

## Studienprojekt SIMPL

---

# Spezifikation

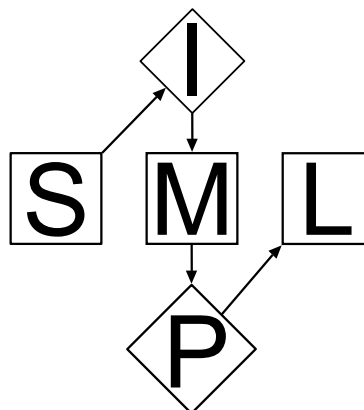
Version 1.0

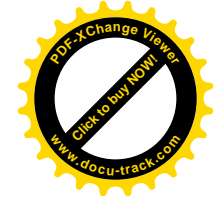
28. September 2009

Verfasser:

Wolfgang Huettig, Michael Hahn, Firas Zoabi, Michael Schneidt, René Rehn

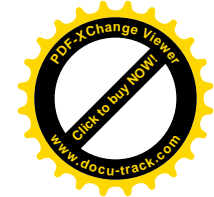
---



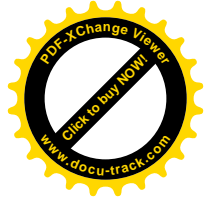
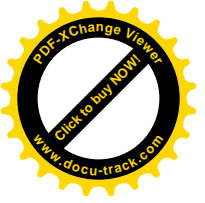


# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
1.1	Zweck des Dokuments . . . . .	4
1.2	Einsatzbereich und Ziele . . . . .	4
1.3	Evolution des Dokuments . . . . .	4
1.3.1	Basisfunktionalität . . . . .	4
1.3.2	Ausblick . . . . .	5
1.4	Definitionen . . . . .	5
1.5	Überblick . . . . .	5
<b>2</b>	<b>Allgemeine Beschreibung</b>	<b>5</b>
2.1	Einbettung . . . . .	6
2.2	Funktionen . . . . .	6
2.3	Sprache . . . . .	7
2.4	Distributionsform und Installation . . . . .	7
2.5	Benutzerprofile . . . . .	7
2.6	Einschränkungen . . . . .	7
<b>3</b>	<b>Nichtfunktionale Anforderungen</b>	<b>7</b>
3.1	Mengengerüst . . . . .	7
3.2	Benutzbarkeit . . . . .	8
3.3	Robustheit . . . . .	8
3.4	Sicherheit . . . . .	8
3.5	Portabilität . . . . .	8
3.6	Erweiterbarkeit . . . . .	8
3.7	Wartbarkeit . . . . .	8
3.8	Skalierbarkeit . . . . .	8
<b>4</b>	<b>Akteure</b>	<b>8</b>
4.1	Prozess-Modellierer . . . . .	9
4.2	Workflow-Administrator . . . . .	9
4.3	ODE Workflow-Engine . . . . .	9
4.4	Eclipse BPEL Designer . . . . .	9
<b>5</b>	<b>Anwendungsfälle (Use-Cases)</b>	<b>9</b>
5.1	Diagramm der Anwendungsfälle . . . . .	9
5.2	Anwendungsfälle der Prozess-Modellierer . . . . .	10
5.2.1	Data-Management-Aktivität erstellen . . . . .	11
5.2.2	Data-Management-Aktivität bearbeiten . . . . .	11
5.2.3	Data-Management-Aktivität löschen . . . . .	11
5.2.4	Prozess ausführen . . . . .	12
5.2.5	Admin-Konsole speichern . . . . .	12
5.2.6	Admin-Konsole zurücksetzen . . . . .	12
5.3	Anwendungsfälle der Workflow-Administratoren . . . . .	13
5.3.1	Admin-Konsole öffnen . . . . .	13
5.3.2	Auditing aktivieren . . . . .	13
5.3.3	Auditing deaktivieren . . . . .	13
5.3.4	Auditing-Datenbank festlegen . . . . .	14
5.4	Anwendungsfälle der ODE Workflow-Engine . . . . .	14
5.4.1	Data-Management-Aktivität ausführen . . . . .	15
5.4.2	Data-Management-Aktivität kompensieren . . . . .	15
5.5	Anwendungsfälle des Eclipse BPEL Designers . . . . .	16



5.5.1	Admin-Konsole laden . . . . .	16
<b>6</b>	<b>Konzepte und Realisierungen</b>	<b>16</b>
6.1	BPEL-SQL . . . . .	16
6.1.1	Datenmanagement Patterns . . . . .	16
6.1.2	Umsetzung der Datenmanagement Patterns . . . . .	17
6.1.3	Resultierende BPEL-Aktivitäten . . . . .	21
6.2	Eclipse BPEL Designer . . . . .	24
6.2.1	Admin-Konsole . . . . .	24
6.2.2	Data-Management-Aktivitäten . . . . .	27
6.3	Auditing . . . . .	28
6.3.1	Momentane Situation - Monitoring von ODE . . . . .	28
6.3.2	Auditing von SIMPL . . . . .	28
<b>7</b>	<b>Materialien, Werkzeuge und Technologien</b>	<b>29</b>
7.1	Materialien . . . . .	29
7.2	Technologien . . . . .	29
7.3	Werkzeuge . . . . .	30
7.3.1	Entwicklungsumgebung . . . . .	30
<b>8</b>	<b>Änderungsgeschichte</b>	<b>30</b>



# 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 Richtlinie für die Entwicklung der Software und den Kunden als Zwischenergebnis zur Kontrolle.

Zum Leserkreis dieser Spezifikation gehören:

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

## 1.2 Einsatzbereich und Ziele

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

## 1.3 Evolution des Dokuments

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

Darüberhinaus wird in weiterführenden Kundengesprächen über die Präferenzordnung von optionalen Implementierungen diskutiert, welche in den entsprechenden Iterationen umgesetzt werden sollen.



### 1.3.1 Basisfunktionalität



Die statische Anbindung von Datenquellen: Relationale Datenbank, XML-Datenbank, Sensor Datenbank und Dateisystem

- Eine grundlegende Adminkonsole, die Funktionen wie das An- und Ausschalten des Auditings und die Eingabe globaler Einstellungen für das Rahmenwerk.

- Bereitstellung von generischen BPEL-Aktivitäten für den Zugriff auf Datenquellen.



### 1.3.2 Ausblick

#### Planung

- Es wird ein Plug-In-System für die Anbindung von verschiedenen Datenquellen umgesetzt.
- Implementierung einer **Datenquellen Registry** **mithilfe derer** Datenquellen über den Eclipse BPEL Designer ausgewählt werden können.
- **Beispiel-Plugin für Dateitypen.**



#### Optional (Kunden Präferenz)

- Unterstützung von Referenzen in BPEL (**Reference Resolution System**)
- **Monitoring.**
- Dynamische Auswahl von Datenquellen (per Strategie)
- Granularitätsfestlegung von Auditing und Monitoring.



## 1.4 Definitionen

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

Auf alle in Abschnitt 6.1.3 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 6.1.3 beschriebenen Aktivitäten, erreicht werden.

## 1.5 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 Funktionalität der Software und deren Verwendung. Weiterhin werden die Ziele und Aufgaben, die für die Realisierung der Software bestehen, aufgezeigt. Anschließend werden die vom Kunden genannten Anforderungen durch die Abschnitte 3 und 5 aufgezeigt. Dabei werden durch die nichtfunktionalen Anforderungen qualitative (Robustheit, Portabilität, usw.) und quantitative (Mengengerüst) Anforderungen an die Software spezifiziert. Die Anwendungsfälle 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 in Abschnitt 7 die verwendeten Materialien, Werkzeuge und Technologien, die für die Erstellung der Software benötigt werden, vorgestellt.

## 2 Allgemeine Beschreibung

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

## 2.1 Einbettung

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

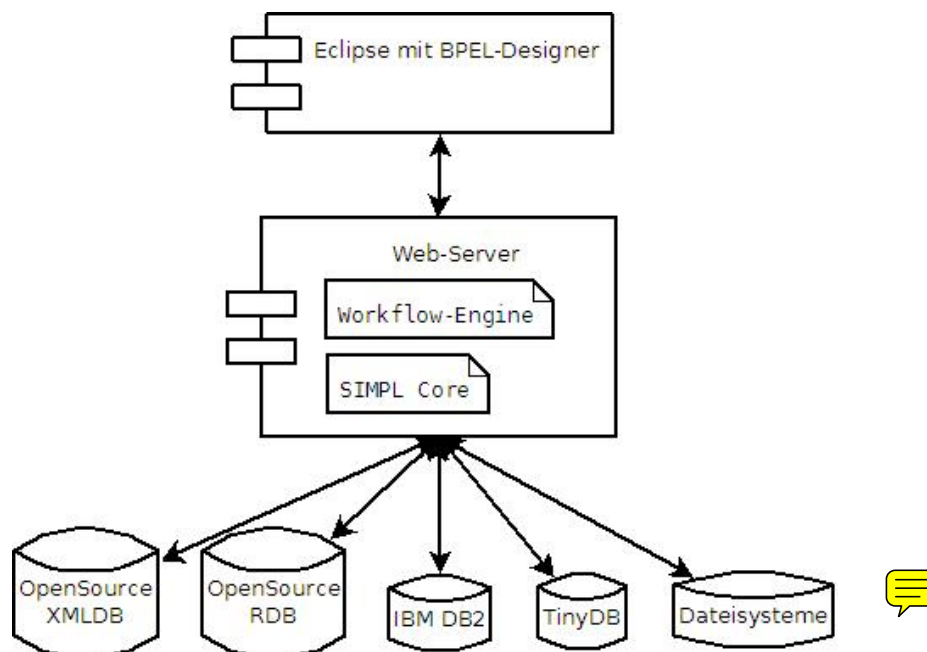


Abbildung 1: Übersicht über die Systemumgebung von SIMPL

## 2.2 Funktionen

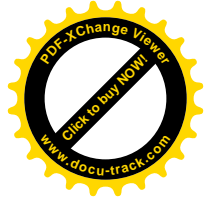
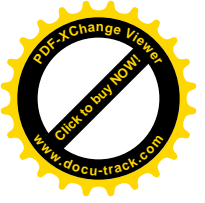
In diesem Abschnitt folgen die wichtigsten Funktionen des Rahmenwerks, die später dessen Kernfunktionalität bilden sollen.

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.

Die vorhandenen BPEL-Aktivitäten des Eclipse BPEL Designer werden dazu um neue Aktivitäten für die Verwaltung von Daten (Data-Management-Aktivitäten) erweitert. Mit deren Hilfe wird die Anbindung von Datenquellen in BPEL-Prozessen vereinfacht. Am Anfang werden die Datenquellen über ihre physikalische Adresse angebunden. In einer späteren Iteration wird eine Datenquellen-Registry bereitgestellt. Diese ermöglicht die Realisierung einer grafischen Auswahlmöglichkeit für Datenquellen im Eclipse BPEL Designer. Eine nähere Beschreibung der Data-Management-Aktivitäten wird in Abschnitt 6.1 gegeben.

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

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



## 2.3 Sprache

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

## 2.4 Distributionsform und Installation

Das Rahmenwerk wird als Teil eines großen Installationspakets ausgeliefert. Dieses Installationspaket besteht aus allen Programmen, die für die Verwendung des Rahmenwerks benötigt werden. Dazu gehört ein Modellierungstool (Eclipse BPEL Designer), ein Web-Server (Apache Tomcat), der eine Workflow-Engine ausführen kann, eine Workflow-Engine (Apache ODE) und natürlich das Rahmenwerk selbst. Mithilfe des Installationspakets ist es möglich viele Einstellungen bereits vorzudefinieren und dem Benutzer die Installation zu erleichtern. Das Installationspaket wird dabei als RAR-Archiv zusammen mit allen wichtigen Dokumenten auf einer CD/DVD ausgeliefert. 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 gesamte 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



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

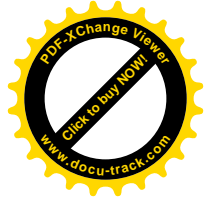
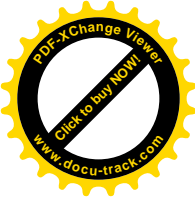
# 3 Nichtfunktionale Anforderungen

In diesem 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



- **Jeder Aktivität kann maximal eine Datenquelle zugewiesen werden.**
- **Für das Auditing** kann nur eine feste aber frei wählbare Datenbank verwendet werden.
- **Das Rahmenwerk kann immer nur von einem Benutzer verwendet werden.**



- Alle laufenden Prozesse einer Workflow-Engine können nur Daten in Höhe des Speichers, der durch das Betriebssystem zur Verfügung gestellt wird, verarbeiten.

### 3.2 Benutzbarkeit

Die Benutzbarkeit soll sich vor allem an Nutzer mit wenig Kenntnissen im Umgang mit Workflows und BPEL richten und die dafür größtmögliche Transparenz liefern, 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 ist hier zu verstehen, dass selbst wenn es zu ungünstigen Bedingungen kommt, SIMPL weiterhin fehlerfrei verwendet werden kann. Dazu sollen Prozesse fehlerfrei ausgeführt werden und entsprechend korrekte Ergebnisse liefern oder das Rahmenwerk soll sicher beendet werden können.



### 3.4 Sicherheit

Da das Rahmenwerk nur lokal ausgeführt wird und alle lokalen Benutzer (Prozess-Modellierer und Workflow-Administrator) die gleichen Rechte besitzen, wird auf Authentifizierungs- und Autorisierungsmaßnahmen verzichtet.



### 3.5 Portabilität

Portabilität bedeutet hier, wenn man bei verschiedenen Softwareumgebungen bzw. verschiedenen Betriebssystemen (z.B. nicht nur bei windows sondern 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

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

### 3.8 Skalierbarkeit

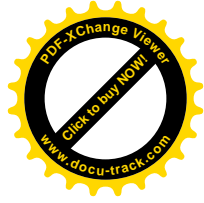
Die Skalierbarkeit des Systems muss eine sehr flexible Infrastruktur erlauben, da die Computersysteme, auf denen SIMPL später ausgeführt wird, in ihrer Leistung sehr weit auseinander gehen können, d.h. vom normalen Desktop-Computer bis zum Supercomputer kann und soll alles möglich sein.



## 4 Akteure

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





## 4.1 Prozess-Modellierer

Ein Prozess-Modellierer besitzt entsprechendes Fachwissen z.B. aus der Biologie, das er bei der Modellierung eines wissenschaftlichen Workflows verwendet. Zur Modellierung der Workflows nutzt er den Eclipse BPEL Designer. Dazu kann er BPEL-Aktivitäten erstellen, bearbeiten und auch löschen. Hat der Prozess-Modellierer den BPEL-Prozess fertig modelliert, kann er diesen im Anschluss auf einer Workflow-Engine 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. Seine Kenntnisse liegen im technischen Bereich, wie z.B. bei der Konfiguration des Rahmenwerks. Er kann beispielsweise über die Admin-Konsole während der Prozesslaufzeit das Auditing an- und abschalten. Ebenso legt er die Datenbank für das Speichern der Auditing-Daten fest.

## 4.3 ODE Workflow-Engine

Die ODE Workflow-Engine ist ein durch Software realisierter Akteur. Sie führt interne Anwendungsfälle aus, die von einem Benutzer durch andere Anwendungsfälle indirekt aufgerufen werden. Zu ihren Aufgaben gehört das Ausführen, Kompensieren und rückgängig Machen von Daten-Management-Aktivitäten.

## 4.4 Eclipse BPEL Designer

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

# 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 der Anwendungsfälle

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

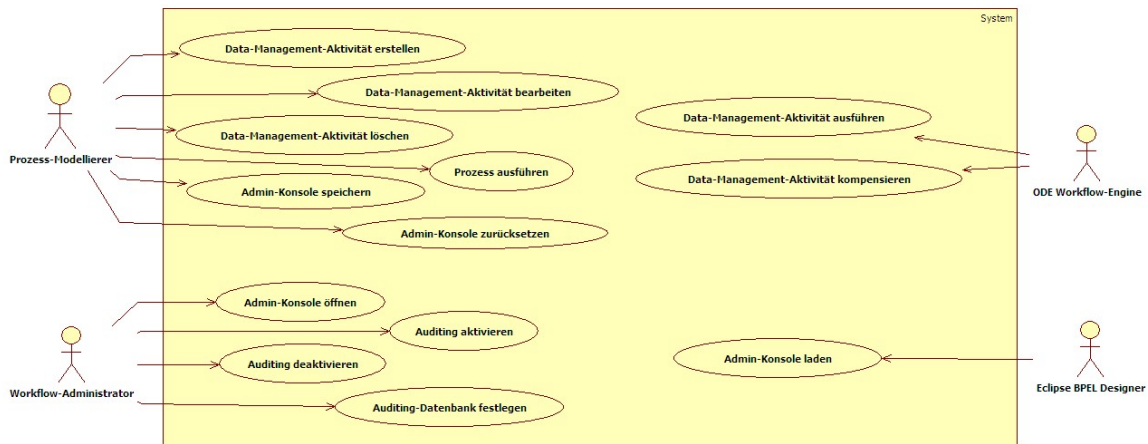


Abbildung 2: Anwendungsfall-Diagramm des gesamten Softwaresystems

## 5.2 Anwendungsfälle der Prozess-Modellierer

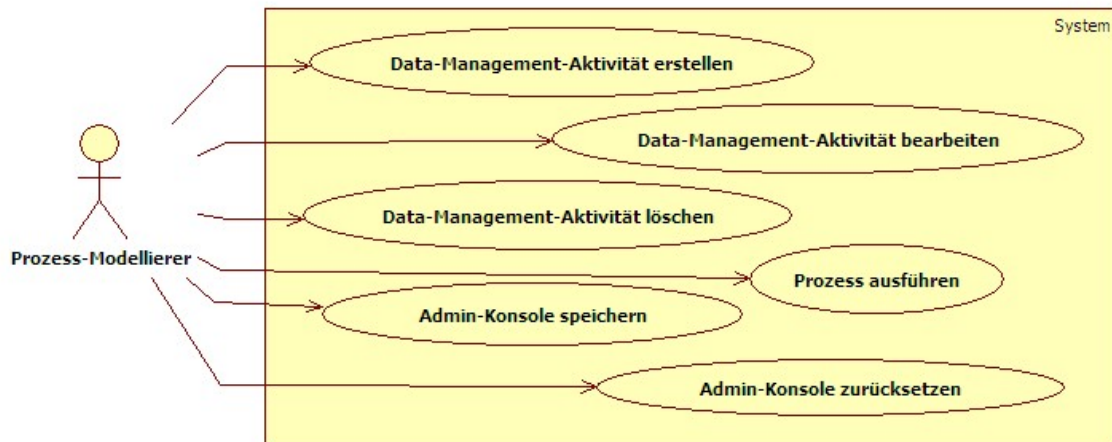


Abbildung 3: Anwendungsfall-Diagramm für den Prozess-Modellierer

### 5.2.1 Data-Management-Aktivität erstellen

Ziel	Erstellung einer <b>neue</b> Daten-Management-Aktivität.
Vorbedingung	Ein vorhandener BPEL-Prozess muss im Eclipse BPEL Designer <b>geöffnet</b> und die BPEL Designer-Palette muss angezeigt werden.
Nachbedingung	Die erstellte Aktivität wurde an der selektierten Position in den Prozess <b>eingefügt und</b> der <b>eingeegebene Name</b> wird angezeigt.
Nachbedingung im Sonderfall	Die erstellte Aktivität wurde an der selektierten Position in den Prozess eingefügt und der <b>vorgeschlagene Name</b> wird angezeigt.
Normalablauf	<ol style="list-style-type: none"> <li>1. Selektion einer Aktivität, durch Auswahl mit der Maus, aus der Palette</li> <li>2. <b>Selektion der Stelle des Prozesses an der die ausgewählte Aktivität eingefügt werden soll</b></li> <li>3. Eingabe eines Aktivitätsnamens</li> </ol>
Sonderfälle	3a. Der angezeigte Namensvorschlag wird vom Benutzer 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 in der "Properties-View" angezeigt.
Nachbedingung im Sonderfall	
Normalablauf	<ol style="list-style-type: none"> <li>1. <b>Selektierung einer Aktivität, durch Auswahl mit der Maus, aus dem Prozess</b></li> <li>2. <b>Eigenschaften der Aktivität werden in der "Properties-View" angezeigt</b></li> <li>3. <b>Änderungen</b> der Eigenschaften</li> </ol>
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 <b>und die zu löschende Aktivität muss ausgewählt</b> sein.
Nachbedingung	Die ausgewählte Aktivität wurde vollständig und korrekt aus dem Prozess gelöscht.
Nachbedingung im Sonderfall	
Normalablauf	<ol style="list-style-type: none"> <li>1. Selektierung einer Aktivität, durch Auswahl mit der Maus, aus dem Prozess</li> <li>2. Löschen der Aktivität</li> </ol>
Sonderfälle	

#### 5.2.4 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 und die Server-View angezeigt werden.
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. Hinzufügen des Prozesses zum ODE-Server in der Eclipse Server-View
Sonderfälle	



#### 5.2.5 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 [Save]
Sonderfälle	1a. Beim Speichern der Werte tritt ein Fehler auf



#### 5.2.6 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 Standardwerte zurückgesetzt.
Nachbedingung im Sonderfall	
Normalablauf	1. Klick auf den Button [Default]
Sonderfälle	

## 5.3 Anwendungsfälle der Workflow-Administratoren

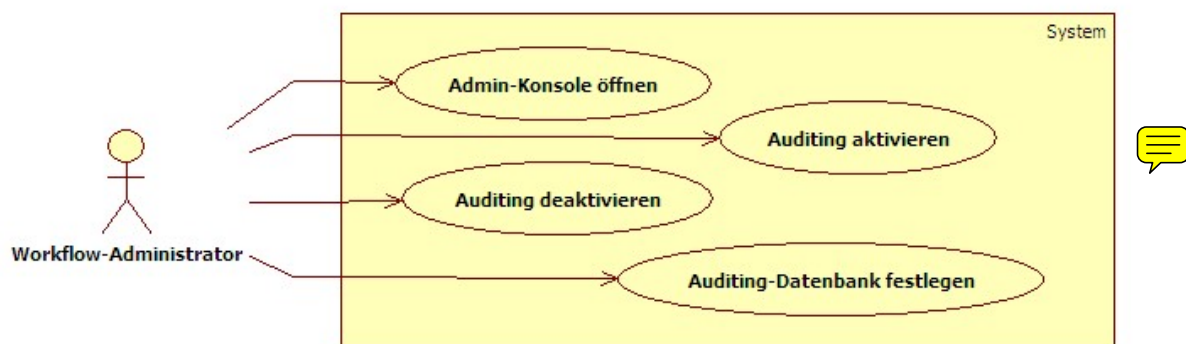


Abbildung 4: Anwendungsfall-Diagramm für den Workflow-Administrator

### 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 Console]
Sonderfälle	

### 5.3.2 Auditing aktivieren

Ziel	Auditing von SIMPL aktivieren.
Vorbedingung	Admin-Konsole muss angezeigt <b>werden und</b> das Auditing ist nicht aktiv.
Nachbedingung	Das Auditing ist aktiviert.
Nachbedingung im Sonderfall	Der Nutzer erhält eine <b>Fehlermeldung die</b> ihn über den entsprechenden Fehler informiert. Das Auditing ist nicht aktiv.
Normalablauf	1. Auditing aktivieren
Sonderfälle	1a. Vom Nutzer wurde keine Auditing Datenbank zum Speichern der Daten festgelegt. 1b. Die angegebene Auditing Datenbank kann nicht verwendet werden

### 5.3.3 Auditing deaktivieren

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

#### 5.3.4 Auditing-Datenbank festlegen

Ziel	Festlegung einer Datenbank für das Auditing.
Vorbedingung	Die Admin-Konsole 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 verwendet werden kann.
Normalablauf	1. Angabe der Datenbank-Adresse (optional Auswahl über Datenbank-Registry)
Sonderfälle	1a. Die angegebene Datenbank kann nicht verwendet werden

#### 5.4 Anwendungsfälle der ODE Workflow-Engine

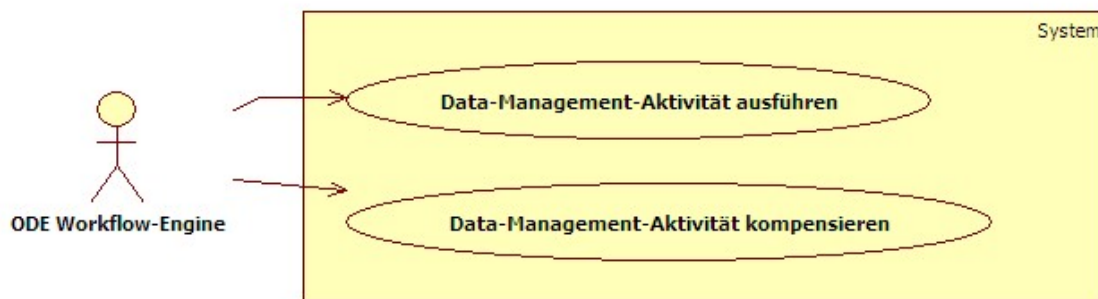



Abbildung 5: Anwendungsfall-Diagramm für die ODE Workflow-Engine

#### 5.4.1 Data-Management-Aktivität ausführen

Ziel	Ausführen einer Data-Management-Aktivität.
Vorbedingung	Es wurde ein Prozess, der eine Data-Management-Aktivität enthält, <b>deployed und</b> es wurde eine Instanz dieses Prozesses erzeugt. 
Nachbedingung	<b>Die Aktivität wurde ausgeführt und das Management der Daten war erfolgreich.</b>
Nachbedingung im Sonderfall	<b>Die Aktivität befindet sich in einem definierten Endzustand.</b>
Normalablauf	<ol style="list-style-type: none"> <li>1. Die Ausführung der Instanz wird gestartet</li> <li>2. Die Aktivität wird als bereit gekennzeichnet</li> <li>3. Die Ausführung der Aktivität wird gestartet</li> <li>4. <b>Ausführung der Data-Management Operationen</b></li> <li>5. Die Ausführung der Aktivität ist beendet</li> <li>6. Die Aktivität wird als abgeschlossen gekennzeichnet</li> </ol>
Sonderfälle	<ol style="list-style-type: none"> <li><b>1a. Die übergeordnete Aktivität meldet einen Fehler</b></li> <li>2a. Die übergeordnete Aktivität meldet einen Fehler</li> <li>2b. Es wird ein Terminate_Activity Event an die Aktivität geschickt</li> <li>2c. Es wird ein Complete_Activity Event an die Aktivität gesendet</li> <li>3a. Die übergeordnete Aktivität meldet einen Fehler</li> <li>3b. Es wird ein Terminate_Activity Event an die Aktivität geschickt</li> <li>3c. Die Aktivität erzeugt einen Fehler</li> <li><b>4a. Die in der Daten-Management-Aktivität angegebene Datenbank oder die angegebenen Datensätze konnten nicht gefunden werden</b></li> <li>4b. Die Aktivität wird als fehlgeschlagen gekennzeichnet</li> <li>5a. Die übergeordnete Aktivität meldet einen Fehler</li> <li>5b. Es wird ein Terminate_Activity Event an die Aktivität geschickt</li> </ol>

#### 5.4.2 Data-Management-Aktivität **kompensieren**

Ziel	Rückgängig machen einer Data-Management-Aktivität, so dass <b>der Zustand</b> vor Ausführung der Aktivität wiederhergestellt wird.
Vorbedingung	<b>Eine Data-Management-Aktivität wurde durchgeführt.</b>
Nachbedingung	Der Zustand vor Ausführung der Data-Management-Aktivität wurde erfolgreich wiederhergestellt.
Nachbedingung im Sonderfall	
Normalablauf	<ol style="list-style-type: none"> <li><b>1. Im Normalablauf einer Data-Management-Aktivität tritt ein Fehler auf</b></li> <li>2. Alle Änderungen werden zurückgesetzt</li> </ol>
Sonderfälle	

## 5.5 Anwendungsfälle des Eclipse BPEL Designers

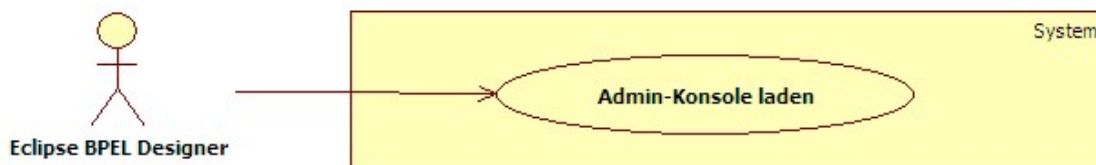



Abbildung 6: Anwendungsfall-Diagramm für den Eclipse BPEL Designer

### 5.5.1 Admin-Konsole laden

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

## 6 Konzepte und Realisierungen

In diesem Abschnitt werden alle Konzepte, die für SIMPL benötigt werden, und deren Realisierung erläutert. Dazu zählt z.B. die Beschreibung und Definition der benötigten Datenmanagement Aktivitäten, die zur Realisierung einer **SQL-inline Unterstützung** für BPEL benötigt werden. Ebenso werden die benötigten **Eclipse BPEL Designer Erweiterungen**, sowie die Realisierung des Datenquellen-Auditing und das dafür zugrunde liegende Event-Modell beschrieben.

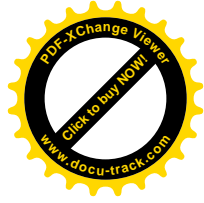
### 6.1 BPEL-SQL

In diesem Abschnitt werden die zu realisierenden Datenmanagement-Patterns zur Umsetzung einer **SQL-inline Unterstützung** vorgestellt und im weiteren Verlauf deren Realisierung für die verschiedenen Datenquellen, d.h. für Dateisysteme, Datenbanken und Sensornetze, erläutert. Aus den in Abschnitt 6.1.2 identifizierten Funktionen, die für die Umsetzung der Datenmanagement-Patterns benötigt werden, werden dann in Abschnitt 6.1.3 neue benötigte BPEL-Aktivitäten abstrahiert, die dann später die entsprechenden Funktionen ausführen werden.

#### 6.1.1 Datenmanagement Patterns

In diesem Abschnitt 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 zwischengespeichert 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.

### 6.1.2 Umsetzung der Datenmanagement Patterns

In diesem Abschnitt wird die Realisierung der acht Datenmanagement Patterns aus Abschnitt 6.1.1 im Zusammenhang mit den jeweiligen Datenquellentypen beschrieben.

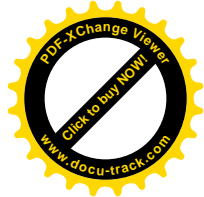


## Dateisysteme

Hier wird die Umsetzung der in Abschnitt 6.1.1 beschriebenen Patterns im Bereich der Dateisysteme durch entsprechende Systemaufrufe (**Betriebssystem**) beschrieben.

- Query Pattern: Realisierung durch eine GET-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 einen Befehl ausgewählt und abgefragt werden können. Eine Möglichkeit wäre z.B. einen Filter-Dialog zur Angabe von Suchparametern über die GUI bereitzustellen, der es ermöglicht Datei-Inhalte zu suchen und zurückzugeben.
- Set IUD Pattern: Realisierung durch eine entsprechende PUT- und eine REMOVE-Methode (RM), die intern java.io Methoden verwenden, um Inhalt in Dateien zu schreiben und aus ihnen zu löschen. Ein Update wird bei Dateien intern durch das Löschen des alten Wertes und anschließendem Einfügen des neuen Wertes ausgeführt. Dafür muss noch ein Konzept erarbeitet werden, wie entsprechende Informationen gezielt über einen Befehl eingefügt und gelöscht werden können.
- Data Setup Pattern: Realisierung durch die Verwendung entsprechender Erzeugungs-Methoden, wie MKDIR und MKFILE zur Erzeugung von Ordnern und Dateien. Diese werden intern wieder durch java.io-Methoden realisiert. Hier sollte es auf jeden Fall möglich sein, Dateien und Ordner in einem Dateisystem erzeugen zu können. Ebenso sollte optional das Löschen auf der gleichen Ebene, d.h. von ganzen Dateien und Ordnern, möglich sein.
- Stored Procedure Pattern: **Findet im Rahmen der Dateisysteme keine Verwendung.**
- **Set Retrieval Pattern:** Realisierung durch die Definition und Bereitstellung einer entsprechenden Aktivität, die Daten kapselt und im Prozessspeicher halten kann, für BPEL. Eine solche Aktivität liefert dann die Möglichkeit, dass in **ihren Ausprägungen** Ergebnisse von Queries (siehe Query Pattern), die über die *Service Data Objects API* (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 Aktivität 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 Aktivität 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 *Data Access Services API* (DAS API) liefern dafür bereits einige Methoden **die verwendet werden können.**

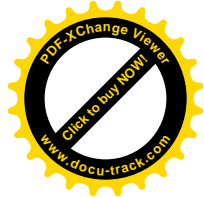




## Datenbanken

Hier wird die Umsetzung der in Abschnitt 6.1.1 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 SQL-CALL (Aufruf über JDBC API möglich).
  - Beispiele:



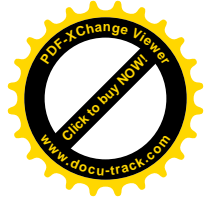
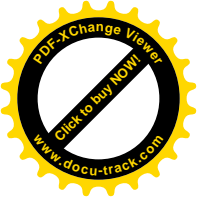
\* CALL parts\_on\_hand (?,?,?) (? = Parameter der Prozedur)

- Set Retrieval Pattern: Realisierung durch die Definition und Bereitstellung einer entsprechenden Aktivität, die eine mengenorientierte Datenstruktur kapseln kann, für BPEL. Eine solche Aktivität liefert dann die Möglichkeit, dass in Ausprägungen dieser Aktivität, 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 Aktivität 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 mengenorientierten Daten 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 6.1.1 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 geschrieben 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 zwischenspeichern 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, uint32, int8, int16, int32} und **field1**, **field2**, usw. sind Spaltennamen, die wie der Tabellennamen 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.

### 6.1.3 Resultierende BPEL-Aktivitäten

In diesem Abschnitt werden die durch Abschnitt 6.1.2 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 ~~momentan~~ mindestens die drei Variablen *type*, *dqAddress* und *statement* vom Typ String besitzen. Die Variable *type* dient zur Angabe des Datenquellentyps für den die Aktivität ausgeführt wird, also ob es sich um ein Dateisystem, eine Datenbank oder ein Sensornetz handelt. Diese Auswahl ist wichtig um intern den richtigen SQL-Dialekt bzw. Befehlssatz für die entsprechende Datenquelle auszuwählen. Die Variable *dqAddress*

dient zur Angabe der Datenquellenadresse und die Variable *statement* zur Haltung des entsprechenden Systemaufrufs bzw. SQL- oder XQuery-Befehls **der ausgeführt werden soll**. Diese drei Variablen besitzt jede der definierten Aktivitäten. Darum werden diese nachfolgend nicht mehr angegeben und nur falls benötigt weitere aktivitätsspezifische Variablen beschrieben. Weiterhin werden die verschiedenen Ausprägungen der einzelnen Aktivitäten im Hinblick auf die zugrundeliegende Datenquelle aufgezeigt, so dass am Ende ein vollständiger Überblick aller definierten Aktivitäten und ihrer Ausprägungen vorliegt. Generell gibt es durch die verschiedenen Datenquellen nur wenige strukturelle Unterschiede in den Aktivitäten. Das liegt vor allem daran, dass auf datenquellenspezifische Eigenschaften bereits in der graphischen Oberfläche, also dem Eclipse BPEL Designer, eingegangen werden soll und diese so durch entsprechende Dialoge intern immer auf dieselbe Struktur abgebildet werden können. **Eben dazu sollen alle Befehle auch über entsprechende graphische Elemente angegeben werden können, indem sie einfach "zusammengeklickt" werden können (siehe Abbildung 11).** Dadurch soll eine einfachere Handhabung realisiert werden, damit der Benutzer schnellstmöglich, ohne detaillierte Kenntnisse der Anfragesprache, alle zur Verfügung gestellten Daten- und Datenquellenbefehle für alle unterstützten Datenquellen ausführen kann.



### Query Activity

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

### Insert Activity

Diese Aktivität ermöglicht es Daten in ~~jeder~~ Datenquelle einzufügen. Da dies für Sensornetze nicht relevant ist, wird diese Aktivität für das sensornetzinterne Einfügen von Sensornetzdaten in Buffer-Tabellen genutzt (wie auch in Abschnitt 6.1.2 beschrieben).

### Update Activity

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

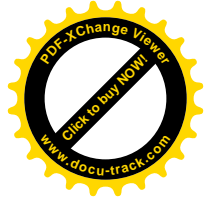
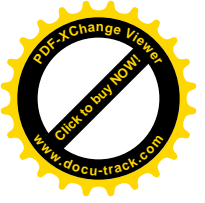
### Delete Activity

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
  - MKDIR FOLDER
  - MKFILE 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

- \* RMDIR FOLDER
- \* RM FILE

– Datenbanken

- \* DROP TABLE
- \* DROP SCHEMA
- \* DROP VIEW

– Sensornetze

- \* DROP ALL





## 6.2 Eclipse BPEL Designer

Abbildung 7 zeigt den erweiterten Eclipse BPEL Designer. In der Palette befinden sich die SIMPL Data-Management-Aktivitäten, die wie bereits vorhandene Aktivitäten zur Modellierung von Prozessen verwendet werden können. Für SIMPL wird ein Menü bereitgestellt, über das alle wichtigen Einstellungen und Informationen des SIMPL Rahmenwerks **erreicht werden können**.

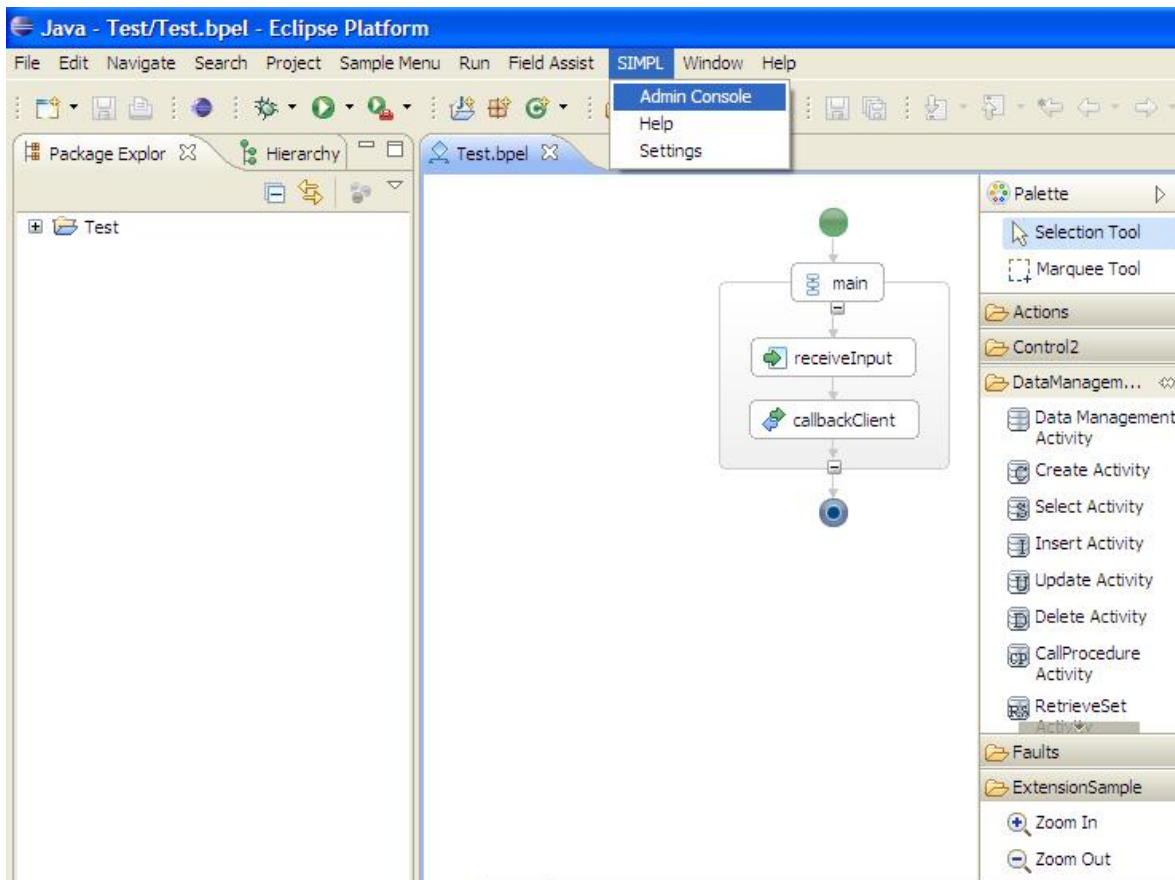


Abbildung 7: SIMPL Menü und Eclipse BPEL Designer mit einigen Data-Management-Aktivitäten

### 6.2.1 Admin-Konsole

Die Admin-Konsole kann über das SIMPL Menü geöffnet werden und bietet die Möglichkeit, Einstellungen **innerhalb** des Rahmenwerks vorzunehmen. Dazu gehört beispielsweise die Angabe von globalen Einstellungen und die Verwaltung des Auditings (siehe Abbildung 8).

Folgende Schaltflächen stehen durchgängig zur Verfügung:

- [Default]: Laden der Standard-Einstellungen
- [Save]: Speichern aller ~~aktuellen Einstellungen~~
- [Cancel]: Schließen der Admin-Konsole und Verwerfen aller Änderungen



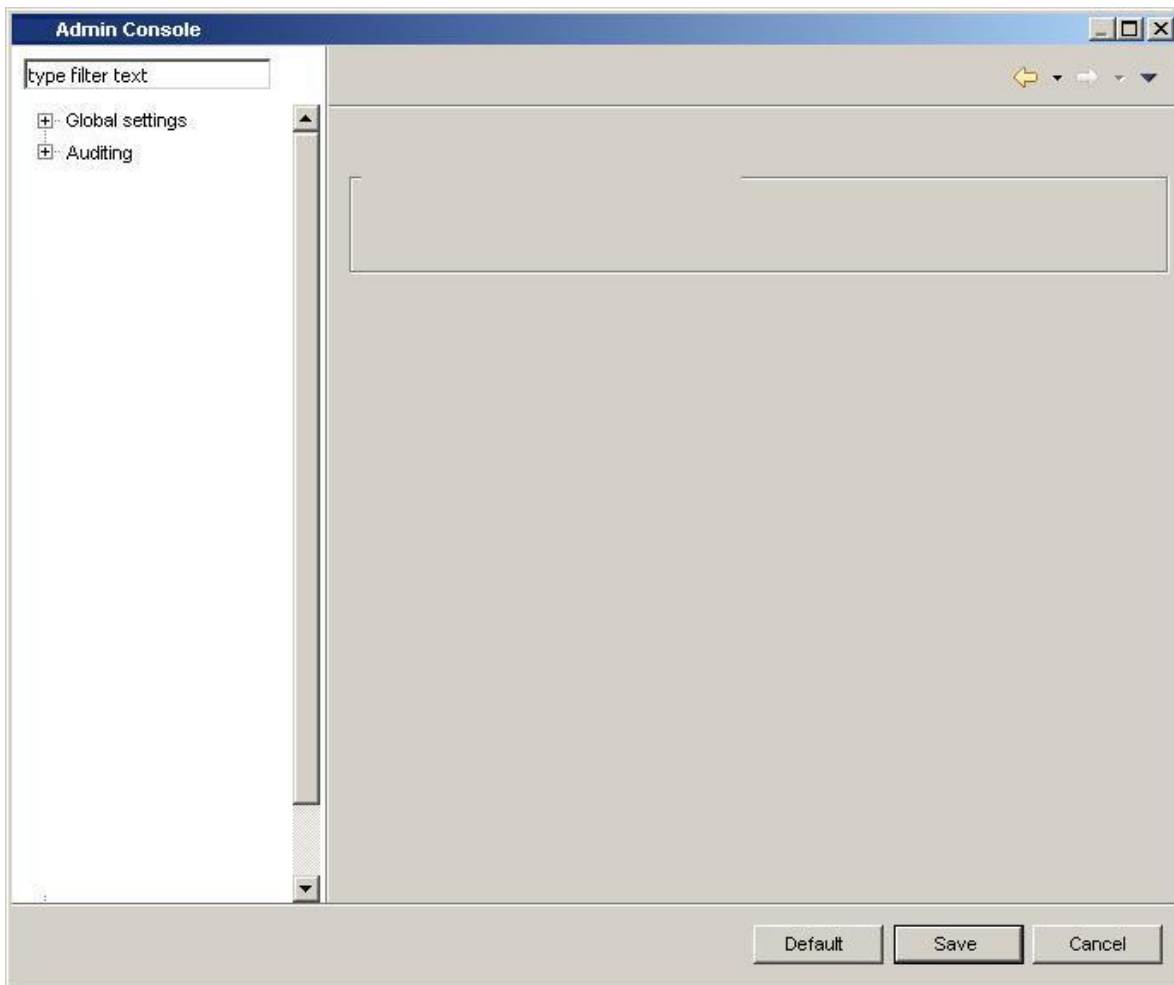


Abbildung 8: Admin-Konsole

**Globale Eigenschaften** In Abbildung 9 wird der Unterpunkt “Global Settings” der Admin-Konsole gezeigt. Hier können z.B. Standardwerte für die Authentifizierung bei Datenquellen angegeben werden.

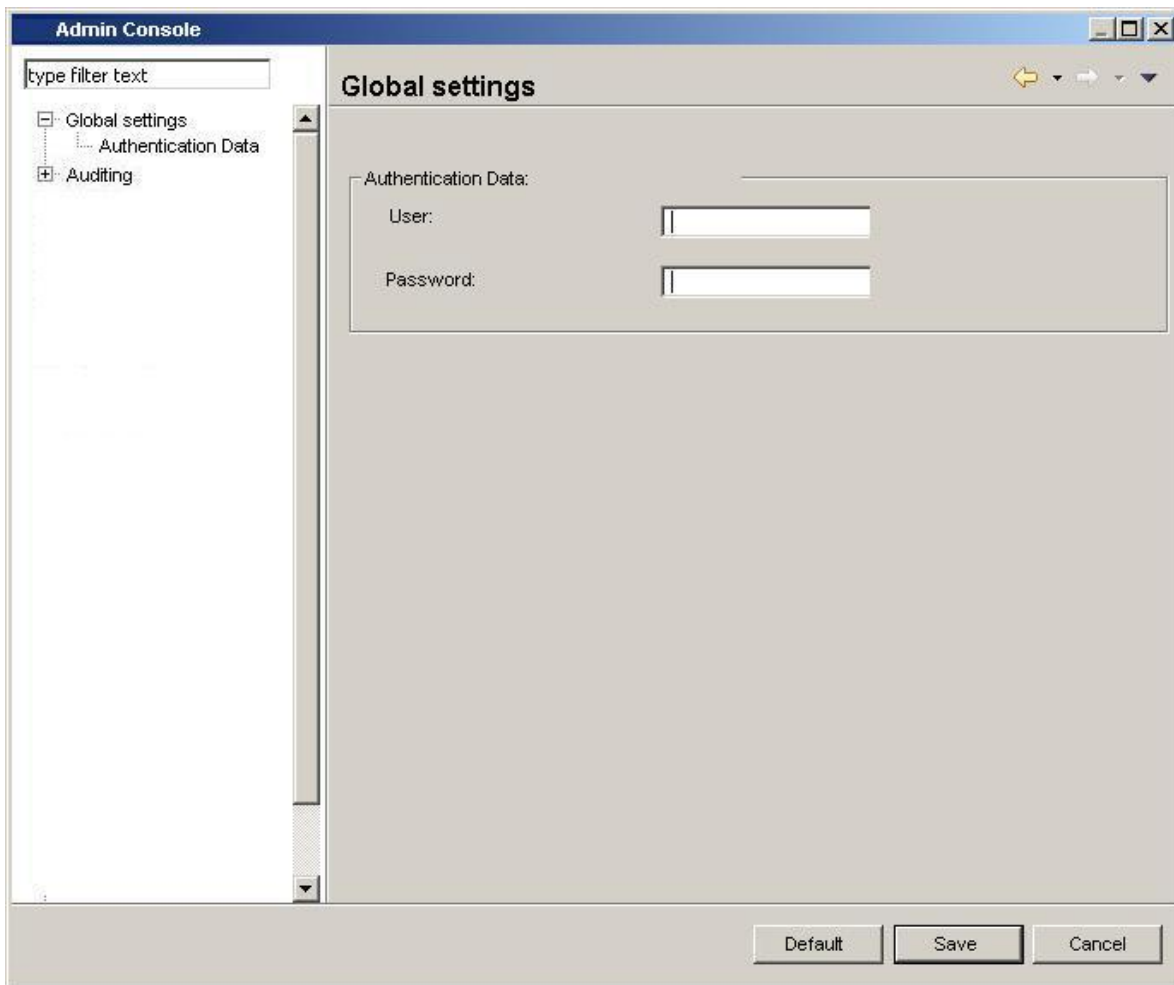


Abbildung 9: Dialog der globalen Eigenschaften

**Auditing** In Abbildung 10 wird der Unterpunkt “Auditing” der Admin-Konsole gezeigt. Hier kann das Auditing an- und abgeschaltet werden und eine Datenbank für das Auditing angegeben werden.

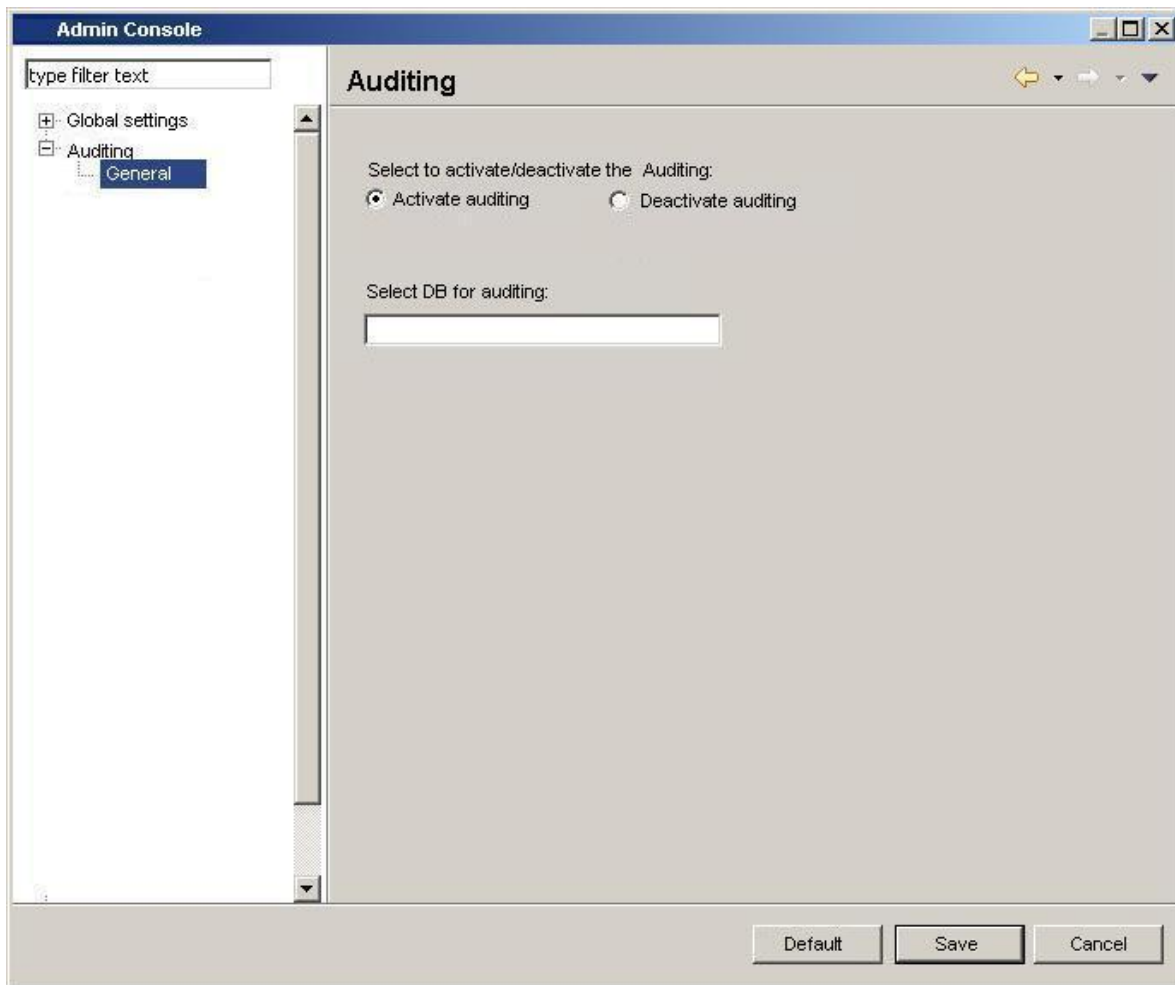


Abbildung 10: Dialog für die Auditing Einstellungen

### 6.2.2 Data-Management-Aktivitäten

In Abbildung 11 wird die “PropertyView” einer Query-Aktivität gezeigt. Hier kann die im Prozess ausgewählte Aktivität **parametrisiert** werden. Dazu wird die Art der Datenquelle ausgewählt, ein Befehl über entsprechende grafische Elemente erstellt und die physikalische Adresse der Datenquelle angegeben.



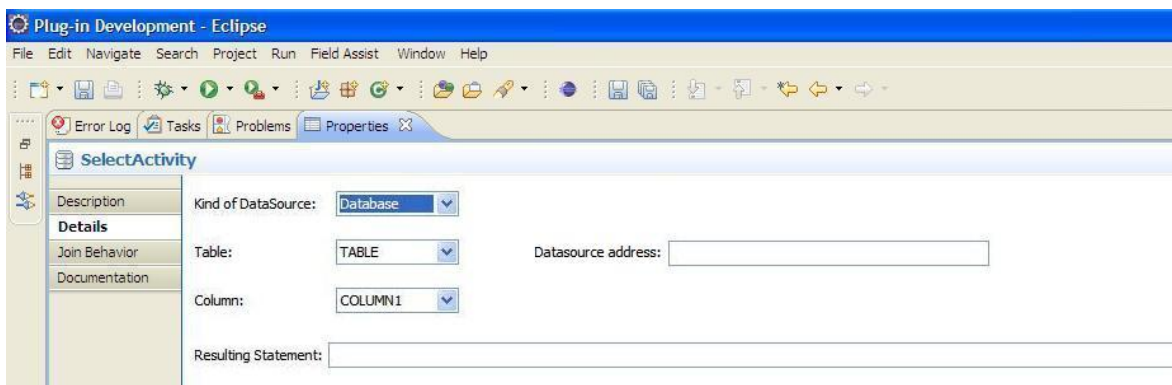


Abbildung 11: Eigenschaftsfenster einer Data-Management-Aktivität am Beispiel einer Query Activity

## 6.3 Auditing

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

### 6.3.1 Momentane Situation - Monitoring von ODE

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

#### Management API:

Die Management API der ODE Engine macht es möglich zahlreiche Informationen abzurufen. So ist es beispielsweise möglich, zu überprüfen **welche Prozesse gerade eingesetzt werden** und welche Prozessinstanzen ausgeführt werden oder beendet sind. Dies ist besonders für das Monitoring wichtig, da es hierdurch möglich ist spezifische Informationen zu einem bestimmten Prozess oder einer bestimmten Instanz abzurufen. Dies kann zum Beispiel das Erstellungsdatum sein, eine Liste der Events **die für diesen Prozess oder diese Instanz generiert wurden** und vieles mehr.

#### Events:

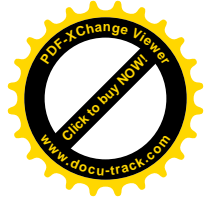
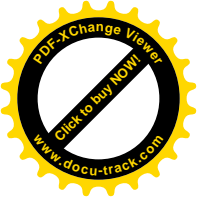
**Events sind Ereignisse die auftreten wenn bestimmte Aktionen innerhalb der ODE Engine durchgeführt werden.** Zum Beispiel das Aktivieren eines Prozesses oder das Erzeugen einer neuen Prozessinstanz. Diese Events machen es möglich, zu verfolgen **was innerhalb der ODE Engine passiert** und produzieren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

#### Event Listener:

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

### 6.3.2 Auditing von SIMPL

Im BPEL Designer ist der Zugriff auf das Auditing unter dem entsprechenden Menüpunkt möglich. Hier kommt man nun zum entsprechenden **Interface in** dem sich verschiedene Einstellungen für das



Auditingtätigen lassen (siehe Abbildung 10). Es ist möglich das Auditing zu aktivieren oder zu deaktivieren und es ist außerdem möglich die Datenbank für das Speichern der Auditing Daten festzulegen. Das Auditing ist standardmäßig aktiviert, kann allerdings über die Adminkonsole deaktiviert und auch erneut aktiviert werden.

Für das Auditing von SIMPL werden die von ODE erzeugten Daten direkt an den SIMPL Core übergeben und weiterverarbeitet, anstatt diese wie bisher in der virtuellen Datenbank von ODE abzulegen. Dies wird realisiert durch die Implementierung eines eigenen Data Access Objects, was entsprechende Funktionalitäten anbietet. Diese Informationen werden anschließend in einer zuvor in der Adminkonsole festgelegten Auditing-Datenbank abgelegt.



## 7 Materialien, Werkzeuge und Technologien

In diesem Abschnitt folgt ein Überblick über die im Projekt genutzten Materialien und die verwendeten Werkzeuge und Technologien.

### 7.1 Materialien

Als Material für SIMPL verwenden wir die Programmiersprache Java in der aktuellsten Version (Java SE 6).

### 7.2 Technologien

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

#### ApacheODE <http://ode.apache.org>

Apache ODE ist eine BPEL-Workflow-Engine. Es wird sowohl WS-BPEL 2.0 als auch BPEL4WS 1.1 unterstützt und die Ausführung von Prozessen in SOA ist möglich. Zudem ist das "Hot Deployment" 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 für die Modellierung von BPEL-Prozessen. Er verfügt über eine leicht verständliche GUI, eine Plausibilitätsprüfung für BPEL-Prozesse und außerdem ist die Schritt für Schritt Ausführung von Prozessen möglich.

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

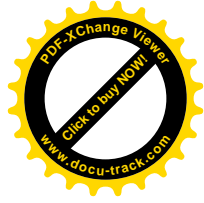
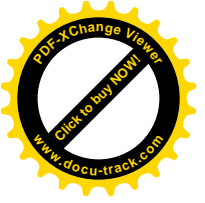
Die IBM-DB2 ist ein Hybriddatenserver mit dem sowohl die Verwaltung von XML-Daten als auch von relationalen Daten möglich ist.

#### Apache Tuscany <http://tuscany.apache.org>

Apache Tuscany ist ein Software-Projekt, welches es zum Ziel hat die Erstellung von Service Oriented Architecture (SOA) Anwendungen zu vereinfachen, indem eine verständliche Infrastruktur und grundlegende Komponenten zur Verfügung gestellt werden.

#### Webservices

Webservices sind Software-Anwendungen, die zur direkten Interaktion mit anderen Software-Agenten dienen.



**Axis2** <http://ws.apache.org/axis2>

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

**Service Data Object (SDO)**

SDO stellt eine einheitliche API zur Verfügung, mit der die Handhabung verschiedener heterogener Daten und Datenquellen vereinfacht wird.

**Data Access Services (DAS)**

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

## 7.3 Werkzeuge

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

**Subversion** <http://svnbook.red-bean.com>

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

**Maven** <http://svnbook.red-bean.com>

Maven ist ein Projekt-Management-Tool. Es wird zur automatischen Erstellung von Software-Projekten genutzt. Mit Maven ist es möglich viele Schritte, die die Entwickler normalerweise von Hand erledigen müssen zu automatisieren.

**Lyx** [www.lyx.org](http://www.lyx.org)

Lyx ist ein Textverarbeitungsprogramm, mit dem es möglich ist auf einfachste Art und Weise L<sup>A</sup>T<sub>E</sub>X-Dokumente zu erstellen.

**Hudson** <https://hudson.dev.java.net/>

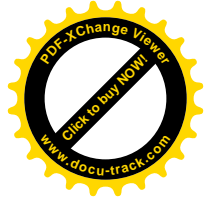
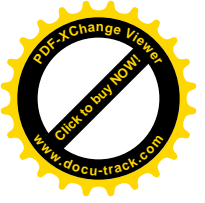
Hudson ist ein webbasiertes System zur kontinuierlichen Integration von Softwareprojekten.

### 7.3.1 Entwicklungsumgebung

Die IDE in der das Projekt SIMPL entwickelt wird, ist Eclipse in der Version 3.5 (Galileo).

## 8 Änderungsgeschichte

- **Version 0.1**, 11. September 2009: Erstellung des Dokuments.
- **Version 1.0**, 28. September 2009: Überarbeitung 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/](http://www.w3.org/TR/ws-addr-core/), 2006.

