



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

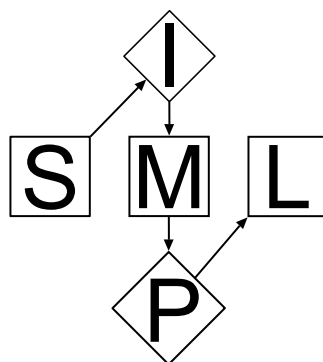
Angebot

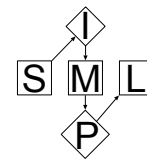
Version 1.1

07. August 2009

Verfasser:

Michael Hahn, Daniel Brüderle, Firas Zoabi, Wolfgang Hüttig





Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Ausgangssituation | 5 |
| 1.1 | Projektbeteiligte | 5 |
| 1.2 | Entwicklungsgrundlagen | 6 |
| 2 | Projektaufgabe | 7 |
| 2.1 | Grobspezifikation | 7 |
| 2.1.1 | Nichtfunktionale Anforderungen: | 7 |
| 2.1.2 | Funktionale Anforderungen: | 8 |
| 2.1.3 | Benutzer: | 11 |
| 2.2 | Grobes Gesamtkonzept und Software-Architektur | 11 |
| 2.3 | Systemumgebung | 13 |
| 3 | Projektteam-Organisation | 14 |
| 3.1 | Rollen im Team | 14 |
| 3.2 | Kontaktdaten der Projektteam-Mitglieder | 16 |
| 4 | Nutzen | 16 |
| 5 | Projektdurchführung | 17 |
| 5.1 | Terminplanung | 17 |
| 5.2 | Beschreibung der Phasen | 19 |
| 5.2.1 | Analyse | 19 |
| 5.2.2 | Einarbeitung | 19 |
| 5.2.3 | Spezifikation | 19 |
| 5.2.4 | Entwurf | 19 |
| 5.2.5 | Implementierung | 20 |
| 5.2.6 | Test | 20 |
| 5.2.7 | Abnahmetest | 20 |



| | | |
|----------|---|-----------|
| 5.2.8 | Puffer | 21 |
| 5.3 | Arbeitspakete | 21 |
| 5.4 | Meilensteine | 25 |
| 5.5 | Qualitätssicherung und Konfigurationsmanagement | 27 |
| 5.5.1 | Durchführung der Reviews | 32 |
| 6 | Unsere Konditionen | 32 |
| 6.1 | Kostenschätzung | 32 |
| 6.2 | Risiken | 33 |
| 7 | Rechtliche Rahmenbedingungen | 34 |
| 7.1 | Gewährleistung | 34 |
| 7.2 | Pflichten des Auftragnehmers | 35 |
| 7.3 | Pflichten des Auftraggebers | 35 |
| 7.4 | Nutzungs- und Urheberrecht | 35 |
| 7.5 | Gültigkeit des Angebots | 35 |



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. Kennzeichnet 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](mailto: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

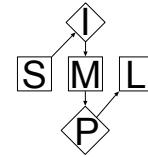
Fax: +49 (0)711 7816-472

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.

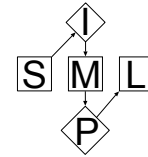
In den nachfolgenden Abschnitten werden die aufgenommenen Anforderungen der Kundengespräche, das vorläufige grobe Gesamtkonzept und die vorläufige Software-Architektur, sowie die vorläufige vollständige Systemumgebung des Rahmenwerks beschrieben.

2.1 Grobspezifikation

Folgende grundlegende Anforderungen sollen im Rahmen des Projektes umgesetzt werden.

2.1.1 Nichtfunktionale Anforderungen:

1. Die Schnittstellen des Rahmenwerks sollen leicht erweiterbar sein und insgesamt soll ein modularer Aufbau zugrunde liegen.
2. Fehlertoleranz: Scopes mit Handler. CompensationHandler und FaultHandler implementieren Logik für die Kompensation von Fehlern (rollback).
3. Ausfallsicherheit soll so gut wie möglich sein.
4. Usability sollte sich an Benutzer orientieren und dazu die größtmögliche Transparenz liefern. 



5. Es gibt keine bestimmte Infrastruktur in der das System später läuft. ODE läuft überall lokal und die Datenbanken sind verteilt.
6. Skalierbarkeit: keine bestimmte Infrastruktur => von schlechten PC's bis zu Supercomputern ist alles möglich.
7. Performance soll so gut wie möglich sein.

2.1.2 Funktionale Anforderungen:

1. Das Rahmenwerk soll als Eclipse Plugin verwendet werden.
2. Verarbeitung von großen heterogenen Datenmengen im Rahmen eines wissenschaftlichen Workflows.

BPEL:

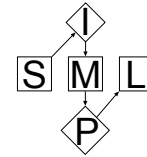
1. Fehlerbehandlung / Kompensation im Workflow.
2. Unterstützung von Referenzen in BPEL-Prozessen
 - (a) Referenzen (z.B. auf eine Tabelle) im SQL Statement via BPEL Variable
 - (b) Implementierung eines Reference Resolution System in einer Testumgebung
 - i. Referenz/ Pointer im BPEL Prozess auflösen
 - ii. Bei Daten, die nicht für die Prozesslogik, soll die Referenz / der Pointer vom Service aufgelöst werden (zur Entlastung der Engine)

BPEL Aktivitäten:

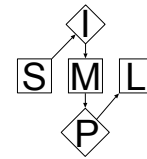
1. Alle generischen BPEL-Aktivitäten, die für den Umgang mit Datenquellen benötigt werden, müssen erstellt werden.
2. Das Anlegen, Löschen und Ändern von DB-Tabellen soll möglich sein.

Anbindung von Datenquellen:

1. Late-binding bei Datenquellen: Kriterien zur Beschreibung von Datenquellen – Annotation von Datenquellen, Modellierung der Anforderungen an eine Datenquelle (durch den Benutzer), Strategien zur Auswahl einer geeigneten Datenbank.



2. Anforderungen an eine Datenquelle sollen, falls die Datenquelle über einen Web-Service eingebunden ist, mit Annotationen mit/über "QUALITY OF SERVICE" angegeben werden, sonst statische Auswahl einer Datenquelle zur Modellierungs- oder Laufzeit.
3. Ein Nutzer soll angeben können, was die Datenquelle können muss und das Rahmenwerk wählt eine passende Datenquelle aus (Strategie).
4. Spezifizierung der Anforderungen, die sinnvoll an eine Datenquelle gestellt werden können sollten, wird über die Definition eines Anforderungskataloges, der nach Fertigstellung mit dem Kunden abgesprochen werden muss, realisiert.
5. Ebenso sollen auch Strategien für die Auswahl von Datenquellen anhand definierter Anforderungen entwickelt und mit dem Kunden abgesprochen werden.
6. Die Strategien, die zur Auswahl einer Datenquelle anhand der vom Benutzer angegebenen Anforderungen verwendet werden, müssen selbst definiert und anschließend mit dem Kunden abgesprochen werden.
7. Die Auswertung der Annotationen und die Verarbeitung der darin enthaltenen Informationen soll, ebenfalls selbst definiert und anschließend mit dem Kunden abgesprochen werden.
8. Adressierung der Datenbanken: Statisch über konkrete Adresse oder dynamisch über logische Adresse.
9. Anbindung soll generisch sein, so viele Anbindungen wie möglich realisieren. Als Minimum gilt: IBM DB2 + OpenSourceDB's (mind. 1*RDB + mind. 1*XMLDB) + mind. 1*SensorDB (z.B. TinyDB).
10. Es sollen auch Zugriffe auf mehrere Datenquellen innerhalb eines Prozess möglich sein, dazu legt jede Aktivität seine eigene Datenquellen-Referenz.
11. Alle Anfragesprachen für den Datenbankzugriff müssen unterstützt werden, mindestens SQL und XQuery.
12. Transaktionen sollen innerhalb von DB und BPEL-Prozessen unterstützt werden. Konzepte frei wählbar.
13. Es soll möglich sein, dass Daten aus Datenbanken aus einem Prozess auch in lokale Dateien exportiert werden können. Dabei gilt: XML als Standardformat (siehe IBM Ansatz: RDB-Tabellen ↔ XML).



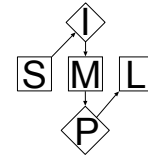
14. Datenbanken existieren bereits und müssen nicht durch das Rahmenwerk erstellt werden. Weiterhin sollen aber Schema-Definitionen möglich sein und das Erstellen, Ändern und Löschen von Tabellen innerhalb der Schemas.

Autorisierung und Authentifizierung:

1. Autorisierung und Authentifizierung soll momentan nur für Datenquellen bereitgestellt werden, allerdings soll eine spätere Erweiterung einfach realisiert werden können.
2. Für die Authentifizierung sind verschiedene Verfahren gewünscht, sodass in jeder Situation das Bestmögliche verwendet wird.
3. Die Autorisierung und die Authentifizierung sollen, über einen Single-Sign-On entweder im Prozess oder in der Instanz, beim Zugriff auf Datenquellen durchgeführt werden.
4. Autorisierung und Authentifizierung sollen auf die Ausführung von Prozessen beschränkt sein.
5. Autorisierungs- und Authentifizierungsparameter für einen Prozess sollen auf Datenquellenebene spezifiziert werden, d.h. bei Datenbanken über Schema und bei Dateisystemen z.B. über extra Dateien.
6. Autorisierung und Authentifizierung ist bei allen Datenquellen notwendig und soll auf verschiedene Arten möglich sein.
7. Angaben über Autorisierung und Authentifizierung sollen extra abgefragt und als Nachricht an die DB geschickt werden.

Monitoring:

1. Monitoring der Prozessausführung: ExecutingHistory in einer angebundenen Datenbank speichern (nicht im lokalen Speicher der Engine).
2. Die Datenbank für das Monitoring muss über das Rahmenwerk frei wählbar sein und es soll nicht über mehrere Datenbanken verteilt sein.
3. Für unsere Zwecke muss ein komplettes Monitoring erstellt werden, d.h. ein Monitoring für BPEL, ODE und anderen Datenquellenaktionen.



4. Das vorhandene Monitoring von BPEL und Apache ODE muss um das Monitoring unserer Funktionalitäten und die Möglichkeit eine variable DB als Monitoring-DB anzugeben erweitert werden.
5. Es soll ein Modell für das Monitoring (Informationen) erstellt und mit den Kunden abgesprochen werden.
6. Die Speicherdauer der Monitoring-Daten soll variabel als Parameter übergeben werden können.
7. Das Monitoring soll standardmäßig aktiv sein, aber auch auf Benutzerwunsch abschaltbar. Dies soll auch über den Eclipse BPEL-Designer mit Deployment-Deskriptoren definierbar sein.



Demo-Programm:

1. Die Inhalte der Demo werden später geklärt. Es sollen darin die wichtigsten Use-Cases vorgeführt und ein kurzer Einblick in die Dokumentation gegeben werden.
2. In der Demo soll der Zugriff auf die IBM DB2 Datenbank, eine OpenSource RDB und eine OpenSource XMLDB, sowie evtl. auf Sensornetzdaten mittels TinyDB (falls öffentlich zugänglich) gezeigt werden. Ebenso soll eine kurze Vorstellung der Monitoring-DB-Anbindung, mit einer der oben genannten Datenbanken, gezeigt werden.



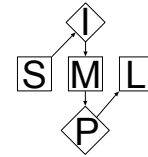
2.1.3 Benutzer:

1. Benutzer sind im Normalfall Wissenschaftler und Ingenieure.
2. Benutzer haben keine bis wenig Vorkenntnisse im Bereich Workflow und Informatik.

2.2 Grobes Gesamtkonzept und Software-Architektur

In diesem Abschnitt wird das vorläufige Gesamtkonzept und die vorläufige Software-Architektur der zu entwickelnden Software beschrieben.

Wie in Abbildung 1 dargestellt, soll das Rahmenwerk in drei Schichten unterteilt werden.



Die Präsentationsschicht wird durch den Eclipse BPEL-Designer realisiert. Dort ist es möglich BPEL-Prozesse mit Datenmanagement-Funktionen zu modellieren und auszuführen. Die Datenmanagement-Funktionen werden durch Extension Activity Plugins in den Designer eingebunden. Einstellungen, die erst zur Laufzeit dem SIMPL Core übergeben werden sollen, können über Deployment-Deskriptoren festgelegt werden. Von dort wird der Prozess an die Anwendungsschicht übergeben.

In der Anwendungsschicht wird der Prozess ausgewertet und von der Workflow-Engine ausgeführt. Bei der Ausführung des Prozess kann auf Funktionalitäten des SIMPL Core zugegriffen werden. Der SIMPL Core realisiert folgende Funktionen und Konzepte über eine Service Component Architecture (SCA):

- Authentifizierte und autorisierte Datenquellenzugriffe
- Monitoring der Datenquellenzugriffe
- Datenquellenunabhängige Zugriffe auf heterogene Daten
- Verwendung von Transaktionen für Datenzugriffe
- Auswahl geeigneter Datenquellen anhand definierter Strategien und vom Benutzer angegebener Eigenschaften
- Binden und Auflösen von logischen Namen

Die Datenschicht beinhaltet alle verwendbaren Datenquellen mit den dazugehörigen Authentifizierungs- und Autorisierungsinformationen und den Eigenschaften der jeweiligen Datenquelle.

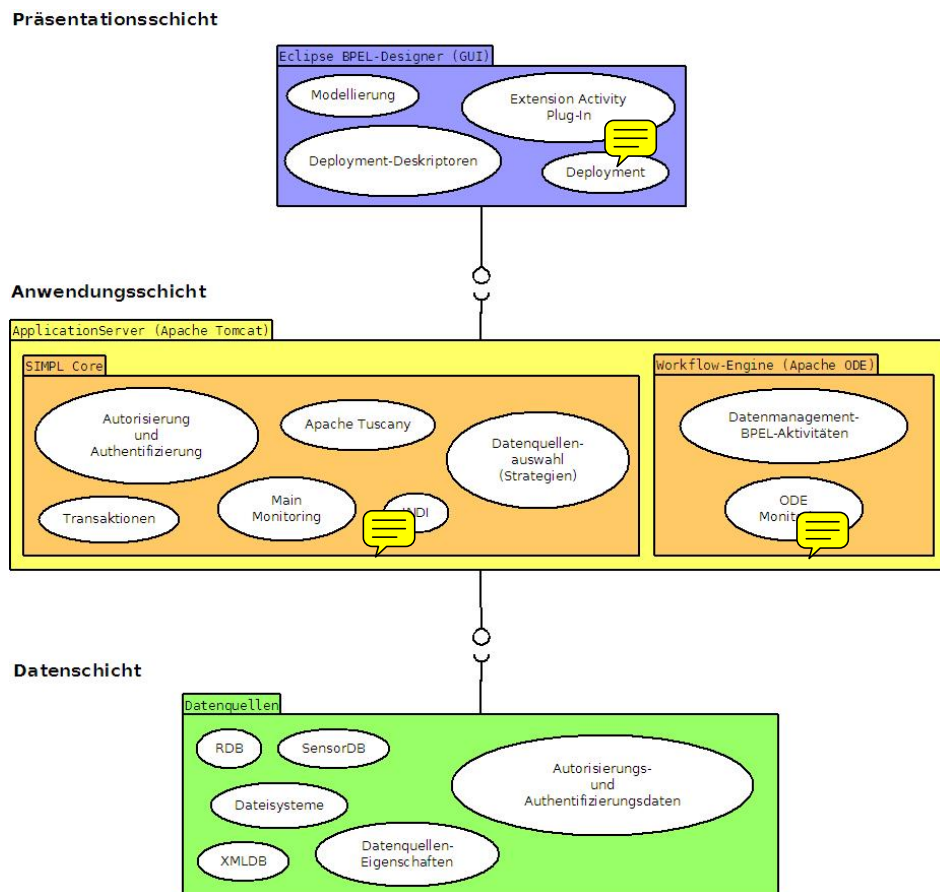


Abbildung 1: Übersicht über das grobe Gesamtkonzept

2.3 Systemumgebung

Die Systemumgebung (siehe Abbildung 2) besteht aus den unterschiedlichen Datenquellen, Eclipse mit dem BPEL-Designer Plug-In und dem Apache Tomcat Web-Server, in dem das Rahmenwerk und die Apache ODE Workflow-Engine ausgeführt wird. Dabei läuft die benötigte Software auf dem lokalen Rechner des Benutzers und die Datenquellen sind verteilt auf verschiedenen Servern.



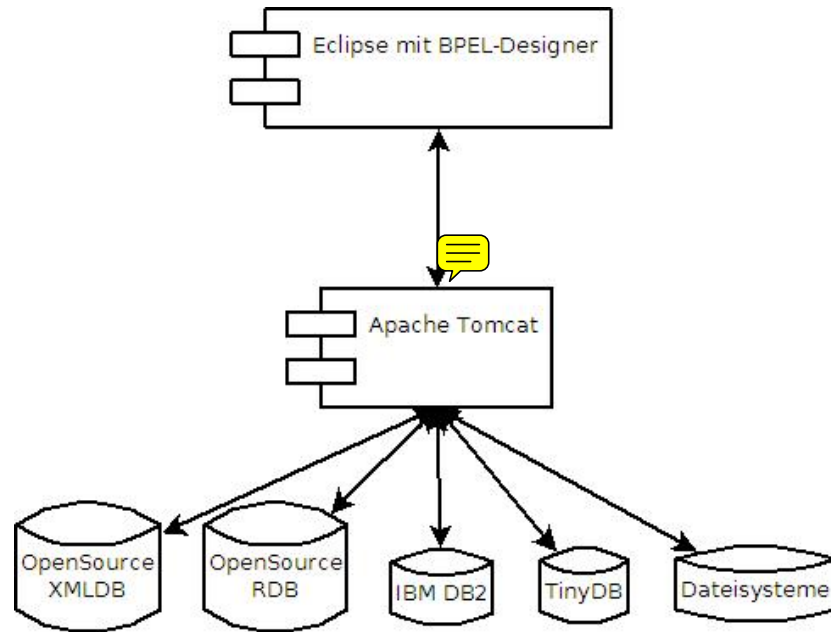


Abbildung 2: Übersicht über die Systemumgebung von SIMPL

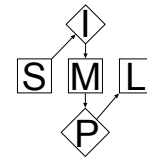
3 Projektteam-Organisation

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 nicht Erreichen 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 2.1 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..

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. Der genaue Ablauf der Qualitätssicherung wird in Abschnitt 5.5 beschrieben.

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.

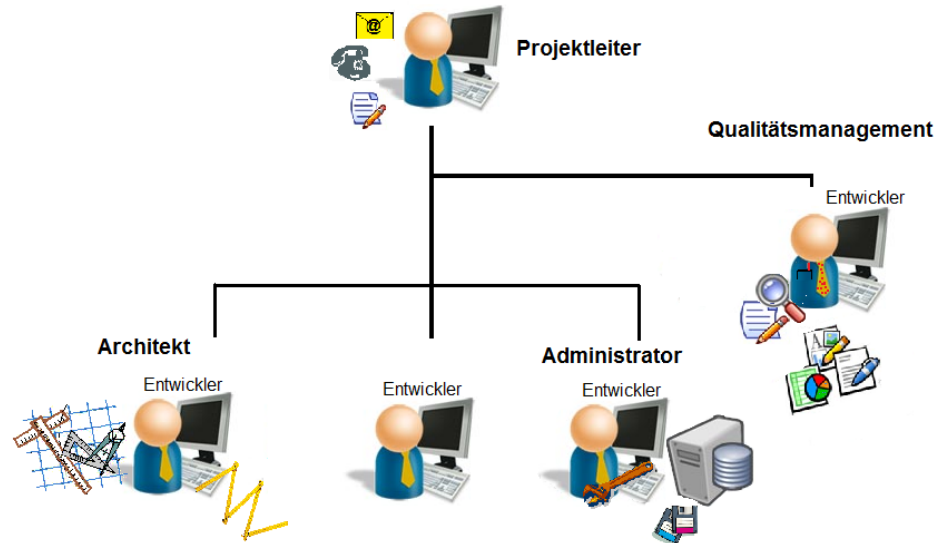
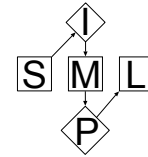


Abbildung 3: Rollen im Team

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

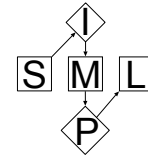
Firas Zoabi: Firas.Zoabi(@)web.de

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 wissenschaftliche 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.

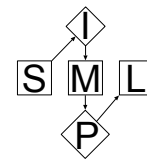
5 Projektdurchführung

Als Prozessmodell für die Durchführung des Projekts wird das Phasenmodell, mit dem Wasserfallmodell als Vorgehensmodell, verwendet. Diese Kombination liefert entscheidende Vorteile, die wichtig sind, da das Studienprojekt einen sehr straffen Zeitplan hat und dieser unbedingt eingehalten werden muss. Einer dieser Vorteile ist es, dass alle Beteiligten (Kunden und Projektmitglieder) zu jeder Zeit genau wissen, in welcher Phase sich das Projekt gerade befindet und welche Dokumente am Ende der Phase existieren müssen, da ein zurückspringen in abgeschlossene Phasen unmöglich ist. Dies bedeutet jedoch nicht, dass nicht zum Beispiel die Spezifikation zu späteren Zeitpunkten noch einmal überarbeitet werden könnte, sondern lediglich, dass man für die Überarbeitung der Spezifikation nicht noch einmal in die entsprechende Phase zurückkehrt, sondern die Überarbeitung in der aktuellen Phase durchführt. Ein weiterer Vorteil ist, dass am Ende jeder Phase, vor dem Erreichen des Meilensteins, alle Dokumente, die in dieser Phase erstellt wurden, geprüft werden, um den Meilenstein zu erreichen und damit die Phase abzuschließen, d.h. der Abschluss einer Phase bzw. das Erreichen eines Meilensteins bedeutet auch gleichzeitig, dass die erforderlichen Dokumente vorhanden und geprüft sind und den Anforderungen entsprechen.

Im nachfolgenden Abschnitt wird die Terminplanung des Projekts kurz erläutert. Danach wird dann auf die definierten Projektphasen, Arbeitspakete, Meilensteine und die Durchführung der Qualitätssicherung und Konfigurationsverwaltung eingegangen.

5.1 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 17%, Spezifikation und Entwurf



| Phase | Von / Bis | Aufwand |
|-----------------|-----------------------|---------|
| Analyse | 13.05.2009-17.07.2009 | 350 h |
| Einarbeitung | 17.07.2009-07.08.2009 | 252 h |
| Spezifikation | 07.08.2009-01.10.2009 | 588 h |
| Entwurf | 01.10.2009-01.12.2009 | 672 h |
| Implementierung | 01.12.2009-22.02.2010 | 924 h |
| Test | 22.02.2010-17.04.2010 | 588 h |
| Abnahmetest | 17.04.2010-27.04.2010 | — |
| Puffer | 17.04.2010-13.05.2010 | 294 h |

Tabelle 1: Phasen des Projekts

(inkl. Fein-Entwurf) 34%, die Codierung und der Test 41% und zuletzt der Puffer 8%. (siehe Abbildung 4)

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.

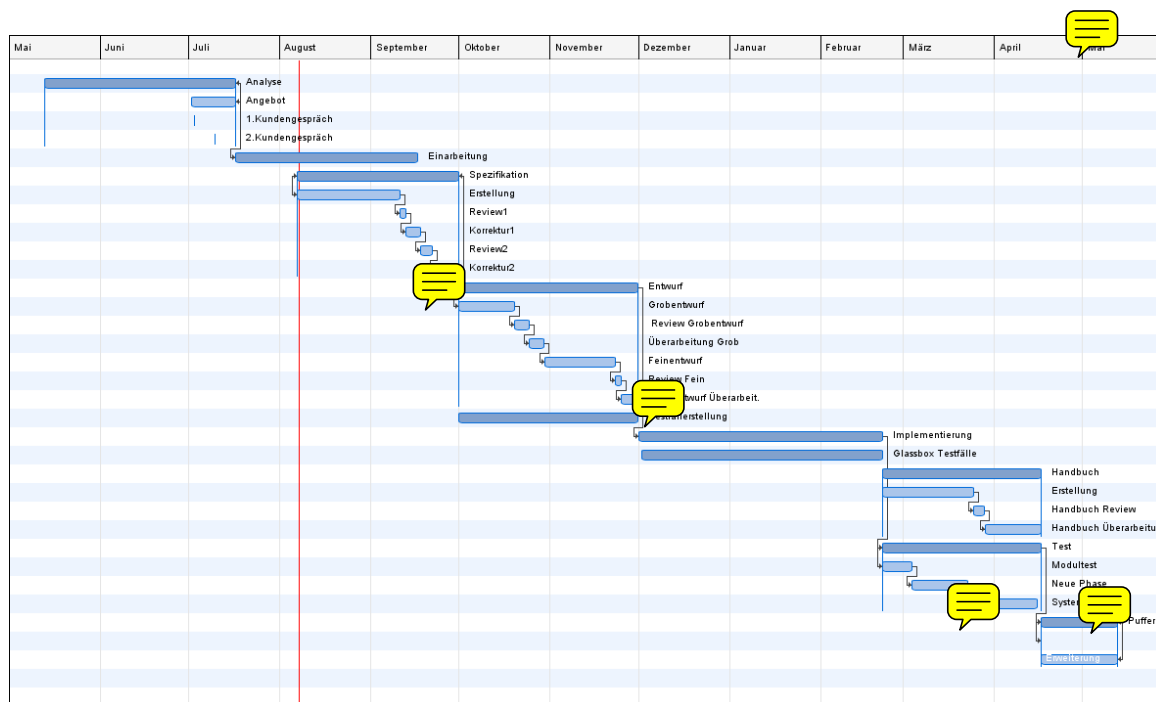
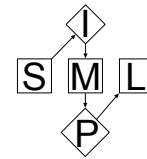


Abbildung 4: Terminplanung für die Entwicklungsphasen



5.2 Beschreibung der Phasen

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

5.2.1 Analyse

Das Ziel der Analysephase ist es, durch Kundengespräche die funktionalen und nicht-funktionalen 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.

5.2.2 Einarbeitung

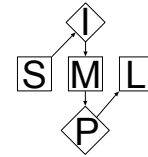
Da wenige bis gar keine Ansätze für Teilbereiche des Projekts wie z.B. die Verwendung von BPEL in wissenschaftliche Workflows existieren, müssen alle vorhandenen Ansätze evaluiert oder z.T. auch neue Ansätze geschaffen werden. Die Einarbeitungsphase dient dazu, sich in die verschiedenen Themenbereiche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

5.2.3 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.

5.2.4 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.

5.2.5 Implementierung

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

5.2.6 Test

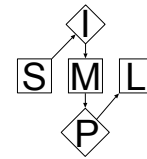
Die Testphase enthält die drei Unterphasen Modultest, Integrationstest und Systemtest. Durch den Modultest werden die einzelnen Software-Module auf ihre korrekte Funktionalität innerhalb des Moduls getestet. Im Integrationstest wird die Integration der Module zum Gesamtsystem überprüft, d.h. es wird 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. Sofern zeitlich möglich, wird 2 Wochen vor dem eigentlichen Abnahmetest, eine Demo-Version für einen Beta-Test zur Verfügung gestellt.

5.2.7 Abnahmetest

Am Ende der Testphase erfolgt die Abnahme der Projekts und aller genannten Dokumente (siehe Kapitel 5.3).

Der Kunden hat dann 10 Tage Zeit sicherzustellen, dass das von uns gelieferte Produkt seinen Anforderungen entspricht. Ist das der Fall, dann gilt das Projekt als abgeschlossen. Andernfalls wird das Projekt vom Kunden einer der folgenden Fehlerkategorien zugeordnet:

nicht einsetzbar Grundlegende Anforderungen wurden unzureichend umgesetzt, die Software ist nicht einsetzbar. Die Software kann nicht akzeptiert werden und das Projekt ist gescheitert.



mangelhaft Einige grundlegende Anforderungen wurden korrekt umgesetzt, jedoch behindern Fehler und Mängel den Einsatz der Software erheblich. Die Software kann nur mit umfangreichen Korrekturmaßnahmen akzeptiert werden.

bedingt einsetzbar Grundlegende Anforderungen wurden größtenteils umgesetzt. Fehler und Mängel behindern in Teilbereichen den Einsatz der Software. Die Software kann nach Korrektur der Teilbereiche zeitnah akzeptiert werden.

einsetzbar Alle grundlegenden Anforderungen wurden korrekt umgesetzt. Kleinere Fehler und Mängel sind vorhanden, beeinflussen aber nicht die Kernfunktionalität der Software. Das Projekt ist erfolgreich abgeschlossen und die Software wird akzeptiert.

Nach der Rückmeldung des Kunden 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.



5.2.8 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.

5.3 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 und zu erstellenden Dokumente. Sofern nicht anders angegeben, sind alle Dokumente auf Deutsch verfasst.

| Id | Bezeichnung | Aufgaben | Dokumente |
|-----------------------|---------------------|--|---|
| <i>anly#proplan</i> | Projektplan | Erstellung des Projektplans | Projektplan (PDF und L ^A T _E X) |
| <i>anly#angeb</i> | Angebot | Erstellung des Angebots für die Kunden | Angebot (PDF) |
| <i>anly#angeb#kor</i> | Angebot korrigieren | Korrektur des Angebots | Angebot 2.Version (PDF) |



| Id | Bezeichnung | Aufgaben | Dokumente |
|--|-------------------------------|---|--|
| <i>einb#tools</i> <ul style="list-style-type: none"> • #1 • #2 • #3 • #4 • #5 • #6 | Einarbeitung in Tools | Einarbeitung in: <ul style="list-style-type: none"> • Apache ODE (einb#tools#1) • Eclipse BPEL Designer • Maven • Hudson • SVN • IBM DB2 (einb#tools#6) | (interne Dokumente) Foliensammlung als Nachschlagewerk |
| <i>einb#konzept</i> <ul style="list-style-type: none"> • #1 • #2 • #3 • #4 | Einarbeitung in Konzepte | Einarbeitung in: <ul style="list-style-type: none"> • 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. |
| <i>spez#erstell</i> | Spezifikation erstellen | Erstellung der Spezifikation | (internes Dokument) Erster Entwurf des Spezifikationsdokuments |
| <i>spez#rev1</i> | 1.Review der Spezifikation | Interner Review der Spezifikation | (internes Dokument) Reviewprotokoll |
| <i>spez#kor1</i> | 1.Korrektur der Spezifikation | Korrektur der Spezifikation nach dem 1.Review | (internes Dokument) Korrigiertes Spezifikationsdokument |



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



| Id | Bezeichnung | Aufgaben | Dokumente |
|------------------------|-----------------------------|---|---|
| <i>handb#erstell</i> | Handbuch erstellen | Erstellung eines Handbuches inklusive Installationsanweisungen | (internes Dokument) Handbuch (auf Englisch und Deutsch) |
| <i>handb#rev</i> | Handbuch Review | Interner Review des Handbuches | (internes Dokument) Review Protokoll |
| <i>handb#kor</i> | Korrigieren des Handbuchs | Handbuch Korrektur | Kundenfassung Handbuch (auf Englisch und Deutsch) (PDF und \LaTeX) |
| <i>test#mod#erst</i> | Modultests erstellen | Erstellung der Modultestfälle | Testfälle (PDF und \LaTeX) |
| <i>test#mod</i> | Modultest | Durchführung des Modultests und Protokollierung der Fehler | Testprotokoll (PDF und \LaTeX) |
| <i>test#mod#kor</i> | Modultest Korrektur | Korrektur aller entdeckten Fehler anhand des Testprotokolls | Korrigierter Quellcode (Java) |
| <i>test#integ#erst</i> | Integrationstests erstellen | Erstellung der Integrationstestfälle | Testfälle (PDF und \LaTeX) |
| <i>test#integ</i> | Integrationstest | Durchführung des Integrationstests und Protokollierung der Fehler | Testprotokoll (PDF und \LaTeX) |
| <i>test#integ#kor</i> | Integrationstest Korrektur | Korrektur aller entdeckten Fehler anhand des Testprotokolls | Korrigierter Quellcode (Java) |
| <i>test#sys#erst</i> | Systemtests erstellen | Erstellung der Systemtestfälle | Testfälle (PDF und \LaTeX) |
| <i>test#sys</i> | Systemtest | Durchführung des Systemtests und Protokollierung der Fehler | Testprotokoll (PDF und \LaTeX) |



| Id | Bezeichnung | Aufgaben | Dokumente |
|---------------------|-------------------------|--|---|
| <i>test#sys#kor</i> | Systemtest Korrektur | Korrektur aller entdeckten Fehler anhand des Testprotokolls | Korrigierter Quellcode (Java) |
| <i>puff#abn</i> | Abnahme | Abnahme durch die Kunden | CD/DVD mit: Installationspaket (RAR), Dokumentation der Erweiterungen (auf Englisch und Deutsch) (PDF und L ^A T _E X), Quellcode (Java) und evtl. bereits ausgelieferte aktualisierte Dokumente |
| <i>puff#erw</i> | Erweiterung/Anpassung | Durchführung möglicher Erweiterungen und/oder Anpassungen | Alle veränderten Dokumente |

5.4 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. Beim Erreichen von externen Meilensteinen werden dem Kunden die Dokumente der Arbeitspakete des Meilensteins zur Verfügung gestellt.

| Meilenstein | zugehörige Arbeitspakete | Abnahmekriterien | Zieltermin |
|--------------------|---|--|-----------------|
| <u>Angebot</u> | <i>anly#angeb, an- ly#angeb#kor</i> | Kunde akzeptiert das Angebot. | 07. August 2009 |
| <i>Projektplan</i> | <i>anly#proplan</i> | Projektplan ist vollständig und wird von Qualitätsmanager und Projektleiter akzeptiert. | 14. August 2009 |



| Meilenstein | zugehörige Arbeitspakete | Abnahmekriterien | Zieltermin |
|---------------------------------|--|---|--------------------|
| <i>Einarbeitung</i> | <i>einb#tools#1</i> bis <i>einb#tools#6</i> , <i>einb#konzept#1</i> bis <i>einb#konzept#4</i> | Einarbeitungsberichte jedes 2er-Teams liegen vor und werden vom Projektleiter akzeptiert | 17. September 2009 |
| <i>Spezifikation 1. Version</i> | <i>spez#erstell</i> , <i>spez#rev1</i> , <i>spez#kor1</i> | Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein. | 17. September 2009 |
| <u>Spezifikation</u> | <i>spez#rev2</i> , <i>spez#kor2</i> | Der Kunde hat keine Anmerkungen und alle im Review entdeckten Fehler und Mängel sind beseitigt. | 01. Oktober 2009 |
| <u>Grobentwurf</u> | <i>grEntw#erstell</i> , <i>grEntw#rev</i> , <i>entw#kor</i> | Kunde akzeptiert den Grobentwurf. | 07. November 2009 |
| <u>Feinentwurf</u> | <i>feinEntw#erstell</i> , <i>feinEntw#rev</i> , <i>feinEntw#kor</i> | Alle im Review entdeckten Fehler und Mängel müssen beseitigt sein. | 01. Dezember 2009 |
| <u>Handbuch</u> | <i>handb#erstell</i> , <i>handb#rev</i> , <i>handb#kor</i> | Der Kunde hat keine Anmerkungen und alle im Review entdeckten Fehler und Mängel sind beseitigt. | 17. April 2010 |
| <i>Modul implementierung</i> | <i>code#module</i> | Alle spezifizierten Use-Cases sind innerhalb der Module implementiert. | 08. Februar 2010 |
| <i>Erweiterungen</i> | <i>code#BPELdes</i> , <i>code#Apache</i> | Alle spezifizierten Erweiterungen sind implementiert. | 08. Februar 2010 |

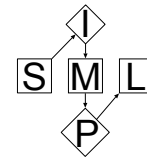


| Meilenstein | zugehörige Arbeitspakete | Abnahmekriterien | Zieltermin |
|-------------------------|--|--|------------------|
| <i>Integration</i> | <i>code#integr</i> | Module wurden erfolgreich zu einer lauffähigen Software integriert. | 22. Februar 2010 |
| <i>Modultest</i> | <i>test#mod#erst,</i> <i>test#mod,</i> <i>test#mod#kor</i> | 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. | 04. März 2010 |
| <i>Integrationstest</i> | <i>test#integ#erst,</i> <i>test#integ,</i> <i>test#integ#kor</i> | Alle erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert. | 23. März 2010 |
| <i>Systemtest</i> | <i>test#sys#erst,</i> <i>test#sys,</i> <i>test#sys#kor</i> | Alle aus der Spezifikation erstellten Testfälle wurden ausgeführt, deren Verlauf protokolliert und der Code anhand der Ergebnisse korrigiert. | 17. April 2010 |
| <u>Abnahmetest</u> | <i>puff#abn</i> | Der Kunde akzeptiert die Software. | 27. April 2010 |



5.5 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 (“Code Conventions for the Java Programming Language”) 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 oder werden noch einem Review unterzogen. 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 5.

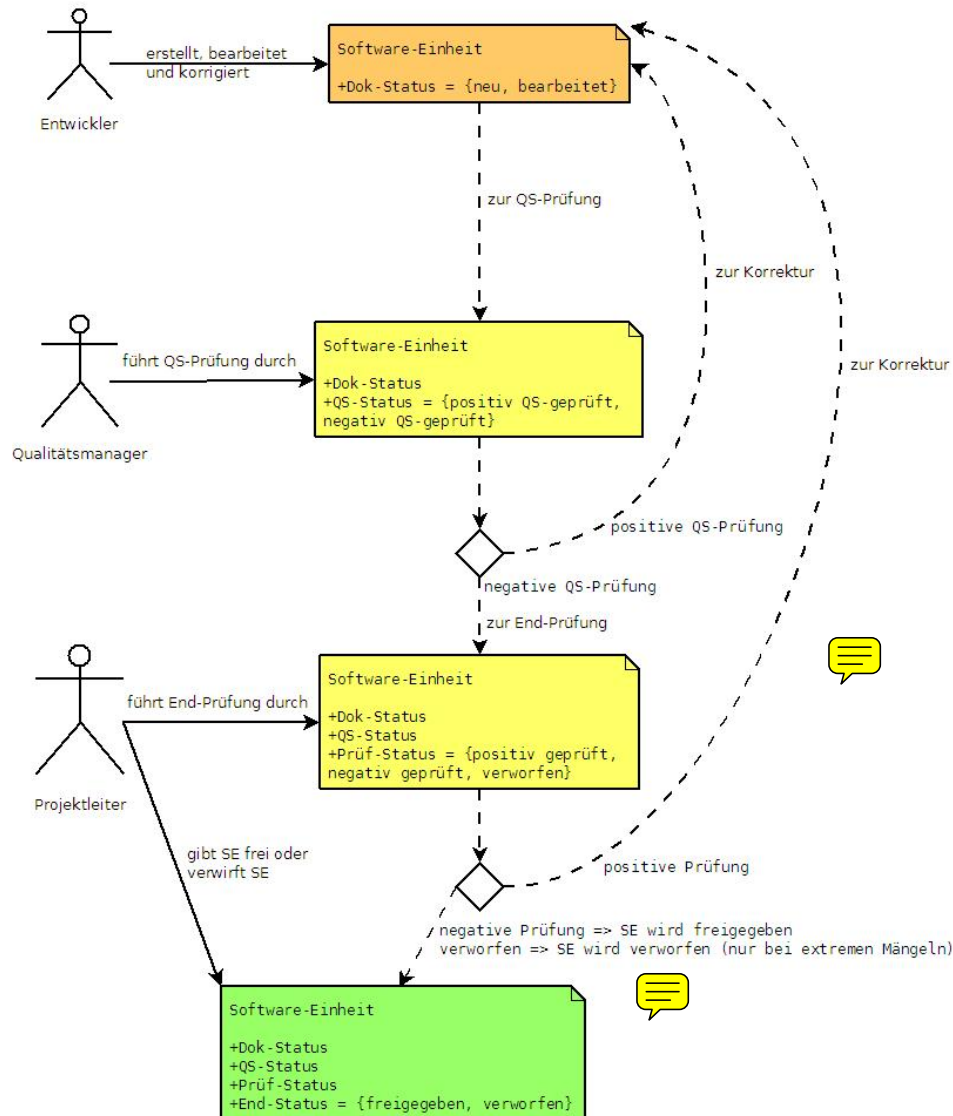
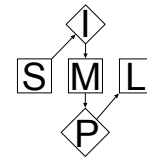


Abbildung 5: Prüfungsablauf für die Qualitätssicherung

Die Konfigurationsverwaltung ist ein wichtiger Bestandteil der Infrastruktur eines Projekts. In diesem Projekt wird zur Realisierung das Konfigurationsverwaltungssystem *Subversion* genutzt. Abbildung 6 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 7 wird der Entstehungsprozess von SE aufgezeigt und welche Umgebungen dabei durchlaufen werden.

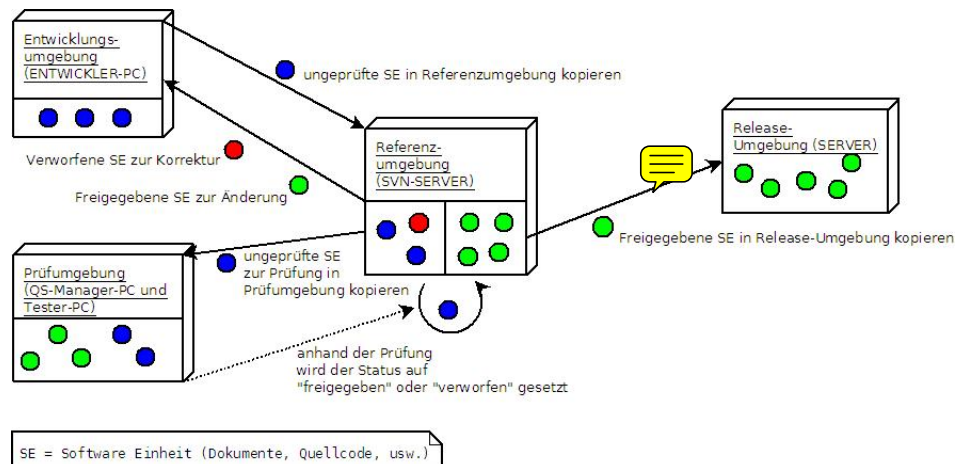


Abbildung 6: Das grundlegende Modell der Konfigurationsverwaltung

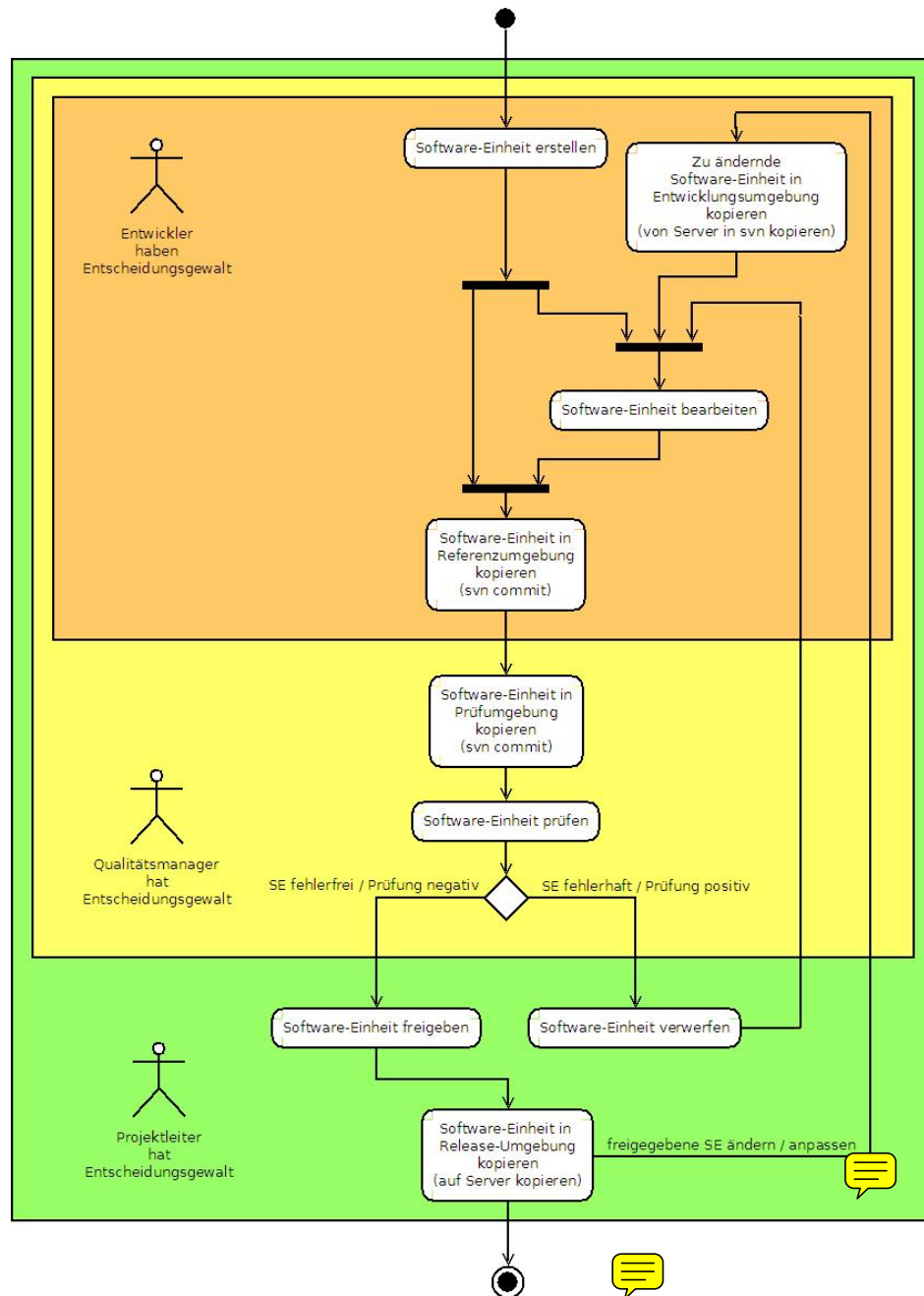
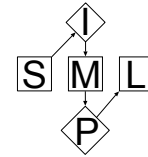


Abbildung 7: Entstehungsprozess für Software-Einheiten



5.5.1 Durchführung der Reviews

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 ihre Befunden einer Fehlerkategorie zu.

Spätestens drei Tage vor Beginn des Reviews schickt der Moderator Einladungen, mit dem zu begutachtenden Dokument, an die Gutachter des Reviews. 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. 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.

6 Unsere Konditionen

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

6.1 Kostenschätzung

Unsere internen Berechnungen nach der Funktion-Point-Methode ergab 268 Adjusted Function Points (AFP). Aus einer Referenztafel 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 hypothetisch und wurden nur zu Planungszwecken durchgeführt.

6.2 Risiken

Während des Projekts können Ereignisse auftreten, die das Projekt behindern oder gar scheitern lassen. In der folgenden Tabellen werden einige dieser Risiken, ihre Eintrittswahrscheinlichkeiten, die dadurch verursachten Kosten und geeignete Gegenmaßnahmen, die den Eintritt oder die Kosten der Risiken senken, aufgelistet.





| Risiko | % | Kosten | Gegenmaßnahmen |
|--|----|--------|--|
| Mitarbeiter fällt aus  | 15 | 2 MM | Wöchentliche Treffen (Informationsaustausch), Ernennung von Stellvertretern und Erstellung von internen Dokumentationen |
| 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  |
| Chaotische Änderungen der Dokumente | 10 | 2 MM | Ausgiebige Dokumentprüfung durch Qualitätssicherung und Projektleiter Konfigurationsverwaltung mit Subversion |

7 Rechtliche Rahmenbedingungen

In diesem Kapitel werden die rechtlichen Rahmenbedingungen des Projekts erläutert.

7.1 Gewährleistung

Die Gewährleistungsfrist beträgt 6 Monate  und beginnt mit der Auslieferung der Software.  Der Kunde verpflichtet sich die Software innerhalb von 10 Tagen nach der Auslieferung zu prüfen. Innerhalb der Gewährleistungsfrist werden nur Korrekturen von entdeckten Fehlern durchgeführt, jedoch keine zusätzlichen Erweiterungen implementiert. Sollten zusätzliche Erweiterungen gewünscht werden, müssen diese neu ausgehandelt werden.



7.2 Pflichten des Auftragnehmers

Der Auftragnehmer verpflichtet sich dazu alle im Angebot aufgeführten Tätigkeiten nach Bestem Wissen und Gewissen auszuführen. Weiterhin verpflichtet sich der Auftragnehmer den Auftraggeber über den Fortschritt des Projekts in regelmäßigen Abständen zu unterrichten, spätestens aber nach Erreichen eines Meilensteins, der bei Auftreten von Problemen, welche den Verlauf des Projekts negativ beeinflussen.

7.3 Pflichten des Auftraggebers

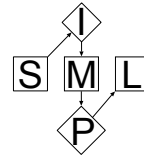
Der Auftraggeber verpflichtet sich die nötige Infrastruktur (Räume, Server und Arbeitsgeräte) während der Projektlaufzeit zur Verfügung zu stellen. Darüber hinaus ist der Auftraggeber dazu verpflichtet für den weiteren Informationsaustausch zur Verfügung zu stehen und das Projektteam über längere Nichterreichbarkeit rechtzeitig zu informieren. Nach Erhalt der Software verpflichtet sich der Auftraggeber die Software in einem Abnahmetest zu prüfen und den Auftragnehmer über die erhaltenen Ergebnisse zu informieren. Sollten beim Abnahmetest Mängel oder Fehler entdeckt worden sein, so muss der Auftraggeber dem Auftragnehmer die Möglichkeit zur Korrektur einräumen. Sollte der Abnahmetest ohne Beanstandungen abgeschlossen werden, so ist der Auftraggeber verpflichtet die Software abzunehmen und den Erhalt, sowie die Korrektheit der Software zu quittieren.

7.4 Nutzungs- und Urheberrecht

Das Projekt und alle erstellten Software-Einheiten stehen unter der Apache 2.0-Lizenz (<http://www.apache.org/licenses/LICENSE-2.0>).

7.5 Gültigkeit des Angebots

Das Angebot in dieser Form ist, ab Erhalt, eine Woche (7 Tage) gültig.



Vertrag zur Auftragserteilung und Auftragsannahme

Durch die nachfolgenden Unterschriften von Auftraggeber und Auftragnehmer erklären sich beide Parteien dazu bereit, den durch das Angebot festgelegten Rahmen für das Projekt, zu akzeptieren. Darüber hinaus bestätigt die Unterschrift des Auftraggebers die Erteilung des Auftrags an den Auftragnehmer. Die Unterschrift des Auftragnehmers bestätigt die Annahme des Auftrags des Auftraggebers.



Stuttgart, den

Ort, Datum

Bevollmächtigter Vertreter des Auftraggebers

Bevollmächtigter Vertreter des Auftragnehmers