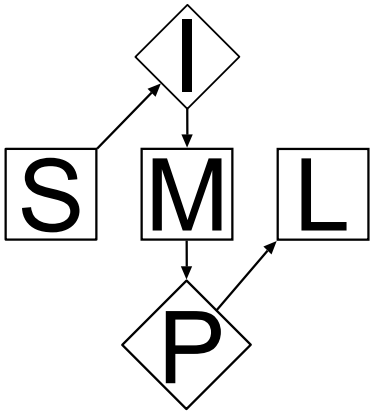
Studienprojekt SIMPL

Projektplan

Version 1.3

28. April 2010



Inhaltsverzeichnis

Änderungsgeschichte

Version	Datum	Autor	Änderungen
0.1	22. Juli 2009	hahnml, zoabisfs	Erstellung des Dokuments.
0.2	02. August 2009	hahnml, zoabisfs, huettiwg	Aktualisierung des Dokuments.
0.3	26. September 2009	huettiwg	Aktualisierung des Terminplans und Überarbeitung des Dokuments.
1.0	31. Oktober 2009	huettiwg	Fertigstellung des aktuellen Dokuments.
1.1	20.12.2009	huettiwg	Anpassen der Zeitplanung.
1.11	21.12.2009	huettiwg	Hinzufügen eines Alternativen Zeitplans und alternativer Meilensteine.
1.2	04.03.2010	huettiwg	Aktualisierung des Zeitplans
1.3	28.04.2010	huettiwg	Aktualisierung des Zeitplans

1 Einleitung

In diesem Kapitel wird ein kurzer Überblick des Projekts und der im Projektverlauf zu erstellenden Produkte vermittelt.

1.1 Wichtige Anmerkung zu den Erweiterungen.

Die Zeitplanung und die Meilensteine wurden aktualisiert und zusätzlich alternative Projektverläufe hinzugefügt. Diese diehnen als Verhandlungsbasis mit den Kunden.

1.2 Projektübersicht

Das Projekt läuft von Mai 2009 bis Mai 2010. In diesem Zeitraum soll das Entwicklungsteam ein erweiterbares, generisches Rahmenwerk 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 für unsere Zwecke erweitert oder angepasst. Dabei wird auch die Sprache BPEL für unsere Zwecke erweitert. Für eine möglichst hohe Flexibilität soll ein dynamischer Ansatz gewählt werden, so dass während der Laufzeit des Systems die Datenquellen festgelegt werden können. Weiterhin sollte auch die Möglichkeit bestehen die Datenquellen statisch anbinden zu können. Der Kunde besteht darauf, dass eine BPEL-Engine sowie ein Modellierungstool um diese gewünschten Funktionen erweitert bzw. angepasst werden. Die BPEL-Prozesse sollen mit dem entsprechenden Modellierungstool modelliert und mit der BPEL-Engine ausgeführt werden können.

Das Projekt besteht aus den folgenden Hauptmeilensteinen/Phasen: Analyse, Einarbeitung, Spezifikation, Entwurf, Implementierung und Test. Eine detailliertere Beschreibung der einzelnen Phasen findet sich in Abschnitt 5.1.

Am Ende des Projekts soll das Rahmenwerk als Eclipse Plug-In und mit allen erforderlichen Tools als Installationspaket zur Verfügung stehen. Darüber hinaus sollen alle Erweiterungen und Anpassungen, die während des Projekts an bestehenden Konzepten durchgeführt wurden, in einer einheitlichen Dokumentation zusammengefasst und ein Handbuch sowie eine Demo für die Arbeit mit dem Rahmenwerk erstellt werden (siehe Tabelle 1).

Das Zeit-Budget des Projekts liegt bei 480h pro Teammitglied bzw. 3360h für das gesamte Team, wie sich das Zeit-Budget auf die einzelnen Projektabschnitte aufteilt zeigt Tabelle 6.

Die Kunden des Projekts sind Katharina Görlach, Peter Reimann und Mirko Sonntag im folgenden ihre Kontaktdaten:

Dipl.-Inf. Katharina Görlach

Institut für Architektur von Anwendungssystemen

Universitätsstraße 38, 70569 Stuttgart (Zimmer 1.328)

Telefon: +49 (0)711 7816-333

Fax: +49 (0)711 7816-472

E-Mail: katharina.goerlach(@)iaas.uni-stuttgart.de

Dipl.-Inf. Peter Reimann

Institut für Parallele und Verteilte Systeme

Universitätsstraße 38, 70569 Stuttgart (Zimmer 2.467)

Telefon: +49 (0)711 7816-445

Fax: +49 (0)711 7816-424

E-Mail: Peter.Reimann(@)ipvs.uni-stuttgart.de

Dipl.-Inf. Mirko Sonntag

Institut für Architektur von Anwendungssystemen

Universitätsstraße 38, 70569 Stuttgart (Zimmer 1.332)

Telefon: +49 (0)711 7816-202

Fax: +49 (0)711 7816-472

E-Mail: mirko.sonntag(@)iaas.uni-stuttgart.de

1.3 Auszuliefernde Produkte

In Tabelle 1 sind alle auszuliefernden Produkte mit den jeweiligen voraussichtlichen Lieferterminen aufgeführt.

Produkt	**Liefertermin**
Spezifikation	01.03.2010
Handbuch	04.06.10
Dokumentation der Erweiterungen	04.06.10
Testfälle	04.06.10
Testprotokolle	04.06.10
Installationspaket, das alle benötigten Software-Komponenten beinhaltet	04.06.10
Eclipse Plug-In	04.06.10
Quellcode	04.06.10

Tabelle 1: Auszuliefernde Produkte mit Terminen

2 Projektorganisation

In diesem Kapitel werden alle organisatorischen Punkte des Projekts aufgezeigt und beschrieben. Dazu gehören das verwendete Prozessmodell, die Organisation und die verschiedenen Verantwortlichkeiten und Rollen des Projekts.

2.1 Prozessmodell

Als Prozessmodell für das Projekt, wird ein inkrementelles Modell zum Einsatz kommen. Das heißt, dass wir nach Beendigung des Grobentwurfs mehrere Entwicklungsiterationen durchlaufen werden, welche aus Planung, Implementierung und Test des entsprechenden Softwareabschnitts bestehen. Dieses Vorgehen bietet sich für das Projekt an, da es sich hierbei um die Entwicklung einer Kernkomponente handelt, welche nach und nach durch verschiedene Module und Plug-Ins erweitert werden kann. Für das Entwicklerteam, aber auch den Kunden, hat dies den Vorteil, dass am Ende jeder Iteration ein lauffähiger Prototyp zur Verfügung steht, an dem der Kunde mögliche Änderungswünsche frühzeitig äußern kann.

2.2 Organisationsstruktur

Der Projektleiter bildet die oberste Instanz des Projekts (siehe Abbildung 1). Er hat als einziger die volle Weisungsbefugnis und Kontrolle über alle Entscheidungen innerhalb des Projekts. Alle anderen Teammitglieder haben nur innerhalb ihres Aufgabengebietes Weisungsbefugnis, da sie am Besten einschätzen können, wie etwas in ihrem Aufgabengebiet umgesetzt werden sollte. Der Projektleiter kann jedoch trotzdem diese Entscheidungen stürzen und so auch demokratische Entscheidungen des Teams gegen den Verantwortlichen des Aufgabengebiets durchsetzen, sollte dies notwendig sein.

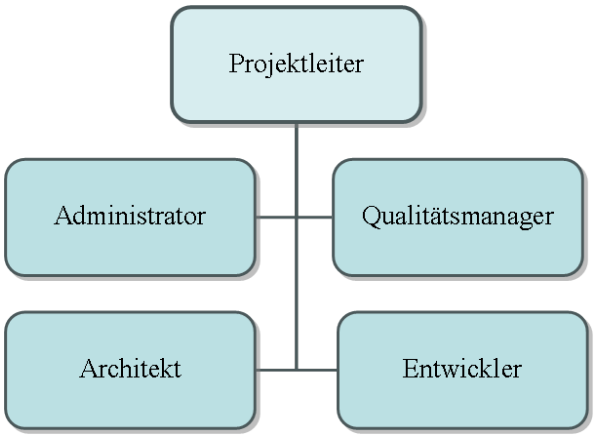


Abbildung 1: Organisationsstruktur als Organigramm

Die Kommunikation innerhalb des Projekts wird durch die Verwendung einer Mailing-Liste sowie wöchentlichen Treffen des gesamten Teams gesichert. Dadurch ist das ganze Team immer über die aktuellen Arbeiten und Fortschritte jedes Teammitglieds informiert und die gewonnenen Erkenntnisse oder getroffenen Entscheidungen sind allen bekannt und für jeden nachvollziehbar.

2.3 Organisationsgrenzen und -schnittstellen

Der Projektleiter bildet die alleinige Schnittstelle des Projektteams nach Außen, d.h. die gesamte Kommunikation mit den Kunden und Betreuern erfolgt über den Projektleiter (siehe Abbildung 2).

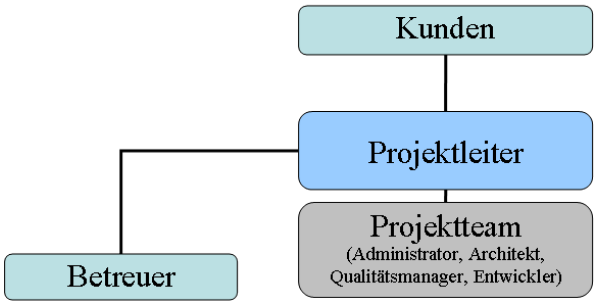


Abbildung 2: Schnittstellen innerhalb der Organisation

2.4 Verantwortlichkeiten

In diesem Abschnitt werden die verschiedenen Verantwortlichkeiten innerhalb des Projekts beschrieben.

Projektleiter: Wolfgang Hüttig (Stellvertreter: Michael Hahn)

Der Projektleiter ist der zentrale Ansprechpartner für den Kunden. Er koordiniert sämtlichen Schriftverkehr und die Kommunikation zwischen den Kunden, den Betreuern und dem Team. Er ist auch an erster Stelle für das Projekt verantwortlich.

Seine zentralen Aufgaben liegen in der Verwaltung des Projekts, d.h. er plant den Ablauf des Projekts und der einzelnen Phasen, er bewertet und kontrolliert abschließend als letzte Instanz alle erstellten Dokumente vor ihrer Freigabe und kümmert sich um die Verteilung der einzelnen Arbeitspakete während des Projekts.

Architekt: Michael Schneidt (Stellvertreter: Daniel Brüderle)

Der Architekt ist zuständig für die Architektur des zu entwickelnden Softwaresystems, die Beschreibung der grundlegenden Komponenten und deren Zusammenspiel innerhalb des Softwaresystems. Er trifft die grundlegenden Entscheidungen über die Art und den Aufbau der Software-Architektur und überwacht deren Einhaltung während der Projektlaufzeit.

Administrator: René Rehn (Stellvertreter: Michael Schneidt)

Der Administrator ist in der Projektarbeit zuständig für die gesamte Infrastruktur auf Software- sowie auch auf Hardware-Ebene. Er ist für den Betrieb und die Wartung der Server und den Betrieb der darauf ausgeführten Software verantwortlich. Weiterhin sorgt er für die Bereitstellung der nötigen Software-Infrastruktur, d.h. er sorgt dafür, dass die zur Verfügung stehende Software, wie z.B. Werkzeuge für Entwicklung, Test und Entwurf, die zur Realisierung des Projekts benötigt wird, für das Projekt eingerichtet und betriebsbereit ist.

Qualitätsmanagement: Michael Hahn, Firas Zoabi

Der Qualitätsmanager sorgt dafür, dass alle Dokumente und auch Software-Komponenten die bestmögliche Qualität besitzen. Diese Qualität soll durch die Definition und Einhaltung von Richtlinien, durch die ständige Durchführung von Qualitätskontrollen und durch die Erstellung von einheitlichen Dokumentvorlagen erreicht werden. So soll während des gesamten Projektverlaufs die Qualität der Dokumente und Software-Einheiten maximiert werden.

Entwickler: Michael Schneidt, René Rehn, Daniel Brüderle, Michael Hahn, Firas Zoabi, Xi Tu

Als Entwickler arbeiten alle Teammitglieder außer dem Projektleiter. Ein Entwickler arbeitet dabei im Rahmen der vom Projektleiter erteilten Aufgaben an Dokumenten und Software-Komponenten.

3 Managementprozess

In diesem Kapitel werden alle für die Verwaltung des Projekts relevanten Informationen aufgezeigt und beschrieben. Dazu gehören die Ziele des Managements, das Risikomanagement und Maßnahmen zur Projektüberwachung.

3.1 Managementziele und -prioritäten

In diesem Abschnitt werden die projektinterne Berichterstattung und die Prioritäten innerhalb des Projektes näher erläutert.

3.1.1 Zeiterfassung

Jedes Mitglied des Teams muss seine Aufwände erfassen und am Ende des Monats dem Projektleiter zukommen lassen, der die Aufwände in einer zentralen Datei erfasst und verwaltet.

3.1.2 Wöchentlicher Bericht

Jedes Mitglied soll in den wöchentlichen Treffen einen kurzen Bericht darüber abgeben was er in dieser Woche gemacht hat und was er in der kommenden Woche zu erledigen gedenkt.

3.1.3 Prioritäten

Das Projekt hat eine festgelegte Maximaldauer, daher besitzen die zeitlichen Anforderungen die höchste Priorität. Direkt danach folgen die Anforderungen des Kunden an die zu entwickelnde Software.

3.2 Annahmen, Abhängigkeiten und Einschränkungen

Das Projekt unterliegt einer strengen Zeitplanung, da es im Rahmen eines Studienprojekts durchgeführt wird und eine feste Laufdauer von 12 Monaten hat.

3.3 Risikomanagement

Aufgrund der geforderten sehr breiten Verwendungsmöglichkeiten im Rahmen der wissenschaftlichen Workflows besteht ein hohes Unsicherheitspotential im Bezug auf die Implementierungsdauer und auf sich ändernde Anforderungen.

Tabelle 2: Übersicht der Risiken und geplanten Gegenmaßnahmen

Risiko	**%**	**Kosten**	**Gegenmaßnahmen**
Mitarbeiter fällt zeitweise aus	15	2 MM	Wöchentliche Treffen (Informationsaustausch), Ernennung von Stellvertretern und Erstellung von internen Dokumentationen
Mitarbeiter fällt dauerhaft aus	10	4 MM	Rücksprache mit den Kunden
Änderung der Anforderungen	20	1-5 MM	genaue Anforderungserhebung und regelmäßige Rücksprache mit den Kunden
Anforderungen werden nicht oder nur teilweise erfüllt	30	3 MM	ausgiebige Tests, regelmäßige Rücksprache mit den Kunden und Prototyping
Ausgewähltes Werkzeug stellt sich als unbrauchbar heraus	20	0,5 MM	genaue Evaluation der Werkzeuge vor ihrem Einsatz
Nicht realisierbare Anforderungen	10	4 MM	frühzeitige Evaluierung der einzusetzenden Technologien und Rücksprache mit den Kunden
Chaotische Änderungen der Dokumente	10	2 MM	ausgiebige Dokumentprüfung durch Qualitätssicherung und Projektleiter Konfigurationsverwaltung mit Subversion

3.4 Projektüberwachung

In diesem Abschnitt werden verschiedene Methoden zur Überwachung des Projekts und seiner Fortschritte beschrieben. Dazu gehören beispielsweise Qualitätssicherungsmaßnahmen, Wasserstandsmeldungen und Activity Reports.

3.4.1 Qualitätssicherung und Konfigurationsmanagement

Um die Qualität der erstellten Dokumente gewährleisten zu können wird jedes Dokument bzw. jede Software-Einheit von zwei unabhängigen Instanzen geprüft. Zuerst wird die Software-Einheit durch die Qualitätsmanager auf formale Aspekte wie Style-Guide Konformität oder die Einhaltung des vorgeschriebenen Layouts hin untersucht. Anschließend wird die SE (Software-Einheit) falls sie Fehler enthält (positive QS-Prüfung) mit entsprechenden Kommentaren zurück an die Verfasser geschickt. Eine formal korrekte (negativ QS-geprüfte) SE wird an den Projektleiter weitergeleitet, der abschließend den Inhalt und die Korrektheit der gesamten SE prüft. Der Projektleiter leitet positiv geprüfte SE mit entsprechenden Kommentaren ebenfalls zurück an die Verfasser, der dann die entsprechenden Korrekturen ausführt. Negativ geprüfte SE werden vom Projektleiter freigegeben und erhalten so Auslieferungsstatus. Wird eine bereits freigegebene SE verändert oder an neue Bedürfnisse angepasst muss sie noch einmal den gesamten QS-Prozess durchlaufen. Sollte der Projektleiter eine SE prüfen, die seiner Meinung nach so viele Fehler oder Mängel enthält das eine Korrektur sich nicht lohnt, so kann er die SE verwerfen, d.h. die Verfasser erhalten eine Nachricht und müssen die SE neu verfassen. Eine detaillierte Beschreibung des gesamten Ablaufs des QS-Prozesses zeigt Abbildung 3.

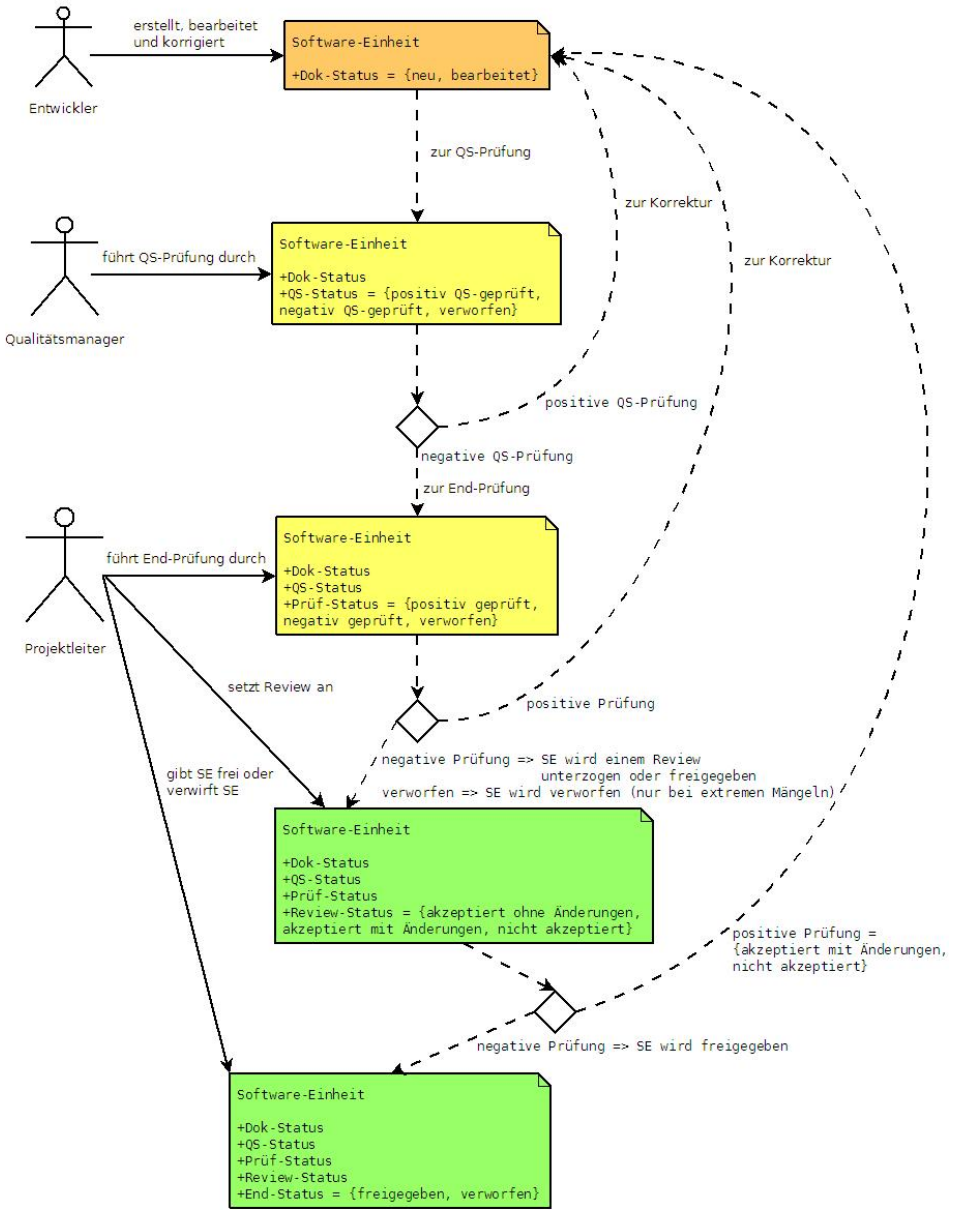


Abbildung 3: Prüfungsablauf für die Qualitätssicherung

Die Konfigurationsverwaltung ist ein wichtiger Bestandteil der Infrastruktur eines Projekts. In diesem Projekt wird zur Realisierung das Konfigurationsverwaltungssystem *Subversion* genutzt. Abbildung 4 zeigt das grundlegende Modell der Konfigurationsverwaltung und die Verwendung von Subversion (SVN). Hierzu werden verschiedene Umgebungen eingerichtet, die durch gewisse Rahmenbedingungen für bestimmte Tätigkeiten ausgelegt sind und so z.B. speziell für den Test oder die Implementierung eingerichtet sind. Die Referenz-Umgebung bildet die Zentrale der Konfigurationsverwaltung und wird durch Subversion realisiert.

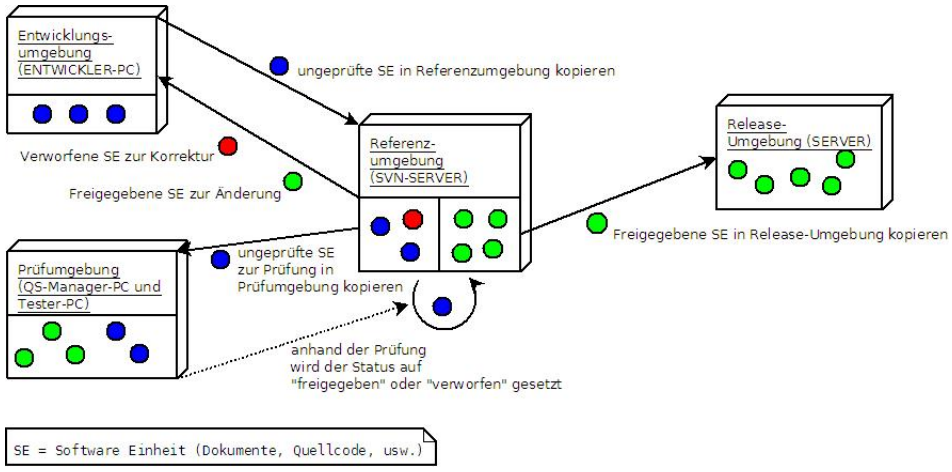


Abbildung 4: Das grundlegende Modell der Konfigurationsverwaltung

3.4.2 Reviews

Hier werden die Reviewtermine aufgelistet und der genaue Ablauf der Reviews beschrieben.

Anmerkung: Review der Spezifikation der zweiten und dritten Iteration werden gegebenenfalls durch einen Walkthrough ersetzt.

Tabelle 3: Interne und externe Reviewtermine

Dokument	**Art des Reviews**	**Begin des Reviewzeitraums (Versenden der Einladungen)**	**Ende des Reviewzeitraums**
Spezifikation erste Iteration	*intern*	18.09.09	23.09.09
Spezifikation erste Iteration	**extern**	28.09.09	07.10.09
Spezifikation zweite Iteration	*intern*	27.02.10	01.03.10
Grobentwurf	*intern*	06.11.09	08.11.09
Handbuch	*intern*	02. 06.10	04.06.10

Die Reviews werden in Gruppen von 5-6 Personen abgehalten. Dabei gilt folgende Rollenverteilung:

Moderator Der Moderator leitet und organisiert das Review. Bei Unklarheiten in Reviewpunkten führt er eine Einigung herbei.

Aktuar Der Aktuar protokolliert die angesprochenen Befunde und deren zugeordnete Fehlerkategorie. Er selbst nimmt nicht an Diskussionen und Wertungen der Fehler teil.

Gutachter Drei bis vier Gutachter untersuchen das Dokument nach vorher festgelegten Gesichtspunkten und ordnen ihren Befunden einer der folgenden Fehlerkategorien zu: Gut, Nebenfehler, Hauptfehler oder kritischer Fehler. Gut bedeutet, dass ein Abschnitt in Ordnung ist und keine Fehler oder Mängel aufweist. Ein Nebenfehler liegt vor, wenn die Nutzbarkeit des Dokuments durch diesen kaum beeinträchtigt wird und eine Korrektur bei Gelegenheit durchgeführt werden

kann und nicht dringend erforderlich ist. Bei einem Hauptfehler ist die Nutzbarkeit des Dokuments bereits erheblich beeinträchtigt und der Fehler sollte vor Freigabe des Dokuments behoben worden sein. Ein kritischer Fehler liegt vor, falls das Dokument für den vorhergesehenen Zweck unbrauchbar ist. Kritische Fehler müssen unbedingt vor der Freigabe des Dokuments behoben worden sein.

Spätestens drei Tage bei internen Reviews bzw. 1 Woche bei externen Reviews vor Beginn des Reviews schickt der Moderator Einladungen, mit dem zu begutachtenden Dokument an die Gutachter des Reviews. Bei externen Reviews nehmen die Kunden als Gutachter am Review teil. Die Gutachter untersuchen dann bis zum Review-Termin das Dokument anhand der vom Moderator festgelegten Kriterien. Am Review-Termin treffen sich alle Teilnehmer und der Moderator eröffnet und leitet das Review. Nun werden die Gutachter der Reihe nach nach ihren Befunden befragt und diese unter den Gutachtern diskutiert. Der Aktuar notiert alle genannten Befunde und deren Fehlerkategorien. Am Ende des Reviews gibt jeder Gutachter eine Empfehlung über den Annahmestatus des Dokuments ab.

Dabei gibt es folgende drei Möglichkeiten:

- *Akzeptieren ohne Änderungen*, falls das Dokument so freigegeben werden kann,
- *Akzeptieren mit Änderungen*, wenn das Dokument im Kern in Ordnung ist und nach der Korrektur freigegeben werden kann und
- *Nicht akzeptieren*, falls das Dokument so viele bzw. schwere Fehler enthält, dass nach der Korrektur ein erneutes Review notwendig ist.

Sollte das Review nach Ablauf von zwei Stunden nicht beendet sein, so vertagt der Moderator das Review frühestens auf den nächsten Tag. Am Ende des Reviews unterschreiben alle Beteiligten das Reviewprotokoll. Der Aktuar schickt das Reviewprotokoll an den Projektleiter, der es an die entsprechenden Entwickler weiterleitet, damit diese die nötigen Korrekturen durchführen können.

3.4.3 Wasserstandsmeldungen

Jede Woche soll eine Wasserstandsmeldung an die Betreuer des Studienprojeks SIMPL erfolgen in der folgende Punkte abgedeckt sind:

- In welcher Phase befindet sich das Projekt aktuell?
- Welche Arbeitspakete werden aktuell bearbeitet?
- Welche Arbeitspakete wurden abgeschlossen?
- Welche Probleme sind aufgetreten und wie wurden sie gelöst?
- Ist ein Treffen mit den Betreuern notwendig?

3.4.4 Activity Reports

Jedes Mitglied des Teams soll *jeden Monat* einen Activity Report erstellen und den Betreuern zukommen lassen, der folgende Punkte enthält:

- Geleistete Stunden
- Rolle im Projekt

- Was habe ich im Berichtszeitraum geleistet?
- Was steht im nächsten Berichtszeitraum an?
- Wieso konnte ich für einen längeren Zeitraum nicht am Projekt arbeiten?
- An welchen Projekttreffen habe ich teilgenommen?
- Was ist im Berichtszeitraum gut gelaufen?
- Was ist im Berichtszeitraum nicht optimal gelaufen?

3.5 Mitarbeiter

Das Team besteht aus eine festen Anzahl von sieben Mitarbeitern. Die Aufteilung der Mitarbeiter und ihre Rollen können im Abschnitt 2.4 nachgelesen werden.

4 Technische Prozesse

In diesem Abschnitt geht es um die Technischen Prozesse des Projektes. Darunter fallen die eingesetzten Technologien, Tools und natürlich auch Planung der Dokumentation und Meilensteine.

4.1 Methoden, Werkzeuge und Techniken

Im folgenden Abschnitt werden die eingesetzten Technologien und Tools näher beschrieben.

4.1.1 Technologien bzw. Werkzeuge für die Entwicklung

Es werden die folgenden Technologien für die Entwicklung der Software eingesetzt:

Hudson [1]

Hudson ist ein webbasiertes System zur kontinuierlichen Integration von Softwareprojekten.

Apache ODE [2]

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 ein Deployment von Prozessen zur Laufzeit (Hot Deployment) möglich, als auch die Analyse und Validierung von Prozessen.

Eclipse BPEL Designer[3]

Der Eclipse BPEL Designer ist ein Eclipse Plugin für die Modellierung von BPEL-Prozessen. Er verfügt über eine leicht verständliche GUI und über eine Syntaxsprüfung für BPEL-Prozesse. Zudem bietet der Eclipse BPEL Designer, in Kombination mit Apache ODE, eine Schritt für Schritt Ausführung von Prozessen.

IBM-DB2 [4]

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

4.1.2 Unterstützende Werkzeuge für die Projektarbeit

Folgende Werkzeuge sollen von den Entwicklern während des Projekts verwendet werden und diese bei ihrer Arbeit unterstützen.

Subversion [5]

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.

Apache Maven [6]

Maven ist ein Projekt-Management-Tool zur standardisierten Erstellung und Verwaltung von Java-Programmen. Mit Maven ist es möglich viele Schritte, die die Entwickler normalerweise von Hand erledigen müssen, zu automatisieren.

Lyx [7]

LyX ist ein Textverarbeitungsprogramm, mit dem es möglich ist, auf einfache Art und Weise LaTeX-Dokumente zu erstellen.

LaTeX [8]

Latex ist eine Designsprache, mit der sich hochwertige Aussehende Dokumente erstellen lassen, die im Normalfall als PDF exportiert werden.

Eclipse-Subversion [9]

Wird während der Implementierung in Eclipse eingesetzt um den Quellcode der Anwendung komfortabel direkt in das Repository übertragen zu können.

TortoiseSVN [10]

Eine grafische Oberfläche, die in Windows eingebunden wird, um SVN Repositories über die grafische Oberfläche von Windows verwalten zu können.

4.2 Dokumentationsplan

Dieser Abschnitt befasst sich mit den Dokumenten die beim Erreichen eines Meilensteins vorliegen müssen und mit der Organisation und Durchführung von Reviews.

4.2.1 Meilensteine

Die Abnahmeverantwortlichkeit für interne Dokumente liegt beim Projektleiter und für externe Dokumente beim Kunden. Interne Meilensteine sind in Tabelle 4 *kursiv* dargestellt und externe Meilensteine unterstrichen. Beim Erreichen von externen Meilensteinen werden dem Kunden die Dokumente des Meilensteins zur Verfügung gestellt. Sofern nicht anders angegeben, sind alle Dokumente auf Deutsch verfasst.

4.2.2 Termine

Dieser Abschnitt enthällt die Meilensteine und Termine des Projekts.

Tabelle 4: Meilensteine mit auszuliefernden Dokumenten

Meilenstein	**Dokumente**	**Abnahmekriterien**	**Zieltermin**

Meilenstein	Dokumente	Abnahmekriterien	Zieltermin
Angebot	Angebot (PDF)	Der Kunde akzeptiert das Angebot.	4. September 2009
Projektplan	Projektplan (PDF und LaTeX)	Projektplan ist vollständig und wird von Qualitätsmanager und Projektleiter akzeptiert.	30. September 2009
Spezifikation 1. Version	(interne Dokumente) *Erster Entwurf des Spezifikationsdokuments, Reviewprotokoll, Korrigiertes Spezifikationsdokument*	Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein.	28. September 2009
Spezifikation erste Iteration	Kundenfassung der Spezifikation (PDF und LaTeX) und Prototyp des erweiterten Eclipse BPEL-Designers (interne Dokumente) *Reviewprotokoll*	Der Kunde hat am **externen Review** des Dokuments teilgenommen. Er hat nach den Korrekturen keine weiteren Anmerkungen und alle im Review entdeckten Fehler und Mängel sind beseitigt.	15. Oktober 2009
Grobentwurf gesamtes System	Kundenfassung und zeitnahe Präsentation des Grobentwurfs (PDF und LaTeX) (interne Dokumente) *Reviewprotokoll*	Der Kunde akzeptiert den Grobentwurf.	7. November 2009
Feinentwurf erste Iteration	Kundenfassung Feinentwurf (PDF und LaTeX) (interne Dokumente) *Reviewprotokoll*	Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein.	15. Januar 2010
Modul-implementierung erste Iteration	Quellcode der Module (Java)	Alle spezifizierten Use-Cases der ersten Iteration sind innerhalb der Module implementiert.	22. Januar 2009
Erweiterungen erste Iteration	Quellcode der Erweiterungen von Apache ODE und Eclipse BPEL Designer (Java)	Alle spezifizierten Erweiterungen der ersten Iteration sind implementiert.	22. Januar 2010

Meilenstein	Dokumente	Abnahmekriterien	Zieltermin
Modultest erste Iteration	Testfälle und Testprotokoll (PDF und LaTeX), Korrigierter Quellcode (Java)	Alle erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert. Weiterhin muss eine Code-Überdeckung von 95% erreicht worden sein.	22. Januar 2010
Integration und Systemtest erste Iteration	Testfälle und Testprotokoll (PDF und LaTeX), Korrigierter Quellcode (Java)	Alle aus der Spezifikation erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert.	01. Februar 2010
Spezifikation zweite Iteration	Kundenfassung der Spezifikation (PDF und LaTeX) und Prototyp des erweiterten Eclipse BPEL-Designers (interne Dokumente) *Reviewprotokoll*	Der Kunde akzeptiert die Spezifikation	22. Februar 2009
Feinentwurf zweite Iteration	Kundenfassung Feinentwurf (PDF und LaTeX) (interne Dokumente) *Reviewprotokoll*	Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein.	29. März 2009
Modul-implementierung zweite Iteration	Quellcode der Module (Java)	Alle spezifizierten Use-Cases der zweiten Iteration sind innerhalb der Module implementiert.	07. Mai 2010
Modultest zweite Iteration	Testfälle und Testprotokoll (PDF und LaTeX), Korrigierter Quellcode (Java)	Alle erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert. Weiterhin muss eine Code-Überdeckung von 95% erreicht worden sein.	07. Mai 2010
Integration	(internes Dokument) Integrationsprotokoll	Module wurden erfolgreich zu einer lauffähigen Software integriert.	21. Mai 2010
Integrations-test	Testfälle und Testprotokoll (PDF und LaTeX), Korrigierter Quellcode (Java)	Alle erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert.	21. Mai.10

Meilenstein	Dokumente	Abnahmekriterien	Zieltermin
Systemtest	Testfälle und Testprotokoll (PDF und LaTeX), Korrigierter Quellcode (Java)	Alle aus der Spezifikation erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert.	04. Juni 10
Handbuch	Kundenfassung Handbuch (auf Englisch und Deutsch) (PDF und LaTeX) (interne Dokumente) *Reviewprotokoll*	Der Kunde hat keine Anmerkungen und alle im Review entdeckten Fehler und Mängel sind beseitigt.	04. Juni 10
Abnahmetest	CD/DVD mit: Installationspaket (RAR), Dokumentation der Erweiterungen (auf Englisch und Deutsch) (PDF und LaTeX), Quellcode (Java) und evtl. bereits ausgelieferte aktualisierte Dokumente	Der Kunde akzeptiert die Software.	04. Juni 10

4.2.3 Dokumentationen

Dokumentationen sind als PDF zu erstellen. Hierfür stehen die Tools Miktex und LyX zur Verfügung.

5 Arbeitspakete, Zeitplan und Budget

In diesem Kapitel werden die verschiedenen Phasen des Projekts, die definierten Arbeitspakete und Meilensteine und die Terminplanung des Projekts beschrieben.

5.1 Beschreibung der Phasen

In diesem Abschnitt wird jede der definierten Phasen und die darin enthaltenen Aufgaben kurz näher erläutert.

5.1.1 Analyse

Das Ziel der Analysephase ist es durch Kundengespräche die funktionalen und nichtfunktionalen Anforderungen zu erheben, die der Kunde an das zu entwickelnde Softwaresystem stellt. Das Ziel ist es einen Einblick in den momentanen Zustand des Arbeitsbereichs für den die Software entwickelt werden soll zu bekommen, um bereits realisierte Anforderungen aufzunehmen.

Am Ende der Analysephase wird aus den gewonnenen Erkenntnissen ein Anforderungskatalog erstellt, der als Referenz für die spätere Spezifikation der Software verwendet wird.

5.1.2 Einarbeitung

Da wenige bis gar keine Ansätze für Teilbereiche des Projekts wie z.B. die Verwendung von BPEL in Scientific Workflows existieren, müssen alle vorhandenen Ansätze evaluiert oder z.T. auch neue Ansätze geschaffen werden. Die Einarbeitungsphase dient dazu sich in die verschiedenen Themenbereiche einzuarbeiten und festzustellen in welchem Umfang Erweiterungsmaßnahmen notwendig sind.

5.1.3 Spezifikation

In dieser Phase werde die zu entwickelnde Softwareeinheiten genau spezifiziert, d.h. in einem Dokument, der Spezifikation, werden alle funktionalen und nichtfunktionalen Anforderungen an die Softwareeinheit und ihre Schnittstellen präzise, vollständig und überprüfbar definiert. Ebenso werden spätere Abläufe im Umgang mit der Software und ihren Funktionen beschrieben. Die Spezifikation dient darüber hinaus als Referenz für viele nachfolgende Dokumente, wie z.B. das Handbuch und den Entwurf und sie ist unerlässlich für die Testphase und die spätere Wartung oder für spätere Erweiterungen der Software. Anhand der Spezifikation kann sich auch der Kunde davon überzeugen, dass wirklich seine Anforderungen umgesetzt werden.

5.1.4 Entwurf

In der Entwurfsphase wird die Architektur der Softwareeinheit festgelegt und in verschiedenen Granularitäten in den Dokumenten Grob-Entwurf (Architektur des Gesamtsystems) und Fein-Entwurf (Architektur der einzelnen Module) festgehalten.

5.1.5 Codierung

In der Codierungsphase werden die Anforderungen aus der Spezifikation mit der Architektur des Entwurfs umgesetzt. Am Ende dieser Phase existiert eine ausführbare Softwareeinheit.

5.1.6 Test

Die Testphase dient dazu die Software gegen die Spezifikation zu prüfen, d.h. alle Anforderungen aus der Spezifikation werden mit geeigneten Testfällen überprüft und die Softwareeinheit entsprechend korrigiert, falls sie nicht die gewünschten Ergebnisse liefert. Dazu wird eine Reihe von Testfällen definiert und deren Ausführung in einem Testprotokoll aufgezeichnet. Diese zwei Dokumente sind auch für den Kunden wichtig, da sie als Referenz für seine internen Tests während der Abnahme genutzt werden können.

5.1.7 Abnahmetest

Am Ende der Testphase erfolgt die Abnahme der Software und aller genannten Dokumente (siehe Tabelle 1). Dem Kunden wird damit Zeit zur Verfügung gestellt sicherzustellen, dass das von uns gelieferte Produkt seinen Anforderungen entspricht. Ist das der Fall, dann gilt das Projekt als abgeschlossen, andernfalls werden die nötigen Nachbesserungen oder Änderungen durchgeführt.

5.1.8 Puffer

Ein extra Zeitfenster vor dem endgültigen Abgabetermin Ende Mai 2010. In dieser Phase können noch kleinere Änderungswünsche, die der Kunde nach dem Abnahmetest eventuell hat, umgesetzt werden.

5.2 Arbeitspakete

Die im vorigen Abschnitt definierten Phasen werden nun in Arbeitspakete gegliedert und aufgeteilt. Die nachfolgende Tabelle enthält alle Arbeitspakete und eine kurze Beschreibung der darin enthaltenen Aufgaben.

Id	**Bezeichnung**	**Aufgaben**	**Dokumente**
anly#proplan	Projektplan	Erstellung des Projektplans	Projektplan
anly#angeb	Angebot	Erstellung des Angebots für die Kunden	Angebot
einb#tools • *#1* • *#2* • *#3* • *#4* • *#5* • *#6*	Einarbeitung in Tools	Einarbeitung in: • Apache ODE (einb#tools#1) • Eclipse BPEL Designer • Maven • Hudson • SVN • IBM DB2 (einb#tools#6)	Foliensammlung als Nachschlagewerk

Id	Bezeichnung	Aufgaben	Dokumente
einb#konzept • *#1* • *#2* • *#3* • *#4*	Einarbeitung in Konzepte	Einarbeitung in: • BPEL (einb#konzept#1) • Apache ODE • Eclipse BPEL Designer • BPEL/SQL (einb#konzept#4)	Berichte als Grundlage für Spezifikation und Implementierung bzw. Erweiterung der Technologien.
spezIter1#erstell	Spezifikation erstellen	Erstellung der Spezifikation	Spezifikationsdokument
spezIter1#rev1	1.Review der Spezifikation	Interner Review der Spezifikation	Reviewprotokoll
spezIter1#kor1	1.Korrektur der Spezifikation	Korrektur der Spezifikation nach dem 1.Review	Korrigiertes Spezifikationsdokument
spezIter1#rev2	2.Review der Spezifikation	Weiterer Review der Spezifikation diesmal mit den Kunden	Reviewprotokoll
spezIter1#kor2	2.Korrektur der Spezifikation	Korrektur der Spezifikation nach dem 2.Review	Kundenfassung der Spezifikation
grEntw#erstell	Grobentwurf erstellen	Erstellung des Grobentwurfs (zusammenhang zwischen Komponenten)	Dokument des Grobentwurfs
grEntw#rev	Review des Grobentwurfs	Interner Review des Grobentwurfs	ReviewProtokoll
entw#kor	Korrektur des Entwurfs	Korrektur des Entwurfs nach dem Review	Grobentwurf
feinEntwIter1# erstell	Feinentwurf erste Iteration erstellen	Erstellen des Feinentwurfes	Dokument des Feinentwurf
codeIter1# erstell	Implementierung	Implementierung der Usecases	Code
spezIter2# erstell	Spezifikation erstellen	Erstellung der Spezifikation	Spezifikationsdokument
spezIter2#rev	1.Review der Spezifikation	Interner Review der Spezifikation	Reviewprotokoll
feinEntwIter2# erstell	Feinentwurf erste Iteration erstellen	Erstellen des Feinentwurfes	Dokument des Feinentwurf
codeIter2#erstell	Implementierung	Implementierung der Usecases	Code
handb#erstell	Handbuch erstellen	Erstellung eines Handbuches	Handbuch
handb#rev	Handbuch review	Interner review des Handbuches	Review Protokoll
handb#kor	Korrigiern des Handbuchs	Handbuch Korrektur	Kundenfassung Handbuch

Id	Bezeichnung	Aufgaben	Dokumente
test#mod	Modultest (durchgeführt wärend jeder Iteration)	Planung und Durchführung des Modulttests mit anschließender Korrektur der entdeckten Fehler	Testprotokoll
test#int	Integrationstest	Planung und Durchführung des Integrationstests mit anschließender Korrektur der entdeckten Fehler	Testprotokoll
test#sysIter1	Systemtest	Planung und Durchführung des Systemtests nach der ersten Iteration, mit anschließender Korrektur der entdeckten Fehler	Testpotokoll
test#sys	Systemtest	Planung und Durchführung des Systemtests mit anschließender Korrektur der entdeckten Fehler	Testpotokoll
puff#abn	Abnahme	Abnahme durch die Kunden	
puff#erw	Erweiterung/Anpassung	Durchführung möglicher Erweiterungen und/oder Anpassungen	

5.3 Abhängigkeiten

Hier werden die Abhängigkeiten zwischen den Arbeitspaketen und zu externen Elementen genauer beschrieben.

Iterationen:

Es bestehen Abhängigkeiten zwischen der ersten Iteration und den beiden darauf folgenden. Keine der Iterationen kann begonnen werden, ohne das die erste Iteration abgeschlossen ist, da in dieser das Basissystem darstellt. Iteration zwei und drei können auch unabhängig von einander erfolgen und die Reihenfolgte könnte vertauscht werden, da zwischen ihnen kein Zusammenhang besteht.

5.4 Zuteilung des Budgets und der Ressourcen auf Projektfunktionen und Aktivitäten

Dieser Abschnitt enthält die berechneten Aufwände der einzelnen Projekt-Phasen. Tabelle 6 zeigt die Zeiträume der einzelnen Phasen und die Aufwände bzw. das zeitliche Budget, das diesen zugeteilt ist.

Tabelle 6: Phasen des Projekts

Phase	Von / Bis	Aufwand
Analyse	13.05.2009-17.07.2009	306 h
Einarbeitung	17.07.2009-17.09.2009	208 h
Spezifikation	07.08.2009-01.03.2010	544 h
Entwurf	15.10.2009-29.03.2010	628 h
Implementierung	13.11.2009-07.05.2010	880 h
Test	15.12.2010-04.06.2010	544 h
Abnahmetest	04. 06. 10	–

5.5 Zeitplan

Iterationen

In der Implementierungsphase werden zwei Iterationen durchlaufen. Jede der zwei Iterationen gliedert sich dabei standardmäßig in Feinentwurf, Implementierung und Modultest. Ein iterationsinterner Systemtest ist nur nach der 1.Iteration vorgesehen, da hier das Basissystem implementiert wird, auf dem alle weiteren Iterationen aufbauen. Nachfolgend werden die zu realisierenden Funktionaliäten der einzelnen Iterationen beschrieben.

1. Iteration Implementierung der Grundfunktionalität des SIMPL-Rahmenwerks. Dazu gehören:

- Umsetzung eines Plug-In System für die Anbindung von verschiedenen Datenquellen.
- Die statische Anbindung von Datenquellen, wobei vorerst nur relationale Datenbanken unterstützt werden sollen.
- Bereitstellung von generischen BPEL-Aktivitäten im Eclipse BPEL Designer für den Zugriff auf Datenquellen.
- Bereitstellung von Query-Editoren für die verschiedenen DM-Aktivitäten.
- Umsetzung einer grundlegenden Admin-Konsole, in der das Auditing aktiviert werden kann, sowie Benutzername und Passwort, für die spätere Authentifizierung an Datenquellen.

2. Iteration Implementierung der weitergehenden Funktionalität des SIMPL-Rahmenwerks.

- Anbindung von Dateisystemen am Beispiel des CSV-Dateiformats.
- Unterstützung von Referenzen in BPEL (siehe [3]), damit auf Daten auch per Referenz im Workflow zugegriffen werden kann. Dies wird aus Gründen der Performanz benötigt, da die Datenübergabe zwischen Workflow und Web Service standardmäßig per Wert erfolgt, was bei großen Datenmengen (bis zu Gigabytes oder gar Terabytes im wissenschaftlichen Bereich) zu erheblichen Performanzeinbußen führt.
- Bereitstellung einer Datenquellen-Registry, mit Hilfe derer Datenquellen über den Eclipse BPEL Designer (siehe [10]) manuell durch den Benutzer oder dynamisch durch das SIMPL Rahmenwerk ausgewählt werden können.

- Unterstützung einer automatischen Auswahl von Datenquellen zur Laufzeit. Dabei kann der Benutzer Anforderungen an Datenquellen formulieren und eine Strategie auswählen, mit deren Hilfe eine Datenquelle, die seine Anforderungen erfüllt, ausgewählt wird.
- Validation von der SIMPL-DM-Aktivitäten

Literatur

[1] Hudson, https://hudson.dev.java.net/

[2] Apache Ode, https://hudson.dev.java.net/

[3] Eclipse BPEL-Designerhttp://www.eclipse.org/bpel

[4] IM-DB2http://www-01.ibm.com/software/data/db2/

[5] Subversion http://subversion.tigris.org/

[6] Maven http://maven.apache.org/

[7] Lyx http://www.lyx.org/

[8] LaTex http://miktex.org/

[9] Eclipse-Subversion http://subclipse.tigris.org/

[10] TortoiseSVN http://tortoisesvn.tigris.org/