Studienprojekt SIMPL

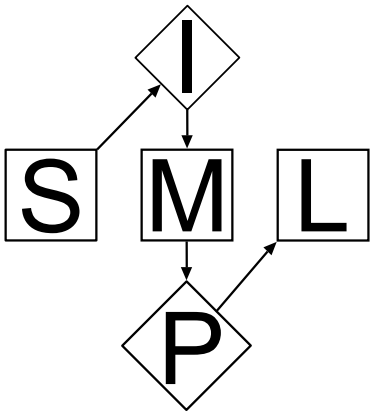
Spezifikation

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 Evolution des Dokuments

1.4 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 6.1.3 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, realisiert 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 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.

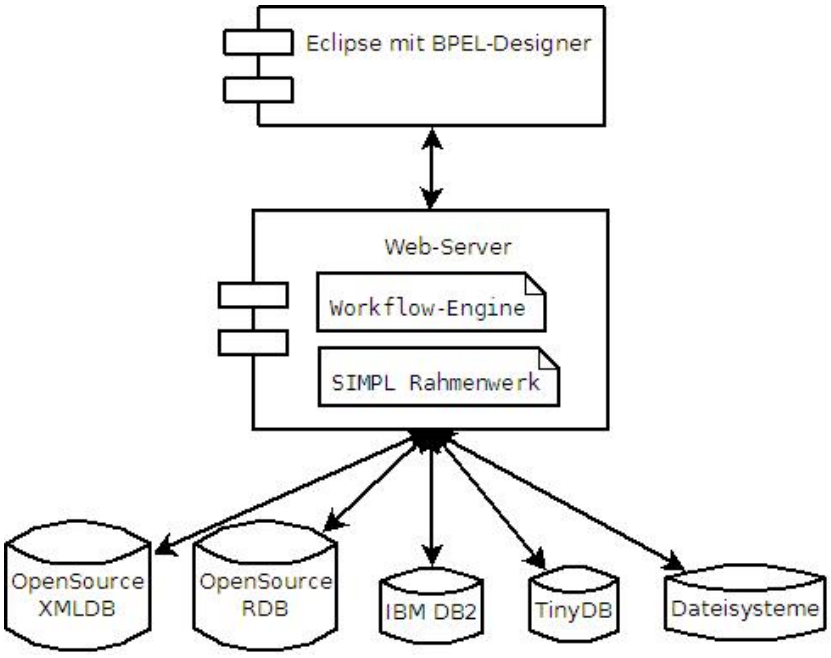


Abbildung 1: Übersicht über die Systemumgebung von SIMPL

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.

Die vorhandenen BPEL-Aktivitäten des Eclipse BPEL Designer werden dazu um neue Daten-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 Daten-Management-Aktivitäten wird in Abschnitt 6.1 gegeben.

Alle Ereignisse, die wärend 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. 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 gehöhrt 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 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.

4.4 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 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.

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.

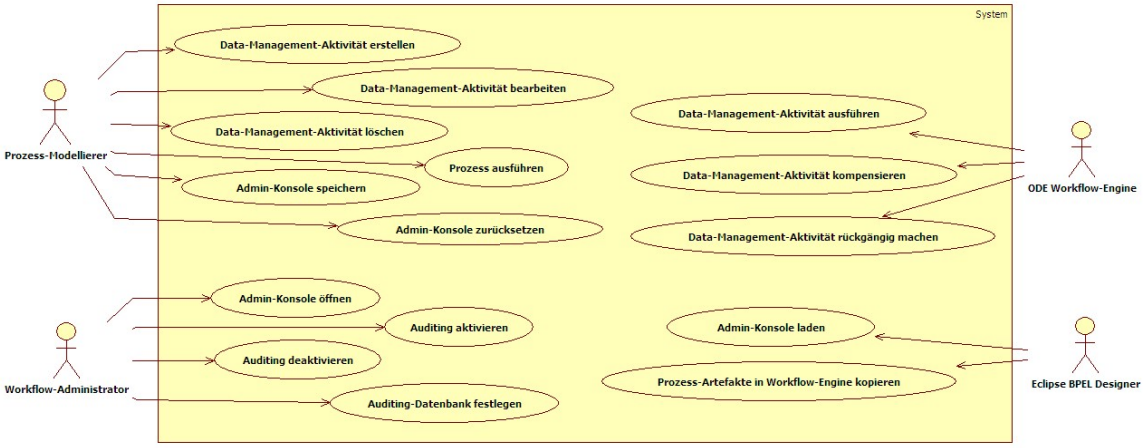


Abbildung 2: Anwendungsfall-Diagramm des gesamten Softwaresystems

5.2 Anwendungsfälle der Prozess-Modellierer

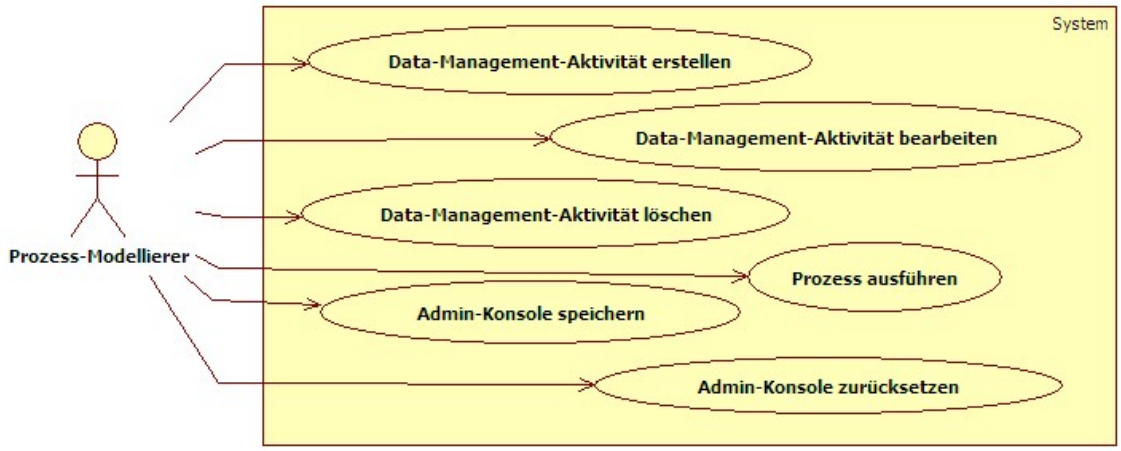


Abbildung 3: Anwendungsfall-Diagramm für die Prozess-Modelierer

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 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
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 [Speichern]
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 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

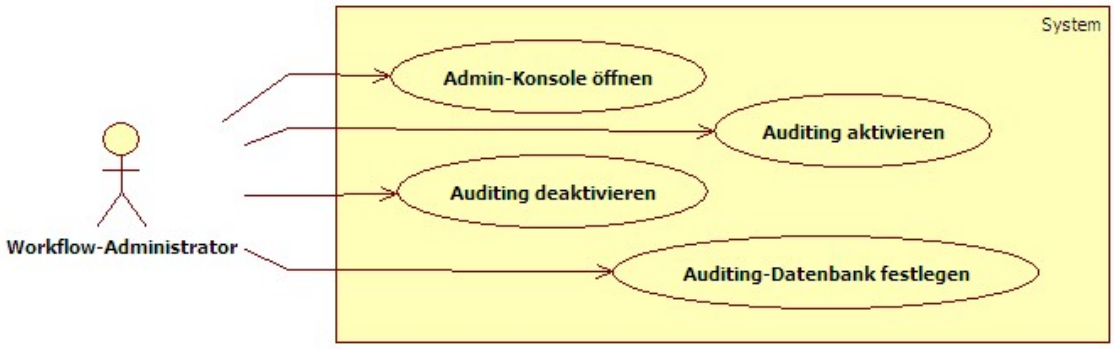


Abbildung 4: Anwendungsfall-Diagramm für die Prozess-Modelierer

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 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	Der Nutzer erhält eine Fehlermeldung die ihn informiert, dass keine Auditing Datenbank festgelegt wurde und wird gebeten eine auszuwählen. Das Auditing ist nicht aktiv.
Normalablauf	1. Auditing aktivieren
Sonderfälle	1a. Vom Nutzer wurde keine Auditing Datenbank zum speichern der Daten festgelegt.

5.3.3 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	

5.3.4 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

5.4 Anwendungsfälle der ODE Workflow-Engine

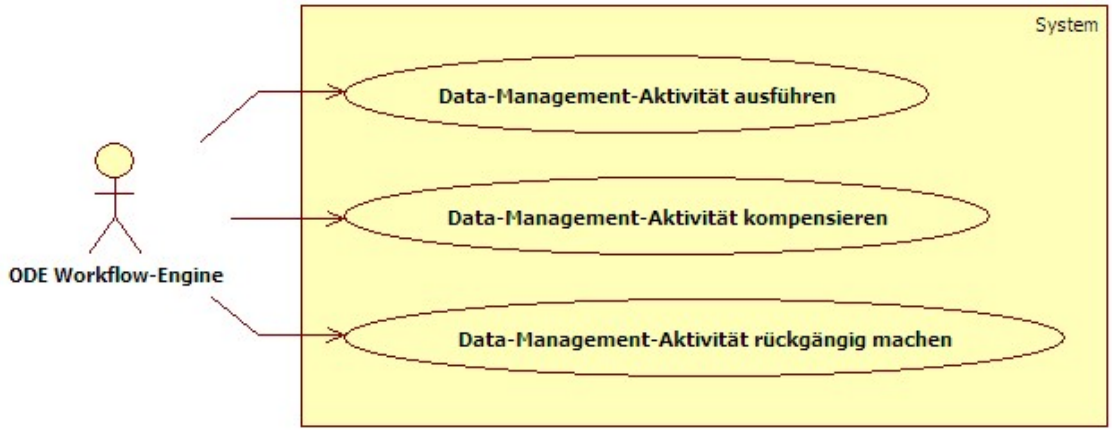


Abbildung 5: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.4.1 Data-Management-Aktivität ausführen

Ziel	Ausführen einer Data-Management-Aktivität.
Vorbedingung	Es wurde ein Prozess, der eine Data-Management-Aktivität enthält deplyoed und es wurde eine Instanz dieses Prozesses erzeugt.
Nachbedingung	Die Aktivität wurde ausgeführ und das Management der Daten war erfolgreich.
Nachbedingung im Sonderfall	
Normalablauf	1. Die Ausführung der Instanz wird gestartet 2. Die Aktivität wird als bereit gekennzeichnet 3. Die Ausführung der Aktivität wird gestartet 4. Ausführung der Data-Management Operationen 5. Die Ausführung der Aktivität ist beendet 6. Die Aktivität wird als vollständig gekennzeichnet
Sonderfälle	

5.4.2 Data-Management-Aktivität kompensieren

Ziel	
Vorbedingung	
Nachbedingung	
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.4.3 Data-Management-Aktivität rückgängig machen (rollback)

Ziel	Rückgängig machen einer Data-Management-Aktivität so das der Zustand vor Ausführung der Aktivität wiederhergestellt wird.
Vorbedingung	Eine Data-Management-Aktivität wurde durchgeführt.
Nachbedingung	Der Zustand vor Ausführung der Data-Management-Aktivität wurde erfolgreich wiederhergestellt.
Nachbedingung im Sonderfall	
Normalablauf	
Sonderfälle	

5.5 Anwendungsfälle des Eclipse BPEL Designers

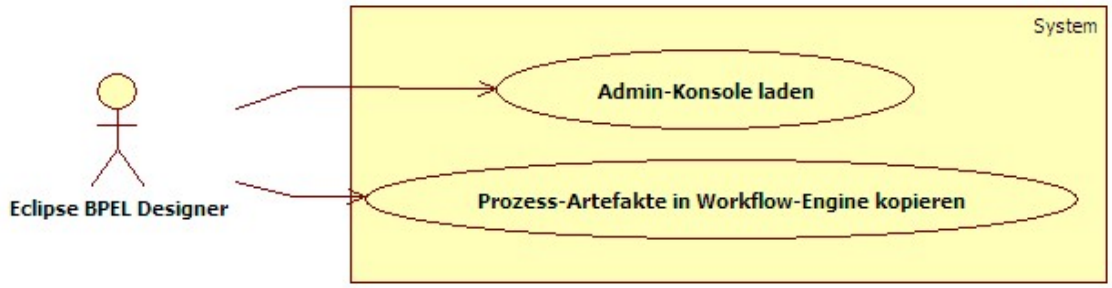


Abbildung 6: Anwendungsfall-Diagramm für die Prozess-Modelierer

5.5.1 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

5.5.2 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

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 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 6.1.2 identifizierten Methoden und Funktionen werden dann in Abschnitt 6.1.3 neue benötigte BPEL-Aktivitäten abstrahiert, die dann später die entpsrechenden Methoden und Funktionen realisieren werden.

6.1.1 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.

6.1.2 Umsetzung der Datenmanagement Patterns

In diesem Kapitel wird die Realisierung der acht Datenmanagement Patterns aus Abschnitt 6.1.1 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 6.1.1 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 6.1.1 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 [(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 6.1.1 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 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.

6.1.3 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 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 6.1.2 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
  - 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

6.2 Eclipse BPEL Designer

Im folgender Abbildung ist der kommplete UI der Rahmenwerk von SIMPL. Auf der rechte Seite befindet sich Menu “DataManagement” mit aller Aktivitäten für die Erzeugung von Data-Aktivität, Select, update etc. zuständig sind.

Beim Auswählen bzw. Drag-and-Drop einer dieser graphische Darstellungen der Aktivitäten, dann erscheint entsprechend einer neue Aktivität in dem Bpel-prozess an der ausgewählte Position im Prozess.

Ausserdem können wir den Popup-Menu vom SIMPL im Eclipse Menuleiste sehen, dass alle Navigationspunkte für die Anwendungsaktivitäten wie “Admin console” oder “Settings” etc. beinhaltet.

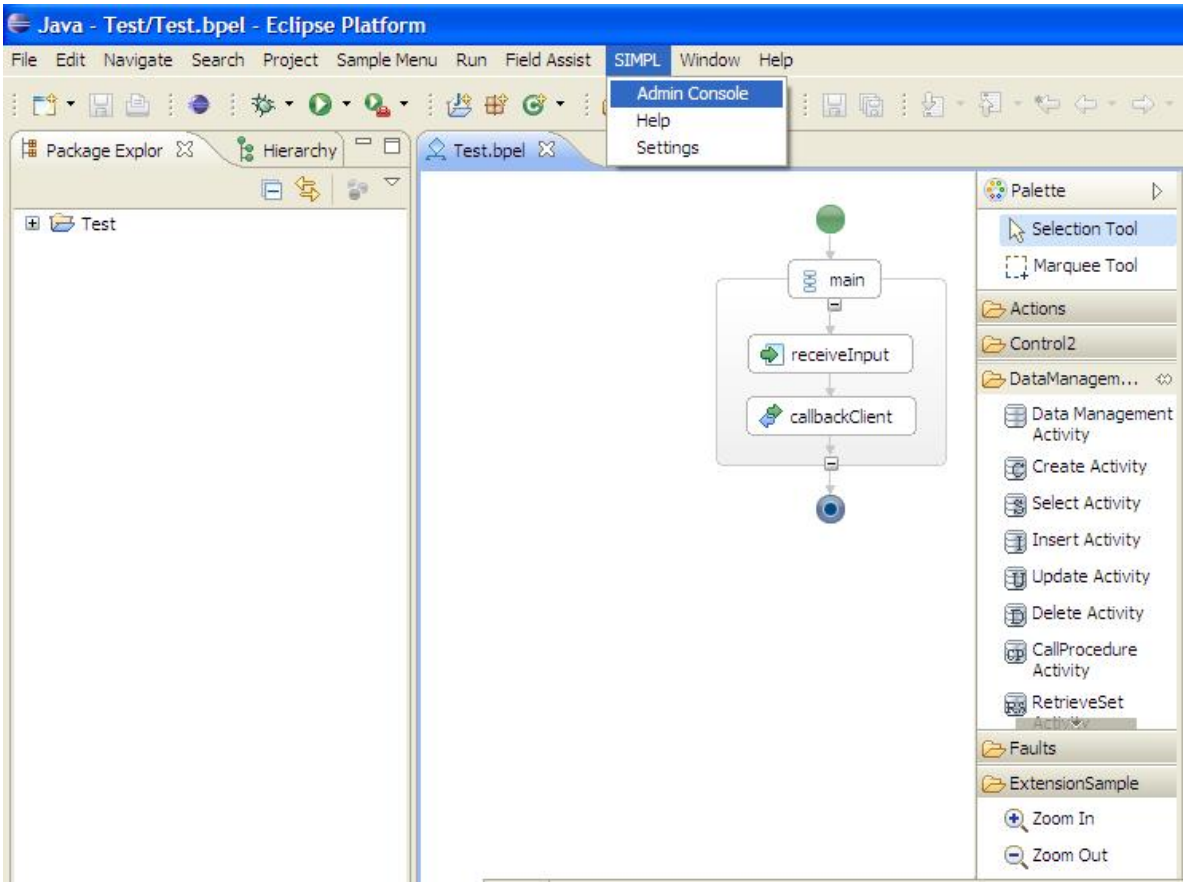


Abbildung 7: SIMPL Menü und Eclipse BPEL Designer mit einigen Data-Management-Aktivitäten

6.2.1 Admin-Konsole

Beim anklicken auf dem Untermenupunkt “Admin Console” öffnet sich der Adminkonsole Dialog.

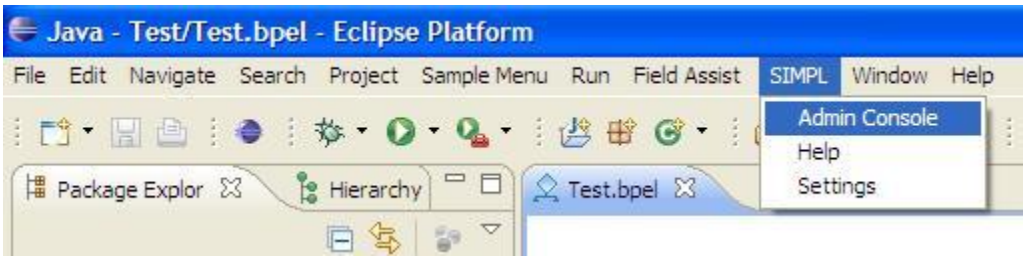


Abbildung 8: Admin-Konsole Untermenupunkt

Hier ist der Adminkonsole Dialog zu sehen. Mithilfe die Unterbaumpunkte können verschiedene Einstellungen angebracht werden wie “Global settings”, die Strategien oder Datenquellen etc.

Auf der untere rechte Seite dieser Dialog befinden sich die zwei Schaltflechen “Save” und “Cancel”.

- “Save” Schaltfläche: damit werden die angebrachte Änderungen gespeichert und der Dialog wird geschlossen.
- “Cancel” Schaltfläche: damit wird einfach der Dialog geschlossen ohne Berücksichtigung der Änderungen, die innerhalb der Konsole angebracht wurden.

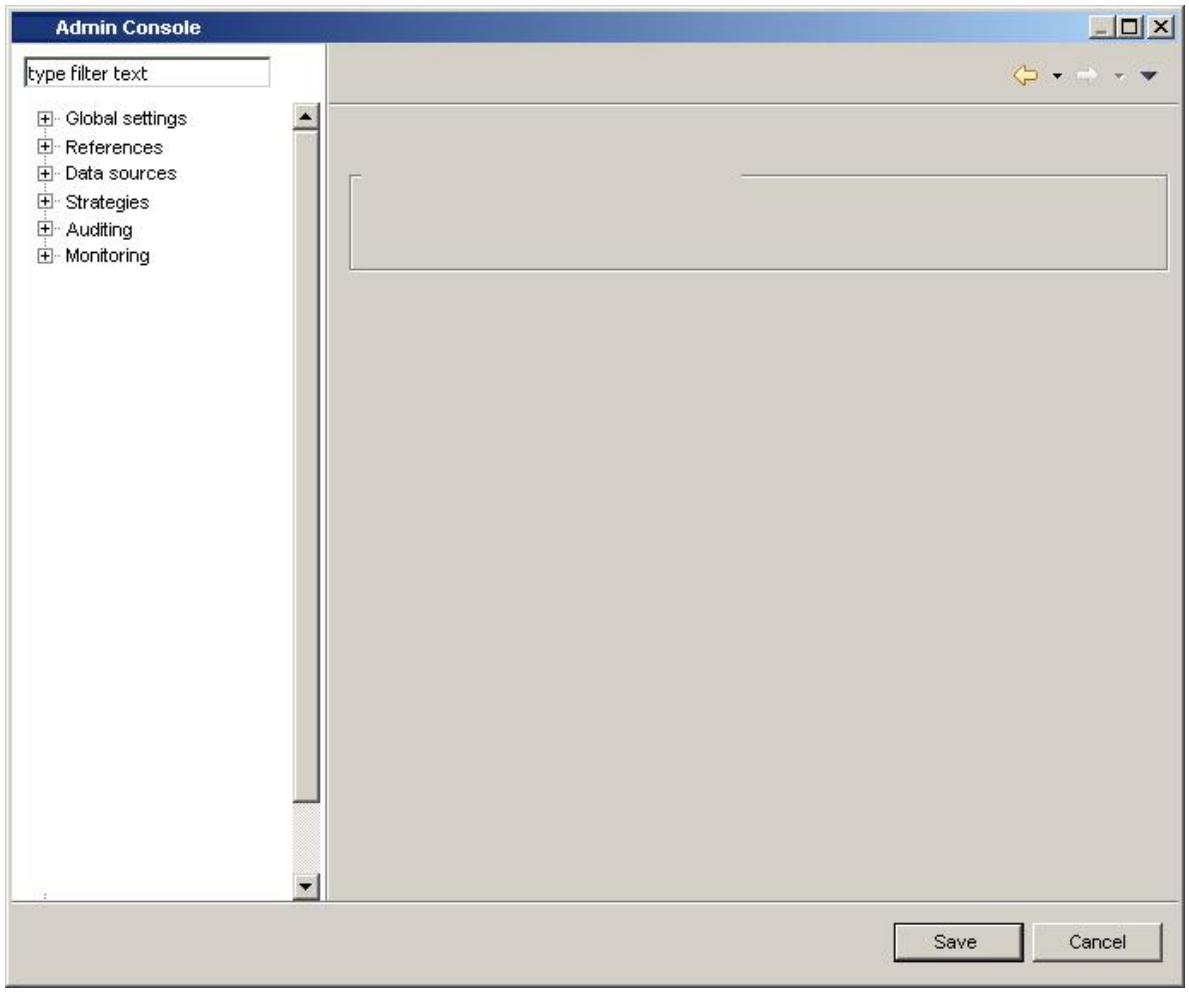


Abbildung 9: Admin-Konsole

Globale Eigenschaften In folgeneden Dialog kann der Benutzer über den Unterbaumpunkt "Global settings-Authentication Data" die Zugangsdaten angeben.

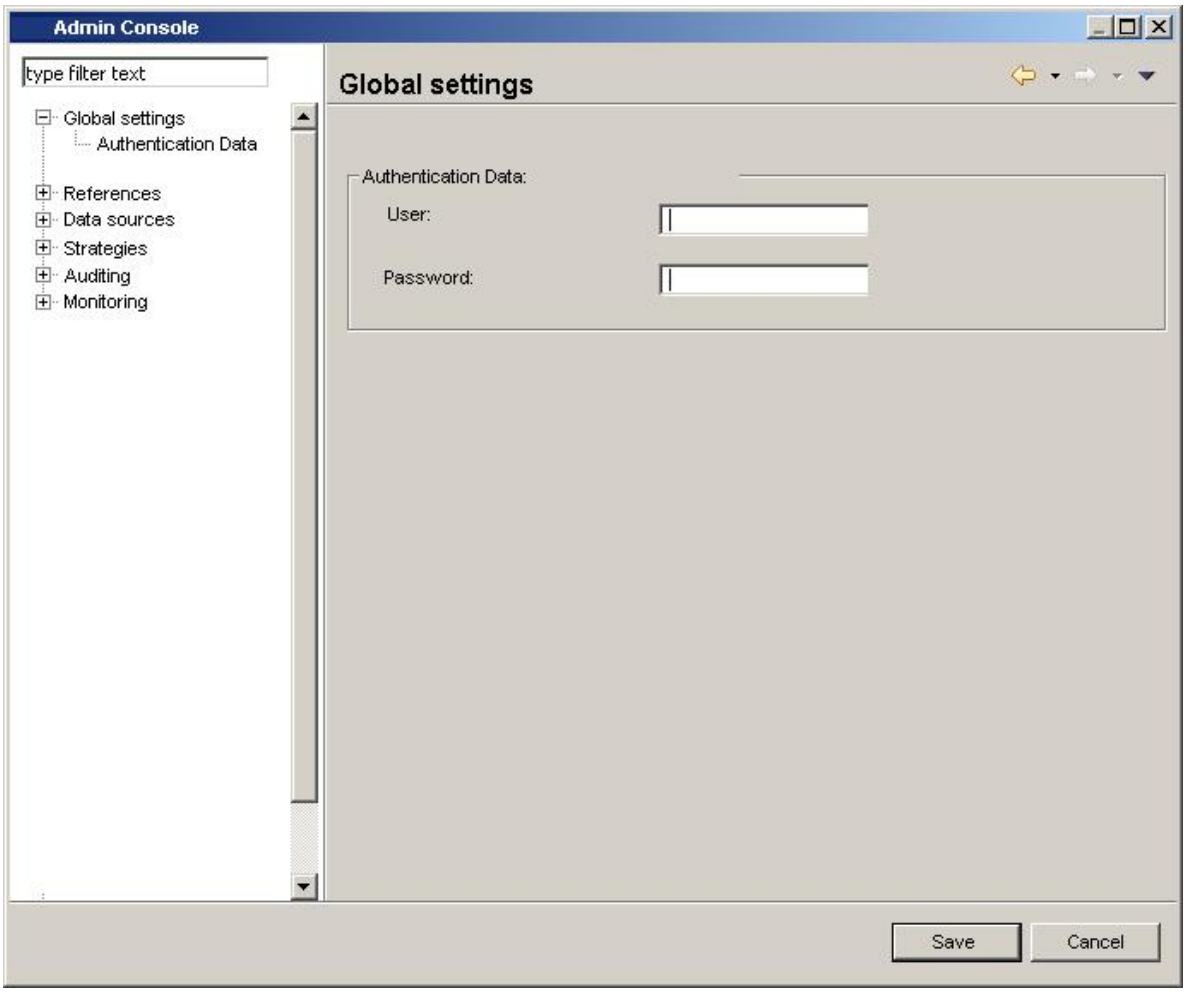


Abbildung 10: Globale Eigenschaften Dialog

6.2.2 Data-Management-Aktivitäten

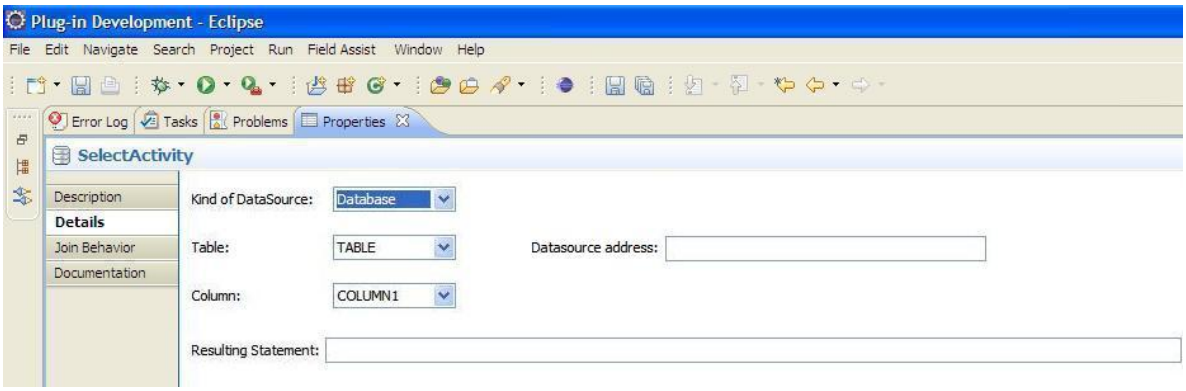


Abbildung 11: Eigenschaftsfenster einer Data-Management-Aktivität am Beispiel einer SELECT Activity

6.3 Monitoring und Auditing

In diesem Abschnitt wird eine Beschreibung des geplannten Auditing und Monitoring von SIMPL gegeben. Zunächst wird auf die bestehenden Möglichkeiten des Monitorings von ODE eingegangen. Hier folgt eine Erläuterung aller relevanten Begriffe die für das Monitoring wichtig sind, sowie einige Übersichten zur Veranschaulichung. Anschließend folgt eine kurze Beschreibung darüber wie das Monitoring und Auditing in ODE aussehen soll, dies wird durch Event Modelle veranschaulicht.

6.3.1 Momentane Situation - Monitoring von ODE

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

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 detailierte 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 durchführen.

Management API:

Die Management API der ODE Engine macht es möglich zahlreiche Informationen abzurufen. So ist es beispielsweies 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.

6.3.2 Monitoring und Auditing von SIMPL

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 und es ist außerdem möglich die Datenbank für das Speichern der Auditing Daten festzulegen.

Das Monitoring und Auditing von SIMPL wird sich nur auf die Prozesse und Instanzen der Datenzugriffsaktivitäten beschränken. Weiterhin ist zu erwähnen, dass das Auditing standardmäßig aktiviert ist, allerdings über die Adminkonsole deaktiviert und auch erneut aktiviert werden kann.

6.3.3 Auditing

Wenn das Audutiting aktiviert wird, im Standardfall wenn SIMPL gestartet wird, wird eine Liste der Prozesse und Instanzen die momentan in der ODE Engine existieren abgerufen. Diese Listen werden in Form von XML-Datein übergeben. Nach Verarbeitung dieser Listen ist es nun weiterhin möglich detailierte Informationen und die aufgetretenen Events zu den einzelnen Prozessen und Instanzen abzurufen. Diese Informationen und Events werden ebenfalls in Form von XML-Datein übergeben. Diese

Informationen werden anschließend als xml-Datein in einer entsprechenden Datenbank gespeichert. Es ist bisher nicht vorgesehen die Auditing Daten in irgendeiner Form weiterzuverarbeiten um diese als Strings in einer relationalen Datenbanke zu speichern.

Der zuvor beschriebene Vorgang des Abrufens und Verarbeitens der Daten, wird, so lange das Auditing aktiv ist, regelmäßig innerhalb einer bestimmten Zeit wiederholt.

6.3.4 Monitoring

Das Monitoring von SIMPL wird als ein Echtzeitmonitoring realisiert. Die Informationsgewinnung des Monitoring würd über direkte Rückmeldungen auf Events die innerhalb der ODE Engine auftreten realisiert. Wenn ein Event auftritt, so zum Beispiel das Aktivieren eines Prozesses, oder das generieren einer neuen Prozessinstanz, dann erhält das Monitoring eine direkte Rückmeldung darüber und die Visualisierung ist möglich.

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.

6.3.5 Event-Modelle

Anmerkung: Optionale Bestandteile des Monitoring und Auditing, die vorraussichtlich nicht im Liefe-rumfang enthalten sind, wurden im Event Modell markiert.

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

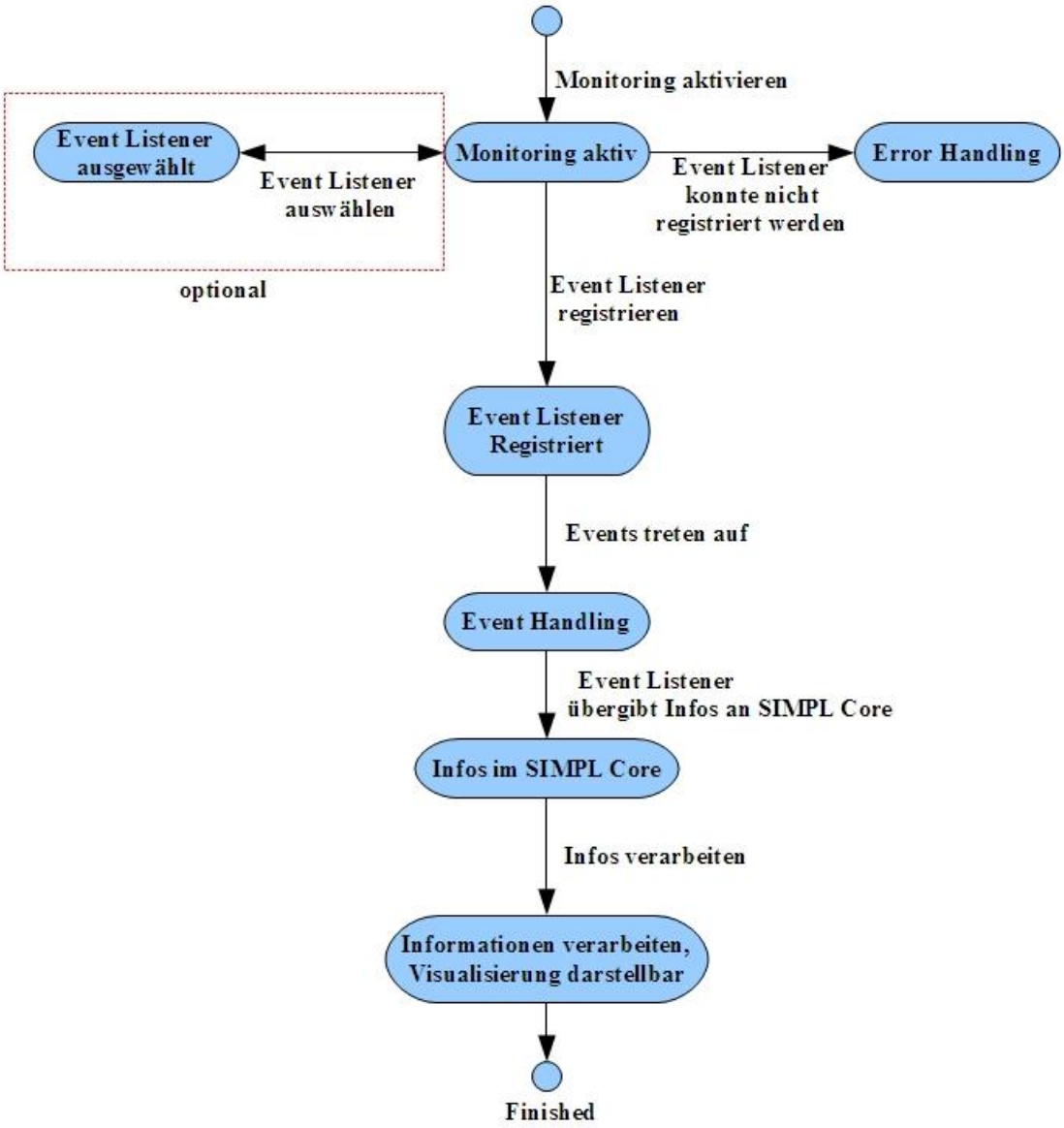


Abbildung 12: Event Modell des Monitoring

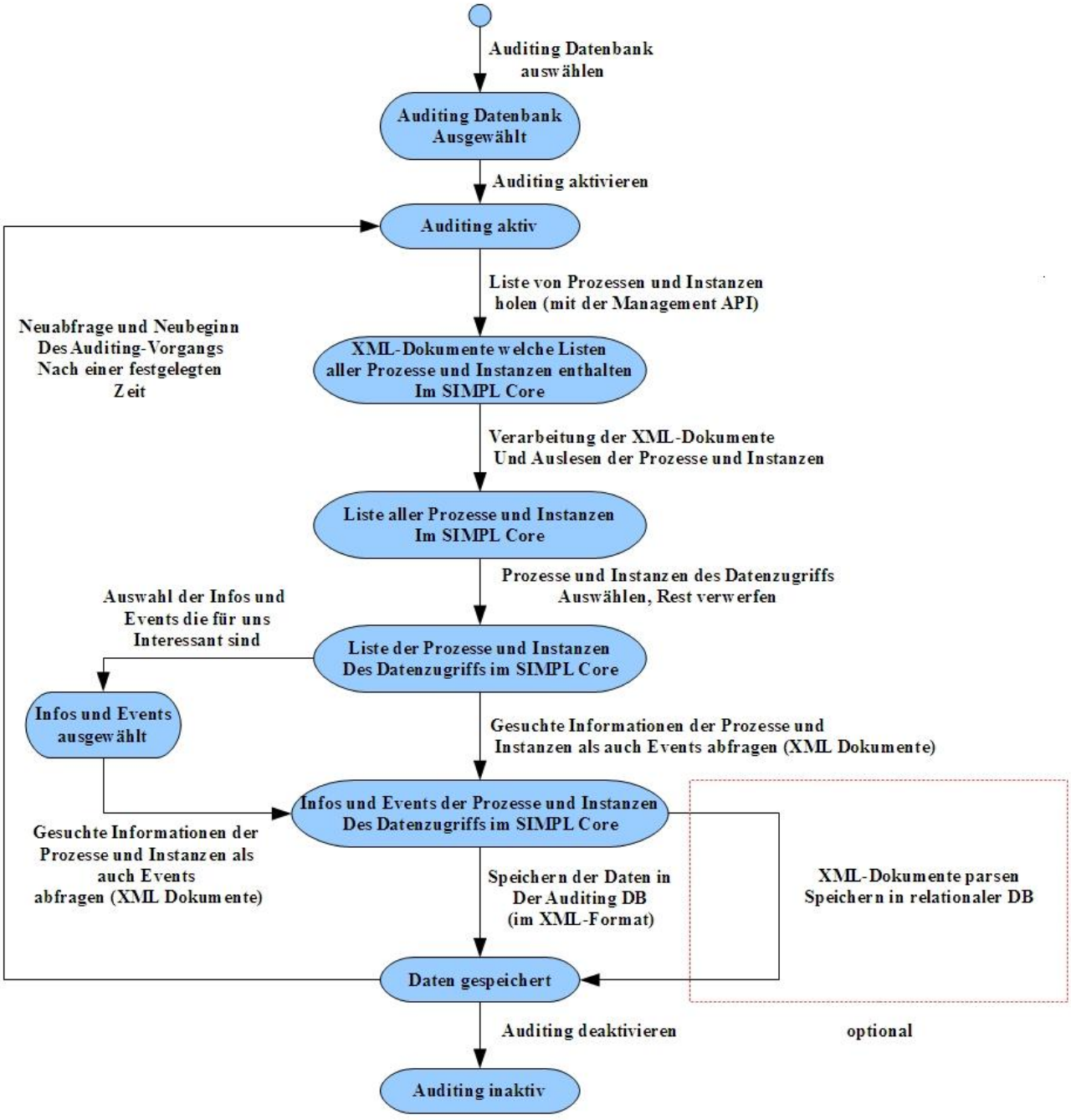


Abbildung 13: Event Modell des Monitoring

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.