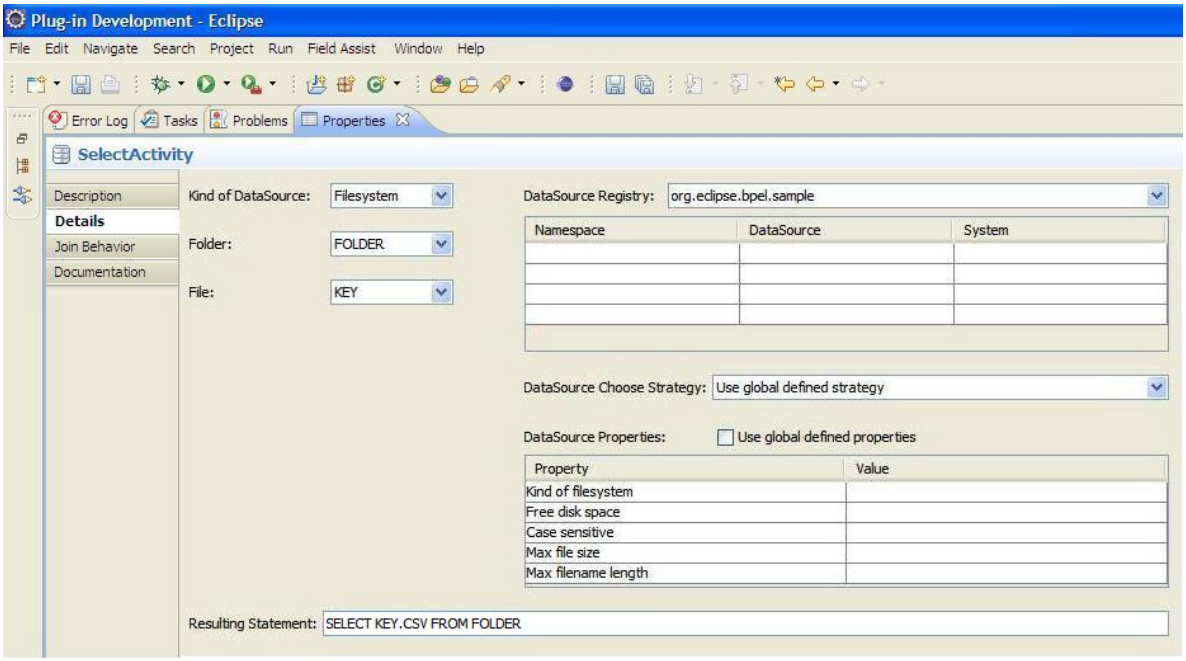


Abbildung 17: Eigenschaftsfenster einer Data-Management-Aktivität am Beispiel einer SELECT Activity

6.4 Monitoring und Auditing

Im BPEL Designer ist der Zugriff auf das Monitoring und Auditing unter dem entsprechenden Menüpunkt möglich. Hier kommt man nun zum entsprechenden Interface in dem sich verschiedene Einstellungen für das Auditing und das Monitoring tätigen lassen (siehe 6.3.1). Es ist möglich das Auditing als auch das Echtzeitmonitoring zu aktivieren oder zu deaktivieren.

6.4.1 Auditing

Wenn das Auditing aktiviert wird (mit Auswahl der entsprechenden Kriterien), wird eine Liste der Prozesse und/oder Instanzen erstellt (mit Hilfe der ODE Management API). Dadurch ist es nun möglich die verschiedene Informationen und aufgetreten Events (in Bezug auf die zuvor ausgewählten Kriterien) abzurufen. Diese Informationen und Events werden in Form einer xml-Datei übergeben. Diese kann nun (optinal) geparst werden um die Informationen als Strings in einer „normalen“ realtionalen Datenbank zu speichern oder aber direkt die xml-Datein in einer entsprechenden Datenbank abzuspeichern.

6.4.2 Monitoring

Das Monitoring von SIMPL wird als ein Echtzeitmonitoring realisiert, welches beginnt Informationen ab dem Zeitpunkt an dem es aktiviert wurde abzurufen (mit Hilfe von Event Listenern) und dies wird solange getan, bis es wieder deaktiviert wurde. Eine Visualisierung nach verschiedenen Gesichtspunkten, wie zum Beispiel Anzahl der aktiven Prozesse oder Prozessinstanzen soll möglich sein. Geplant sind bisher zwei Arten der Visualisierung die in der Konzept-Grafik in 6.3.1 zu sehen sind. Die Monitoring Informationen werden nicht gespeichert sondern dienen jedeglich der Visualisierung.

6.4.3 Event-Modelle

6.5 Strategien zur dynamischen Datenquellenauswahl

In diesem Abschnitt werden verschiedene Strategien, mithilfe derer später die Datenquellen über angegebene Eigenschaften ausgewählt werden, definiert und erläutert.

6.5.1 "Erster Fund" (First Find)

Diese Strategie liefert immer die erste passende Datenquelle, aus der Datenquellenliste, die alle vom Benutzer angegebenen Eigenschaften erfüllt. Der Vorteil dieser Strategie liegt klar in der Geschwindigkeit in der eine passende Datenquelle, falls eine solche existiert, gefunden wird. Als Nachteil muss man bei dieser Strategie in Kauf nehmen, dass man keinen weiteren Einfluss auf die Auswahl der Datenquelle hat, solange die Eigenschaften erfüllt werden. D.h. konkret, dass man bei dieser Strategie nicht weiß in welchem Maße die ausgewählte Datenquelle die angegebenen Eigenschaften erfüllt, also ob die Datenquelle die Eigenschaften gerade so erfüllt oder sogar eventuell unnötigerweise um ein vielfaches überbietet.

6.5.2 "Bester Fund" (Best Find)

Diese Strategie liefert die beste gefundene Datenquelle, aus der Datenquellenliste, die alle Eigenschaften erfüllt. Das heisst es werden alle passenden Datenquellen verglichen und davon die Größte, Funktionsfähigste, ... und "Beste" ausgewählt. Der Vorteil dieser Strategie ist, dass bei sich ändernden Anforderungen, wie z.B. eines höheren Speicherbedarfs, nicht zwangsweise der Prozess neu ausgeführt und die Datenquelle gewechselt werden müssen. Nachteile sind eine etwas längere Suchzeit, da alle verfügbaren und passenden Datenquellen miteinander verglichen werden müssen und ebenso wieder die Möglichkeit, dass eine "zu gute" Datenquelle, die nicht voll ausgelastet wird, ausgewählt wurde und dadurch möglicherweise denen die sie wirklich benötigen nicht mehr zur Verfügung steht.

6.5.3 "Passender Fund" (Matching Find (Standard-Strategie))

Diese Strategie liefert die passendste gefundene Datenquelle, aus der Datenquellenliste, die alle Eigenschaften erfüllt. Das heisst es werden alle passenden Datenquellen verglichen und davon diejenige ausgewählt, die gerade so alle Eigenschaften erfüllt und somit den Angaben des Benutzers entspricht. Der Vorteil dieser Strategie ist, dass dadurch verhindert wird, dass z.B. eine viel zu große Datenquelle, die Andere dringender bräuchten, ausgewählt wird. Die Nachteile sind wieder die etwas längere Suchzeit und die fehlende Flexibilität dynamisch auf steigende Anforderungen zu reagieren, da diese eben genau erfüllt werden.

6.6 Datenquellen-Eigenschaften

In diesem Abschnitt sollen sinnvolle Datenquellen-Eigenschaften, anhand derer Datenquellen verglichen und automatisch ausgewählt werden können, definiert werden. Dazu werden zuerst spezifische Eigenschaften der drei grundlegenden Datenquellenkategorien (Dateisysteme, Datenbanken und Sensornetze) identifiziert und anschließend um weitere gemeinsame bzw. allgemeine Eigenschaften ergänzt.

6.6.1 Dateisysteme

In Tabelle 1 werden nachfolgend einige für den Benutzer relevante Eigenschaften von Dateisystemen genannt und kurz beschrieben.

Tabelle 1: Dateisystem-Eigenschaften

Eigenschaft	Beschreibung
Art des Dateisystems	Zur Abfrage um welches Dateisystem es sich handelt: Linux (ext3), Apple-Macintosh (HFS+), Microsoft (NTFS) oder Andere.
Verfügbarer Speicherplatz	Der auf einem Dateisystem noch zur Verfügung stehende Speicherplatz.
Unterscheidet Groß-/ Kleinschreibung	Zur Abfrage, ob das Dateisystem bei Dateinamen oder Verzeichnispfaden zwischen Groß- und Kleinschreibung unterscheidet.
Maximale Dateigröße	Zur Abfrage, wie groß eine einzelne Datei maximal sein darf.
Maximale Dateinamenlänge	Maximale Anzahl von Zeichen, die ein Dateinamen lang sein darf.

6.6.2 Datenbanken

In Tabelle 2 werden nachfolgend einige für den Benutzer relevante Eigenschaften von Datenbanken genannt und kurz beschrieben.

Tabelle 2: Datenbank-Eigenschaften

Eigenschaft	Beschreibung
Art der Datenbank	Zur Abfrage um welche Art von Datenbank es sich handelt: Relationale DB, XML-DB, Objektrelationale DB oder Andere.
Datenbank-Version	Welche Versionsnummer die zugrundeliegende Datenbank besitzt.
Maximale Spaltenzahl in Tabellen	Wieviele Spalten in einer Tabelle erstellt werden können.
Maximale Anzahl von Tabellenangaben pro Befehl	Gibt an, wieviele Tabellen z.B. in einem SQL-SELECT Befehl angegeben werden können, d.h. aus wievielen Tabellen, Daten gemeinsam gelesen werden können.
Maximale Befehlslänge	Zur Abfrage, wie lang ein Befehl, der an die Datenbank geschickt wird, maximal sein darf.
Maximale Anzahl von Bytes pro Tabellenzeile	Gibt an, welche Menge an Daten (in Byte) in eine einzige Zeile einer Tabelle geschrieben werden können.
Maximale Länge von Tabellennamen	Zur Abfrage der maximalen Länge von Tabellennamen, die von der Datenbank unterstützt werden.
Maximale Anzahl an konkurrierenden Verbindungen	Liefert die maximale Anzahl an Verbindungen, die gleichzeitig (konkurrierend) auf Daten der Datenbank zugreifen.
Unterstützung von *stored procedures*	Zur Abfrage, ob die Verwendung von *stored procedures* von einer Datenbank unterstützt wird.
Unterstützung von Transaktionen	Zur Abfrage, ob Transaktionen von der Datenbank unterstützt werden.

6.6.3 Sensornetze

In Tabelle 3 werden nachfolgend einige für den Benutzer relevante Eigenschaften von Sensornetzen genannt und kurz beschrieben.

Tabelle 3: Sensornetz-Eigenschaften

Eigenschaft	Beschreibung

6.6.4 Gemeinsame/Allgemeine Eigenschaften

In Tabelle 4 werden nun nachfolgend noch alle relevanten gemeinsamen bzw. allgemeinen Eigenschaften, die für alle Datenquellen gelten, genannt und kurz beschrieben.

Tabelle 4: Gemeinsame/Allgemeine Eigenschaften aller Datenquellen

Eigenschaft	Beschreibung
Name	Der definierte Name einer Datenquelle.
Typ	Zur Abfrage um welche Art von Datenquelle es sich handelt: Dateisystem, Datenbank oder Sensornetz.
Adresse	Die physikalische Adresse der Datenquelle.
Benutzername	Wird für die Authentifizierung und Autorisierung auf zugriffsgeschützten Datenquellen benötigt.
Passwort	Wird für die Authentifizierung und Autorisierung auf zugriffsgeschützten Datenquellen benötigt.

7 Materialien, Werkzeuge und Technologien

7.1 Materialien

7.2 Technologien

Es werden die Verchiedene Technologien in Entwicklung eingesetzt:

ApacheODE http:/ode.apache.org

Apache ODE ist eine BPEL-fähige 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 das hot deplyoment von Prozessen möglich als auch die Analyse und Validierung von Prozessen.

Eclipse-Bpel-Designer http://www.eclipse.org/bpel

Der Eclipse BPEL Designer ist ein Eclipse Plugin zum Arbeiten mit BPEL Prozessen. Er verfügt über eine leicht verständliche GUI, eine Fehlererkennnung für BPEL Prozesse und außerdem ist die Schritt für Schritt Ausführung von Prozessen möglich.

IBM-DB2 http://pub...w/v9r5/index.jsp

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

7.3 Werkzeuge

Subversion http://svnbook.red-bean.com

Subversion ist eine Software zur Versionskontrolle und eine Weiterentwicklung von CVS.

Maven http://svnbook.red-bean.com

Maven ist ein Projekt-Management-Tool.

Lyx www.lyx.org

LaTeX http://www.latex-project.org

Eclipse-Subversion http://javathreads.de/2009/07/subversion-unter-eclipse-galileo-konfigurieren/

TortoiseSVN http://tortoisesvn.net/

Hudson https://hudson.dev.java.net/

Hudson ist ein webbasiertes System zur kontinuierlichen Integration von Softwareprojekten.

7.3.1 Entwicklungsumgebung

7.3.2 Sonstige Werkzeuge

8 Änderungsgeschichte

- **Version 0.1**, 11. September 2009: Erstellung des Dokuments.

Literatur

[1] Begriffslexikon

[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.

[4] W3C. Web Services Addressing 1.0 - Core, W3C Recommendation. http://www.w3.org/TR/ ws-addr-core/, 2006.