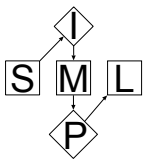


SIMPL, Universitätsstraße 38, 70569 Stuttgart
Katharina Görlach,
Peter Reimann,
Mirko Sonntag
Universitätsstraße 38
70569 Stuttgart

Wolfgang Hüttig
Projektleiter SIMPL
Universitätsstraße 38
70569 Stuttgart

Angebot 07. August 2009

Sehr geehrte Frau Görlach, sehr geehrter Herr Reimann, sehr geehrter Herr Sonntag,

vielen Dank für Ihr Interesse an einem Angebot unserer Firma. Gerne würden wir ihre Vorstellungen im Rahmen eines Softwareprojekts umsetzen.

Sie finden hierzu auf den folgenden Seiten unser ausführliches Angebot. Über eine Zusammenarbeit in diesem Projekt würden wir uns sehr freuen.

Bei Rückfragen können Sie sich selbstverständlich gerne jederzeit mit uns in Verbindung setzen.

Mit freundlichen Grüßen

Wolfgang Hüttig

Studienprojekt SIMPL

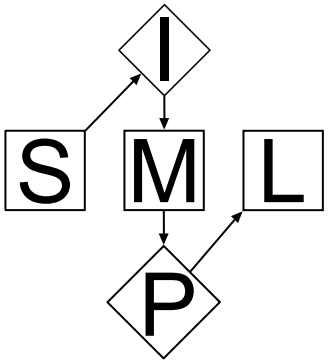
Angebot

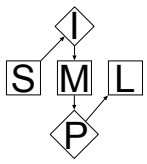
Version 1.1

07. August 2009

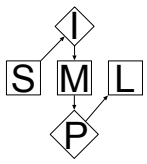
Verfasser:

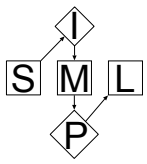
Daniel Brüderle, Firas Zoabi, Michael Hahn, Wolfgang Hüttig





Inhaltsverzeichnis





1 Ausgangssituation

Zur Modellierung und Ausführung von Business Workflows wird die Sprache BPEL (Business Process Execution Language) bereits lange und effektiv eingesetzt. Die Stärken von BPEL, die ,wie der Name schon sagt, im Bereich der Geschäftsprozesse liegen, sollen nun auch für wissenschaftliche Workflows genutzt werden können. Im wissenschaftlichen Bereich gelten andere Anforderungen und dafür ist BPEL bis jetzt noch nicht geeignet. Da aber auch im wissenschaftlichen Bereich immer öfter Workflows eingesetzt werden, sei es bei der Durchführung von komplexen Berechnungen im Rahmen von Simulationen oder bei der Ausführung von Experimenten, sollen mithilfe einer Anpassung von BPEL die wissenschaftlichen Bedürfnisse an BPEL erfüllt werden. Gekennzeichnet sind wissenschaftliche Workflows in der Regel durch große Datenmengen, die Heterogenität der Daten und somit den Bedarf nach großer Rechenkapazität. Diesen Anforderungen soll durch die neu zu entwickelnde Software Rechnung getragen werden.

1.1 Projektbeteiligte

In diesem Abschnitt geht es um die beteiligten Parteien des Projekts. Diese sind, nachfolgend als die Kunden bezeichnet, Katharina Görlach, Peter Reimann und Mirko Sonntag, sowie die Firma SIMPL, nachfolgend als Auftragnehmer bezeichnet, die durch Wolfgang Hüttig vertreten wird. Der Auftragnehmer besteht aus den Projektmitarbeitern: Wolfgang Hüttig, Michael Schneidt, René Rehn, Michael Hahn, Firas Zoabi, Daniel Brüderle und Xi Tu, wobei Wolfgang Hüttig der zentrale Ansprechpartner des Auftragnehmers für das Projekt ist. Im Rahmen dieses Angebots kommt dabei ein Vertrag zwischen den Kunden und dem Auftragnehmer zustande.

Die Kontaktdaten der Kunden sind wie folgt:

Dipl.-Inf. Katharina Görlach

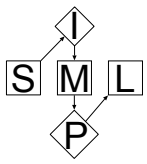
Institut für Architektur von Anwendungssystemen

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

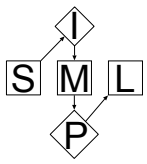
Fax: +49 (0)711 7816-472

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

1.2 Entwicklungsgrundlagen

Die Entwicklung von SIMPL erfolgt nach erprobten softwaretechnischen Prinzipien. Dadurch wird eine hohe Qualität des Produkts gewährleistet. Wichtige Qualitäten sind dabei die Wartbarkeit und die Erweiterbarkeit des Systems, da es über einen längeren Zeitraum bestehen soll und in Zukunft um die Anbindung weiterer Datenquellen, Konzepte für den Datenzugriff und den Umgang mit weiteren Datenformaten ergänzt werden soll. Eine weitere wichtige Qualität ist die Skalierbarkeit des Systems, die eine sehr flexible Infrastruktur erlauben muss, da die Computersysteme, auf denen SIMPL später ausgeführt wird, in ihrer Leistung sehr weit auseinander gehen, d.h. vom normalen Desktop-Computer bis zum Supercomputer kann und soll alles möglich sein.

Robustheit und Usability der Software sind für uns innerhalb des Entwicklungsprozesses von großem Interesse. Die Usability sollte sich vor allem an unerfahrene und Nutzer mit wenig Kenntnissen im Umgang mit Workflows und BPEL richten und dafür die bestmögliche Transparenz liefern, d.h. das die interne Prozesslogik der Software bestmöglich vom Benutzer abgeschirmt wird und das er eine möglichst einfache und schnell verständliche Schnittstelle zur Software erhält, um die Verwendung von SIMPL für alle Benutzergruppen zu ermöglichen.



2 Projektaufgabe

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.

3 Projektteam-Organisation

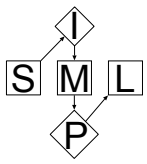
In diesem Abschnitt wird die Organisation des Projektteams vorgestellt. Alle Teammitglieder haben dabei mindestens eine bestimmte Rolle, die mit entsprechenden Aufgaben und Pflichten verknüpft ist. Nachfolgend werden diese Rollen beschrieben und die Teammitglieder, die diese Rolle besitzen, angegeben. Danach folgen noch die Kontaktdaten aller Teammitglieder, damit der Kunde bei spezifischen Fragen direkt mit dem Verantwortlichen oder bei nichterreichen des Projektleiters mit dessen Stellvertreter Kontakt aufnehmen kann.

3.1 Rollen im Team

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 er 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 und 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. Die Architektur soll die Einhaltung der unter Abschnitt ??? genannten Softwarequalitäten sicherstellen.

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..

Qualitätsmanagment: 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.

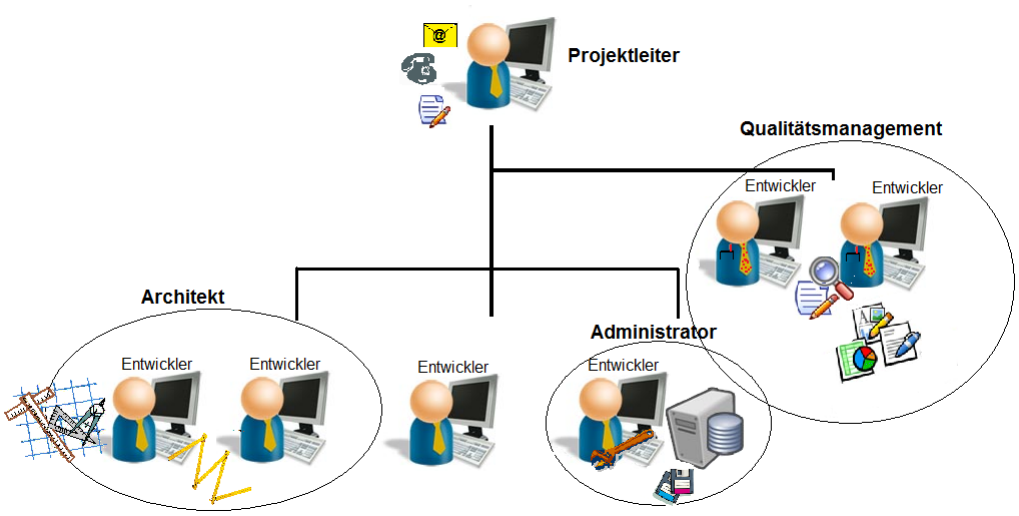


Abbildung 1: Rollen im Team

3.2 Kontaktdaten der Projektteam-Mitglieder

Wolfgang Hüttig: w.huettig(@)yahoo.de

Daniel Brüderle: daniel(@)brotherlee.de

Michael Schneidt: schneimi(@)studi.informatik.uni-stuttgart.de

René Rehn: Rene_Rehn(@)gmx.de

Firas Zoabi: Firas.Zoabi(@)web.de

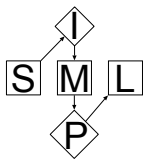
Tu Xi: tu_xi(@)hotmail.com

Michael Hahn: hahnml(@)studi.informatik.uni-stuttgart.de

4 Nutzen

Die Arbeit im wissenschaftlichen Bereich und die Datenmengen, die damit verbunden sind, werden immer größer, heterogener und komplexer. Zur Vereinfachung und zur Automatisierung werden im Moment nur vereinzelt Scientific Workflows eingesetzt.

Durch das in diesem Projekt erstellte Rahmenwerk soll die Modellierung und die Ausführung von wissenschaftlichen Workflows mit BPEL, welches im Business-Bereich bereits viele Jahre erfolgreich eingesetzt wird, vereinfacht werden. Dazu soll das Rahmenwerk den Datenzugriff direkt aus einem Workflow heraus ermöglichen. . Dafür werden



der Eclipse BPEL-Designer, Apache ODE und BPEL selbst erweitert, um so die direkte Anbindung von Datenquellen innerhalb von BPEL-Prozessen zu realisieren. Dadurch ist es auch für ungeübte Workflow-Benutzer wie z.B. Wissenschaftler und Ingenieure möglich, einfach und schnell wissenschaftliche Workflows zu erstellen und zu nutzen.

5 Leistungen im Rahmen des Angebots

5.1 Auszuliefernde Software-Einheiten

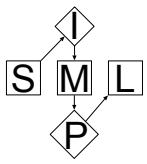
- Anforderungskatalog
- Projektplan
- Spezifikation
- Begriffslexikon
- Entwurf
- Handbuch
- Dokumentation der Erweiterungen
- Testfälle
- Testprotokolle
- Installationspaket, das alle benötigten Software-Komponenten beinhaltet
- Quellcode

Alle Dokumente werden im PDF-Format und als LaTeX-Quelldateien geliefert.

5.2 Software-Einheiten, die nicht zum Lieferumfang gehören

Folgende Dokumente gehören nicht zum Lieferumfang:

- Alle internen und organisatorischen Dokumente, wie z.B. Protokolle



6 Projektdurchführung

6.1 Terminplanung

Das Studienprojekt läuft über zwei Semester und endet im Mai 2010. Jeder Projektteilnehmer hat dabei einen Arbeitsaufwand von ca. 480 Stunden zu leisten, dadurch entsteht ein Gesamtaufwand von ca. 3360 Stunden für alle 7 Teilnehmer. Die Projektphasen Analyse und Einarbeitung beanspruchen dabei 35%, Spezifikation und Entwurf (inkl. Fein-Entwurf) 24% und zuletzt die Codierung und der Test 33%. (siehe Abbildung 2)

Nach Berücksichtigung aller Einflussfaktoren von Seiten des Kunden und des Entwicklungsteams wird durch diese Planung vorausgesetzt, dass der Kunde das Endprodukt Mitte April 2010 testet und zwischen Anfang und Ende Mai 2010 nach Bedarf noch kleinere Änderungen durchgeführt werden.

Phase	**Von / Bis**	**Aufwand**
Analyse	13.05.2009-17.07.2009	658 h
Einarbeitung	17.07.2009-07.09.2009	526 h
Spezifikation	07.09.2009-31.10.2009	547 h
Entwurf	31.10.2009-24.12.2209	243 h
Implementierung	24.12.2009-12.03.2010	779 h
Test	12.03.2010-15.04.2010	334 h
Abnahmetest	15.04.2010	–
Puffer	15.04.2010-13.05.2010	273 h

Tabelle 1: Phasen des Projekts

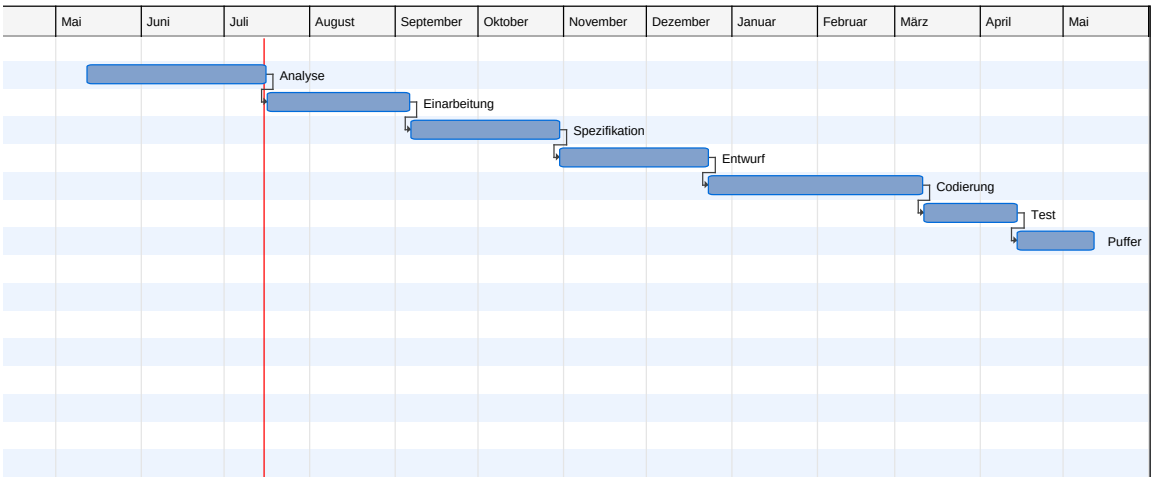


Abbildung 2: Terminplanung für die Entwicklungsphasen

6.2 Beschreibung der Phasen

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

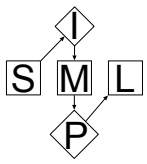
6.2.1 Analyse

Das Ziel der Analysephase ist es, durch Kundengespräche die funktionalen und nichtfunktionalen Anforderungen zu erheben, die der Kunde an das zu entwickelnde Softwaresystem stellt.

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

6.2.2 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.



6.2.3 Spezifikation

In dieser Phase wird die zu entwickelnde Software genau spezifiziert, d.h. in einem Dokument, der Spezifikation, werden alle funktionalen und nichtfunktionalen Anforderungen an die Software 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. 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 alle seine Anforderungen aufgenommen und nach seinen Vorstellungen spezifiziert wurden.

6.2.4 Entwurf

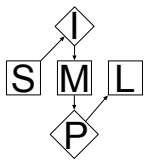
In der Entwurfsphase wird die Architektur der Software festgelegt und in verschiedenen Granularitäten in den Dokumenten Grob-Entwurf (Architektur des Gesamtsystems) und Fein-Entwurf (Architektur der einzelnen Module) festgehalten.

6.2.5 Implementierung

In der Implementierungsphase werden die Anforderungen aus der Spezifikation mit der Architektur des Entwurfs umgesetzt. Nach der Integration der implementierten Software-Module exisitert, am Ende dieser Phase, eine ausführbare Software.

6.2.6 Test

Die Testphase enthält die drei Unterphasen Modultest, Integrationstest und Systemtest. Durch den Modultest werden die einzelnen Software-Module auf ihre korrekte Funktionalität innerhalb des Moduls getestet. Im Integrationstest wird die Integration der Module zum Gesamtsystem überprüft, d.h. es wird überprüft, ob alle Module korrekt zusammenarbeiten. Der Systemtest dient dazu die Software gegen die Spezifikation zu prüfen, d.h. alle Anforderungen aus der Spezifikation werden mit geeigneten Testfällen überprüft und die Software entsprechend korrigiert, falls sie nicht die gewünschten Ergebnisse liefert. Für alle drei Testarten gilt, dass immer eine Reihe von Testfällen definiert und deren Ausführung in einem Testprotokoll aufgezeichnet wird. Falls während eines Tests Fehler auftreten, so werden diese nach Ausführung des Test von den



Entwicklern korrigiert und dann erneut geprüft. Die in der Testphase erzeugten Dokumente, die Testfälle und Testprotokolle, sind auch für den Kunden wichtig, da sie als Referenz für seine internen Tests während der Abnahme genutzt werden können.

6.2.7 Abnahmetest

Am Ende der Testphase erfolgt die Abnahme der Projekts und aller genannten Dokumente (siehe Kapitel 5.1).

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. Sollten nach den Nachbesserungen noch erhebliche Mängel bestehen, so können diese z.B. im Rahmen eines HiWi-Jobs korrigiert werden.

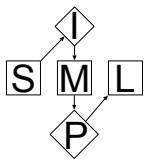
6.2.8 Puffer

Die Puffer-Phase ist 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.

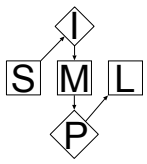
6.3 Arbeitspakete

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.

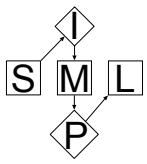
Id	**Bezeichnung**	**Aufgaben**	**Dokumente**
anly#proplan	Projektplan	Erstellung des Projektplans	Projektplan (PDF und LaTeX)
anly#angeb	Angebot	Erstellung des Angebots für die Kunden	Angebot (PDF)
anly#angeb#kor	Angebot korrigieren	Korrektur des Angebots	Angebot 2.Version (PDF)



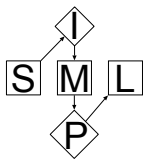
Id	Bezeichnung	Aufgaben	Dokumente
einb#tools • *#1* • *#2* • *#3* • *#4* • *#5* • *#6*	Einarbeitung in Tools	Einarbeitung in: • Apache ODE (einb#tools#1) • Eclipse BPEL Designer • Maven • Hudson • SVN • IBM DB2 (einb#tools#6)	(interne Dokumente) Foliensammlung als Nachschlagewerk
einb#konzept • *#1* • *#2* • *#3* • *#4*	Einarbeitung in Konzepte	Einarbeitung in: • BPEL (einb#konzept#1) • Apache ODE • Eclipse BPEL Designer • BPEL/SQL (einb#konzept#4)	(interne Dokumente) Berichte als Grundlage für Spezifikation und Implementierung bzw. Erweiterung der Technologien.
spez#erstell	Spezifikation erstellen	Erstellung der Spezifikation	(internes Dokument) Erster Entwurf des Spezifikationsdokuments
spez#rev1	1.Review der Spezifikation	Interner Review der Spezifikation	(internes Dokument) Reviewprotokoll
spez#kor1	1.Korrektur der Spezifikation	Korrektur der Spezifikation nach dem 1.Review	(internes Dokument) Korrigiertes Spezifikationsdokument



Id	Bezeichnung	Aufgaben	Dokumente
spez#rev2	2.Review der Spezifikation	Weiterer Review der Spezifikation diesmal mit den Kunden	(internes Dokument) Reviewprotokoll
spez#kor2	2.Korrektur der Spezifikation	Korrektur der Spezifikation nach dem 2.Review	Kundenfassung der Spezifikation (PDF und LaTeX)
grEntw#erstell	Grobentwurf erstellen	Erstellung des Grobentwurfs (zusammenhang zwischen Komponenten)	(internes Dokument) Grobentwurf
grEntw#rev	Review des Grobentwurfs	Review des Grobentwurfs	(internes Dokument) ReviewProtokoll
entw#kor	Korrektur des Entwurfs	Korrektur des Grobentwurfs nach dem Review	Kundenfassung Grobentwurf (PDF und LaTeX)
feinEntw# erstell	Feinentwurf erstellen	Erstellen des Feinentwurfs	(internes Dokument) Feinentwurf
feinEntw# rev	Feinentwurf Review	Interner Review des Feinentwurfs	(internes Dokument) ReviewProtokoll
feinEntw# kor	Feinentwurf Korrektur	Korrektur des Feinentwurfs nach dem Review	Kundenfassung Feinentwurf (PDF und LaTeX)
code#BPELdes	BPEL-Designer erweitern	Erweiterung des Eclipse BPEL-Designer	Quellcode der Erweiterungen (Java)
code#Apache	Apache ODE erweitern	Erweiterung der Apache ODE Workflow-Engine	Quellcode der Erweiterungen (Java)
code#module	Module implementieren	Basismodule des Rahmenwerks implementieren	Quellcode der Module (Java)
code#integr	Integration	Integration der einzelnen Basismodule zum Rahmenwerk	(internes Dokument) Integrationsprotokoll
handb#erstell	Handbuch erstellen	Erstellung eines Handbuches inklusive Installationsanweisungen	(internes Dokument) Handbuch



Id	Bezeichnung	Aufgaben	Dokumente
handb#rev	Handbuch review	Interner Review des Handbuches	(internes Dokument) Review Protokoll
handb#kor	Korrigiern des Handbuchs	Handbuch Korrektur	Kundenfassung Handbuch (PDF und LaTeX)
test#mod#erst	Modultests erstellen	Erstellung der Modultestfälle	Testfälle (PDF und LaTeX)
test#mod	Modultest	Durchführung des Modulttests und Protokollierung der Fehler	Testprotokoll (PDF und LaTeX)
test#mod#kor	Modultest Korrektur	Korrektur aller entdeckten Fehler anhand des Testprotokolls	Korrigierter Quellcode (Java)
test#integ#erst	Integrationstests erstellen	Erstellung der Integrationstestfälle	Testfälle (PDF und LaTeX)
test#integ	Integrationstest	Durchführung des Integrationstests und Protokollierung der Fehler	Testprotokoll (PDF und LaTeX)
test#integ#kor	Integrationstest Korrektur	Korrektur aller entdeckten Fehler anhand des Testprotokolls	Korrigierter Quellcode (Java)
test#sys#erst	Systemtests erstellen	Erstellung der Systemtestfälle	Testfälle (PDF und LaTeX)
test#sys	Systemtest	Durchführung des Systemtests und Protokollierung der Fehler	Testprotokoll (PDF und LaTeX)
test#sys#kor	Systemtest Korrektur	Korrektur aller entdeckten Fehler anhand des Testprotokolls	Korrigierter Quellcode (Java)

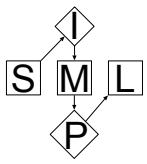


Id	Bezeichnung	Aufgaben	Dokumente
puff#abn	Abnahme	Abnahme durch die Kunden	CD/DVD mit: Installationspaket (RAR), Dokumentation der Erweiterungen (PDF und LaTeX), Quellcode (Java) und evtl. bereits ausgelieferte aktualisierte Dokumente
puff#erw	Erweiterung/Anpassung	Durchführung möglicher Erweiterungen und/oder Anpassungen	Alle veränderten Dokumente

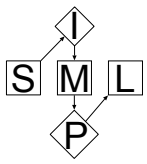
6.4 Meilensteine

Die Abnahmeverantwortlichkeit für interne Dokumente (siehe Arbeitspaket-Tabelle) liegt beim Projektleiter und für externe Dokumente beim Kunden. Interne Meilensteine sind in der nachfolgenden Tabelle *kursiv* dargestellt und externe Meilensteine unterstrichen.

Meilenstein	zugehörige Arbeitspakete	Abnahmekriterien	Zieltermin
Angebot	*anly#angeb, anly#angeb#kor*	Kunde akzeptiert das Angebot.	07. August 2009
Projektplan	*anly#proplan*	Projektplan ist vollständig und wird von Qualitätsmanager und Projektleiter akzeptiert.	14. August 2009
Einarbeitung	*einb#tools#1* bis *einb#tools#6*, *einb#konzept#1* bis *einb#konzept#4*		



Meilenstein	**zugehörige Arbeitspakete**	**Abnahmekriterien**	**Zieltermin**
Spezifikation 1. Version	*spez#erstell, spez#rev1, spez#kor1*	Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein.	17. September 2009
Spezifikation	*spez#rev2, spez#kor2*	Der Kunde hat keine Anmerkungen und alle im Review entdeckten Fehler und Mängel sind beseitigt.	01. Oktober 2009
Grobentwurf	*grEntw#erstell, grEntw#rev, entw#kor*		07. November 2009
Feinentwurf	*feinEntw#erstell, feinEntw#rev, feinEntw#kor*		01. Dezember 2009
Handbuch			17. April 2010
Modul implemen tierung	*code#module*		08. Februar 2010
Erweiterungen	*code#BPELdes, code#Apache*		08. Februar 2010
Integration	*code#integr*		22. Februar 2010
Modultest	*test#mod#erst, test#mod, test#mod#kor*		04. März 2010
Integrationstest	*test#integ#erst, test#integ, test#integ#kor*		23. März 2010
Systemtest	*test#sys#erst, test#sys, test#sys#kor*		17. April 2010
Abnahmetest	*puff#abn*		



Meilenstein	zugehörige Arbeitspakete	Abnahmekriterien	Zieltermin

6.5 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 gesammten 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 detailierte Beschreibung des gesamten Ablaufs des QS-Prozesses zeigt Abbildung 3.

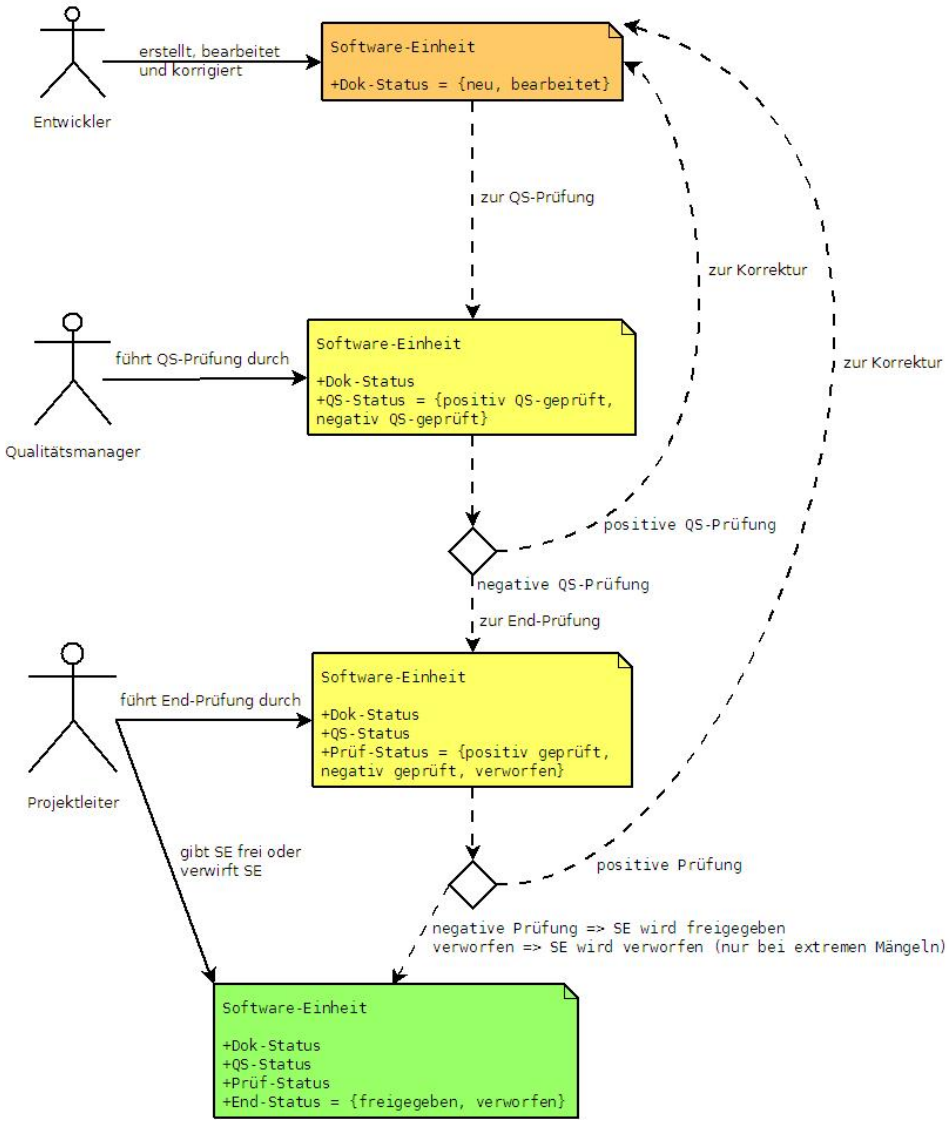


Abbildung 3: Prüfungsablauf für die Qualitätssicherung

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 eingerich-

tet sind. Die Referenz-Umgebung bildet die Zentrale der Konfigurationsverwaltung und wird durch Subversion realisiert. In Abbildung 5 wird der Entstehungsprozess von SE aufgezeigt und welche Umgebungen dabei durchlaufen werden.

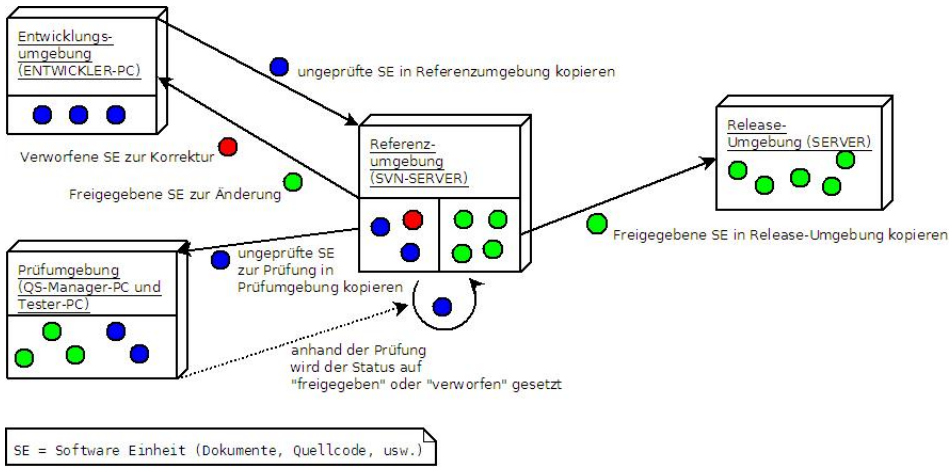


Abbildung 4: Das grundlegende Modell der Konfigurationsverwaltung

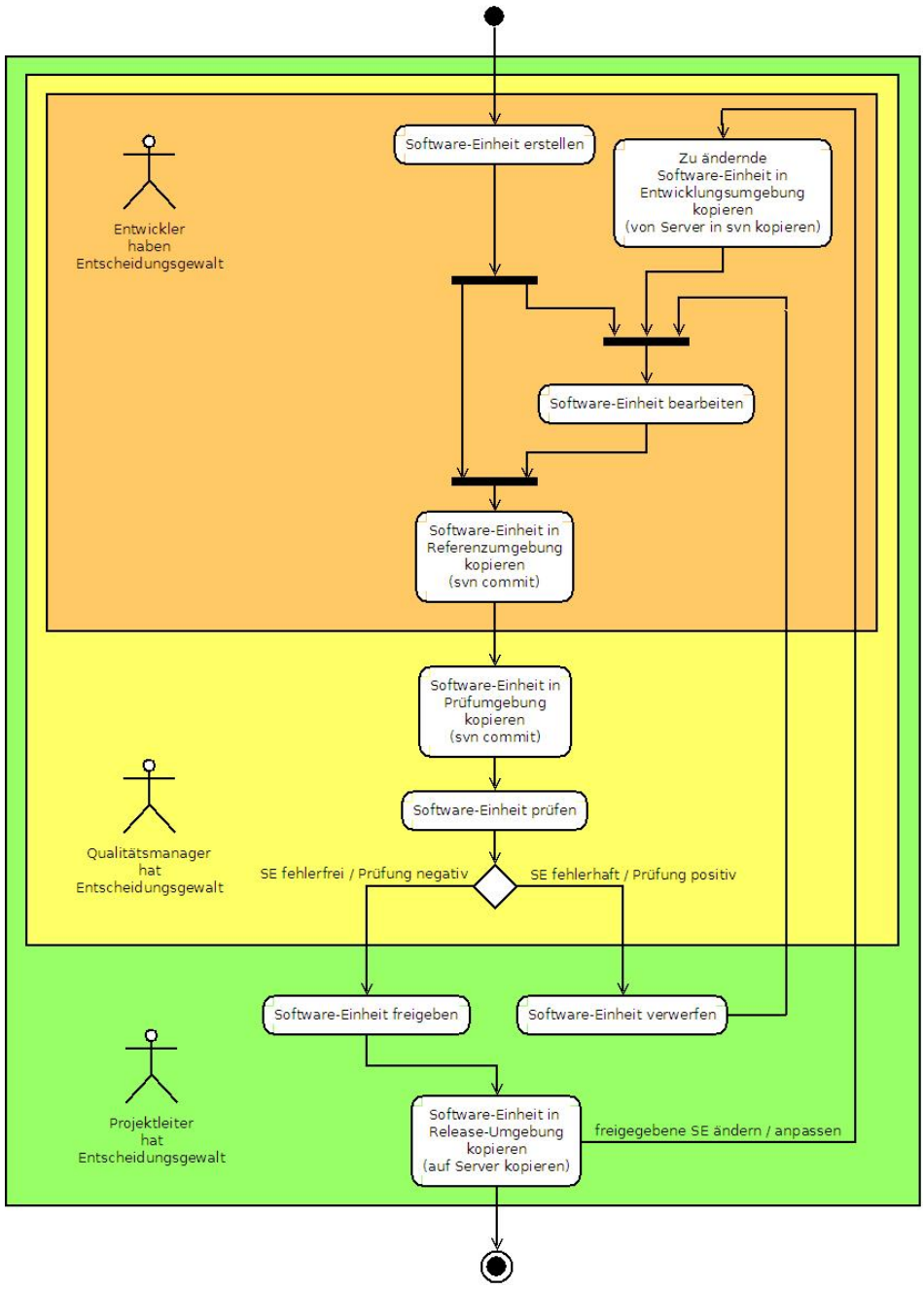
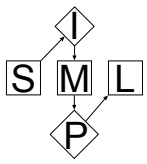


Abbildung 5: Entstehungsprozess für Software-Einheiten



7 Unsere Konditionen

In diesem Abschnitt werden die Kosten und die Risiken des Projekts aufgezeigt und beschrieben.

7.1 Kostenschätzung

Unsere internen Berechnungen nach der Funktion-Point-Methode ergab 268 Adjusted Function Points (AFP). Aus einer Referenztabelle für abgeschlossene Projekte ergibt sich folgende Kostenschätzung:

Gesamtentwicklungszeit:

19 MM (Mann-Monate)

Arbeitszeit pro Mitarbeiter:

19 MM / 7 Mann = 2,71 Monate Entwicklungszeit pro Mitarbeiter

Zuschlag Arbeitszeit (wg. Krankheit / Urlaub):

2,71 Monate * 1,10 = 2,981 Monate Entwicklungszeit pro Mitarbeiter

Mitarbeiterstunden:

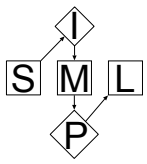
Die Mitarbeiter haben alle eine 5 Tage Woche mit jeweils 8 Stunden, können also im Monat 160 Stunden arbeiten.

Bei einer effektiven Entwicklungszeit von gerundet 3 Monaten (2,981) wird jede Person 480 Stunden arbeiten.

Stundensätze

Entwickler : 69€ zzgl. MwSt.

Projektleiter : 99€ zzgl. MwSt.



Gesamtkostenschätzung:

6 Entwickler x 69€/Stunde x 160 Stunden x 3 Monate = 198.720€

1 Projektleiter x 99€/Stunde x 160 Stunden x 3 Monate = 47.520€

Gesamt: 246.240€ zzgl. 19 % MwSt. = 293.025€

Da es sich um ein Studienprojekt handelt, wird kein Geld bezahlt. Die Berechnungen sind hypotetisch und wurden nur zu Planungszwecken durchgeführt.

7.2 Risiken

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.

Risiko	%	**Kosten**	**Gegenmaßnahmen**
Mitarbeiter fällt aus	20	2 MM	Wissen teilen, Zeitarbeit
Produkthaftung	5	1-10 MM	Versicherung abschliessen
Änderung der Anforderungen	50	1-5 MM	Flexible Struktur herstellen, regelmäßige Rücksprache mit Kunden
Qualität nicht ausreichend	30	3 MM	längere Testphase
Werkzeuge nicht einsetzbar	20	5 MM	Organisatorischer Slack schaffen
Infrastruktur Probleme	20	1 MM	einheitliche, zertifizierte Hardware
Anforderungen werden unterschätzt	40	5 MM	Schulungen
Software nicht lauffähig	20	6 MM	Prototyping
“Wildwuchs” der Änderungen	10	2 MM	Versionsmanagement
Entwicklungsverzug	30	6 MM	Überstunden
Kundenwünsche unrealistisch	30	1-5 MM	Anforderungen reduzieren.