SS 2006
Software Engineering (SE)
Artefaktenbeschreibung

Übungsreihe Software Engineering 1 und 2
Projektpraktikum im betrieblichen Umfeld (PPR)

Barbara Schuhmacher
Version 1.0, 07.04.2006

Institut für Softwaretechnik und Interaktive Systeme, Quality Software Engineering, http://qse.ifs.tuwien.ac.at

Inhaltsverzeichnis

Allgemeines

1 Einleitung

Erwartungen von Projektmanagern in industriellen Projekten an Softwareingenieure, die frisch von der Universität kommen fallen typischerweise in folgende Bereiche:

- Gutes Fachwissen über den Stand der Technik in Hardware und Betriebssystemen
- Gutes theoretisches Hintergrundwissen in der Informationstechnik
- Die Fähigkeit neue Methoden und Werkzeuge, die diese Methoden unterstützen, in einigen Wochen zu verstehen und anzuwenden
- Überdurchschnittliches Verständnis für aktuelle wesentliche Paradigmen in der Softwaretechnik und Informationstechnologie
- Umfassendes Verständnis für Prozesse in der Software-Entwicklung von der Analyse bis zur Wartung, inklusive Projektmanagement, Qualitätssicherung und Teamarbeit (siehe Abb. 1)
- Die Fähigkeit und Bescheidenheit jede Aufgabe im Entwicklungsverlauf zu übernehmen, auch wenn ihr Prestigewert nicht hoch erscheint, etwa Wartung oder Dokumentation
- Die Fähigkeit und Bescheidenheit jede Aufgabe im Team zu übernehmen und dabei sowohl den eigenen Beitrag zum Projekt zu verstehen wie auch den Beitrag der anderen Rollen zum Projekt zu respektieren
- Verständnis und Respekt für den Wert tatsächlicher Projekterfahrung von geeichten Software-Ingenieuren

Die ersten vier Punkte in dieser Liste sind konzentriertes Ziel für entsprechende Lehrveranstaltungen im aktuellen Informatiklehrplan, wodurch sich der Software Engineering LU / UE Spielraum eröffnet, Methoden und Erfahrungen mit Projekttauglichkeit im Team zu vermitteln.

2 Ziele

Ziel der Software Engineering LU / UE ist die praktische Vermittlung von bewährten Methoden und Techniken der Softwareentwicklung unter Verwendung von relationalen Datenbanken. Die Übungsreihe SE 1 zielt darauf ab, die Methoden der Softwareentwicklung anhand eines kleinen Projektes zu erlernen und diese Fähigkeiten in SE 2, PPR, Praktika usw. in einem mittelgroßen Projekt mit kommerzieller Aufgabenstellung für einen realen Kunden anzuwenden und zu vertiefen.
Eine wichtige Rolle in der SE Lehrveranstaltungsreihe spielen moderne Technologien, die eine hohe Verwendung in der Praxis aufweisen. Aus diesem Grund liegt - sowohl bei SE 1 als auch bei SE 2 - der Fokus auf der Umsetzung eines Software-Projektes unter Verwendung aktueller Technologien und Methoden. Bei der Implementierung können z.B. J2EE, .NET oder Web Services verwendet werden. Außerdem besteht die Möglichkeit mobile Applikationen mit J2ME oder .NET CF zu entwickeln. Für die Analyse/Design Phase werden Tools von Rational zur Verfügung gestellt.
Darüber hinaus besteht die Möglichkeit im Rahmen der SE-Lehrveranstaltungen MCP-Prüfungen (Microsoft Certified Professional) zu Sonderkonditionen zu absolvieren. Genauere Informationen gibt es auf der LVA-Webseite.
Die SE LU / UE ist eine Laborübung für mehr als 200 Studenten pro Jahr, denen etwa 15 Tutoren zur Verfügung stehen. Die meisten Anfänger sind etwa 20 Jahre alt, haben einen relativ modernen und leistungsfähigen Computer oder Laptop zu Hause, können in einer strukturierten bzw. objektorientierten Programmiersprache gut bzw. mäßig programmieren und haben keinerlei Erfahrung mit Software-Entwicklung im Team oder mit größeren Systemen.

Die große Zahl Studenten und die begrenzten Labor- und Betreuungskapazitäten bedingen einen pragmatischen Ansatz in der Übungsplanung.

Wahl der Methoden. Die zu vermittelnden SE-Methoden müssen miteinander auch wirklich gut funktionieren. Neu hinzukommende Methoden brauchen realistische Beispiele, die durch geeignete Werkzeuge unterstützt werden.

Wahl der Werkzeuge. Werkzeuge sollen in der Entwicklung die Umsetzung der Denkmodelle in funktionierende Systeme unterstützen. Die Werkzeuge müssen daher möglichst leicht und schnell erlernbar sein und dann in den Hintergrund treten. Werkzeuge, die durch umständliche Installation und Bedienung oder häufige Fehler und Abstürze die Aufmerksamkeit ungebührlich auf sich ziehen, sind nach Möglichkeit zu vermeiden.

Rollen im Team. Die Gruppierung von Aufgaben und Aktivitäten um spezifische Rollen – wie Teamkoordinator, Technischer Architekt, Programmierer, Tester oder Dokumentenbeauftragter – tragen viel zum Verständnis für den Beitrag dieser Rolle zum Projekt bei und erleichtern durch gut definierte Verantwortlichkeiten die Projektorganisation.

Arbeitsorganisation auf Teamebene und auf individueller Ebene. Die meisten Studenten sind an das individuelle Lösen von technischen Problemen gewöhnt. Nur wenige haben Erfahrungen mit der effektiven Durchführung von Besprechungen, Verhandlungen oder dem Management von Projekten. Die Herausforderungen in den Teamprojekten erfordern neben technischer Kompetenz auch ein Minimum an gemeinsamem Handeln und der Abstimmung der individuellen Vorgehensweisen. Die häufigsten Ursachen für das Scheitern eines Teamprojekts liegen im Bereich vernachlässigter Kommunikation bzw. Arbeitsorganisation.

Diese Vorgehensweise bringt wirklichen Bedarf für Projektmanagement und Arbeitsorganisation sowie klaren Verantwortlichkeiten im Team mit sich.[1]

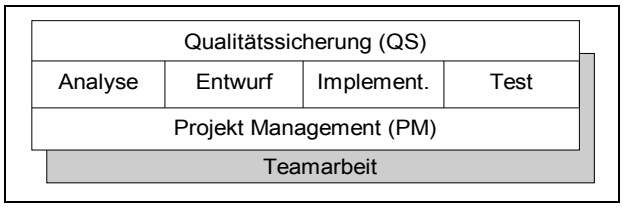


Abb. 1: SE-Methoden im Kontext

Im Lauf der Übung sollen die Teilnehmer Bestandteile und Ablauf des Software-Entwicklungsprozesses in allen Phasen der Software-Herstellung anhand realitätsähnlicher Aufgaben kennen lernen.

1 Das Vermitteln von SE-Methoden anhand idealisierter kleiner Spielzeugbeispiele ist zwar auch möglich, die Praxistauglichkeit des so erworbenen Wissens ist allerdings bisweilen nicht im erwünschten Ausmaß gegeben.

Artefaktenbeschreibung

1 Allgemeiner Aufbau eines Dokuments

Hier wird der allgemeine Aufbau eines Dokuments beschrieben. Es ist absolut notwendig, dass die unten angeführten Anforderungen eingehalten werden.

Dieser Aufbau muss, bei jedem zu erstellenden Dokument, eingehalten werden.

1.1 Deckblatt

Jedes Dokument muss ein Deckblatt enthalten, welches folgende Inhalte vorweist:

- Titel des Dokumentes (z.B.: Architekturdokument)
- Kurzbeschreibung des Inhalts
- Autor des Dokumentes
- Die Gruppe, welcher der Autor angehört

1.2 Kopfzeile

In der Kopfzeile wird die Übung oder Vorlesung angegeben, für welche das Dokument erstellt wird.

1.3 Fußzeile

In der Fußzeile müssen der Name des Dokuments und die Seitennummerierung (z.B.: Seite 1 von 10) vorhanden sein. Weiters können hier noch Erstellungsdatum, Gruppennummer und Autorenkürzel angegeben werden.

1.4 Änderungshistorie

Wird ein Dokument überarbeitet, so muss dies im Dokument ersichtlich sein.

Nr.	Datum	Autorenkürzel	Änderung
1	01.10.2005	MM	Dokument erstellt
2	02.10.2005	MM	Den Use-Case „Kunde anlegen“ und dessen Beschreibung aufgenommen
3	05.10.2004	LH	Den Beschreibung des Use-Case „Kunde anlegen“ überarbeitet

2 Projektauftrag

Der Projektauftrag ist die Grundlage für die Durchführung des Projektes. Er enthält vor allem die Problemstellung und welche Ziele mit dem Projekt verfolgt werden. Er beinhaltet die Vereinbarungen, die mit dem Auftraggeber getroffen werden, d.h. es wird angegeben, welche Komponenten abzuliefern sind, welche Erweiterungen möglich sind. Das Projekt soll aber bereits eine deutliche Abgrenzung aufweisen, um auch einen realistischen Terminplan und eine grobe Aufwandsabschätzung zu ermöglichen, was auch bereits im Projektauftrag festgehalten werden soll. Weiters müssen hier auch die Schnittstellen, die Architektur, die Struktur sowohl des Programms als auch der Datenbank angeführt werden.

2.1 Problembeschreibung und Zweck des Projektes

Hier wird kurz zusammengefasst, welche Probleme vorhanden sind, bzw. es wird der derzeitige IST-Zustand wiedergegeben. Daraus ergeben sich dann der wesentliche Inhalt und Zweck des Produktes. Es soll hier beschrieben werden, welche Funktionalität das Produkt haben soll, in welchem Umfeld es eingesetzt wird, wer es benutzt usw.

Bsp.:
Das Unternehmen besteht aus einer Zentrale in Wien und 8 weiteren Filialen, welche in den Hauptstädten der übrigen Bundesländer positioniert sind. Derzeit werden in den Filialen neue Kunden in einem EXCEL-Sheet erfasst. Dieses EXCEL-Sheet wird am Ende des Tages via Mail nach Wien gesandt; dort wird dann jedes einzelne File in die zentrale Datenbank eingespielt. Es soll ein Softwarelösung erstellt werden, welche es ermöglicht über eine Webapplikation, die Kundendatenpflege direkt in der zentralen Datenbank abzuwickeln.

2.2 Ziele des Projektes

In diesem Abschnitt des Dokuments werden die Ziele des Projektes, bzw. jene Personen für die das Projekt entwickelt wurde (z.B.: die Anwender).

Wir unterscheiden folgende Ziele:

- betriebswirtschaftliche Ziele

  Bsp.: Durch den Einsatz der neu entwickelten Software, soll die Erfassung aller relevanten Kundendaten für eine Bestellung von 15min auf 9min reduziert werden. Damit ergibt sich bei 3000 Bestellungen pro Monat eine Einsparung von 300 Arbeitsstunden.

- funktionale Ziele

Bsp.: Durch das Speicher aller Daten in einer zentralen Datenbank und den direkten Zugriff darauf mittels einer Webapplikation, ist es möglich den Kunden besser zu unterstützen, auf Kundenwünschen effizienter reagieren zu können, bzw. die Entwicklung der Kundenwünsche besser abschätzen zu können.

- Marketing Ziele

  Bsp.: Durch die rasche und qualitativ hochwertige Entwicklung des Produktes, schätzen uns unsere Kunden und man wird langfristiger Partner der Kunden.

- soziale Ziele

  Bsp.: Dem Kunden wird der Arbeitsalltag, durch die Verwendung unserer Software, erleichtert, da er seine Bestellungen, rasch und bequem durchführen kann.

- Umfeldziele: Welche Ziele können durch den Einsatz des Produktes im näheren Umfeld noch erreicht werden.

  Bsp.: Image, Stellenwert, bestmögliche Note auf die UE, usw..

- Zielgruppen

  Bsp.: Diese Software dient Klein- und Mittelbetrieben um Bestellvorgänge jederzeit rasch und bequem durchführen zu können.

2.3 Abgrenzung

Hier soll klar beschrieben werden, was das Produkt nicht kann.

Bsp.: Unsere Kunden haben keine Möglichkeit, ihre Lagerverwaltung bzw. eine Optimierung der Lieferzeit mit dieser Software abzuwickeln.

2.4 Vertragspartner

Hier wird festgehalten, welche Partner am Projekt mitarbeiten und wie die Verantwortungsbereiche unter den Vertragspartnern verteilt sind. Meist handelt es sich hier um 2 Vertragspartner, den Auftraggeber und den Auftragnehmer.

Der Auftraggeber formuliert die Anforderungen an das Projekt. Hier werden auch die Adresse des Auftraggebers und die Ansprechpersonen vermerkt.

Die Auftragnehmer sind jene Personen, welche den Projektauftrag des Auftraggebers annehmen. Auch hier werden bestimmte Rollen und Ansprechpartner vermerkt. Wir unterscheiden folgende Rollen:

- Organisatorischer Projektleiter = Teamkoordinator (TK): hat die Verantwortung für das Projekt und ist unter anderem für folgende Aufgaben zuständig:

  - Projektplanung
  - Projektsteuerung
  - Erstellung eines Projektauftrages
  - Teamkoordination
  - Monatsberichte
  - Direkter Ansprechpartner für den Auftraggeber
  - ...

- Stellvertretender organisatorischer Projektleiter: dient als Backup für den Projektleiter und hat, bei dessen Abwesenheit, die Aufgabe den Projektleiter in allen Angelegenheiten zu vertreten.

- Technischer Archtitekt (TA): hat die technische Verantwortung und ist unter anderem für folgende Aufgaben verantwortlich:
  - Erstellung und Entwurf der Systemarchitektur
  - Erstellung der Codierungsrichtlinien
  - Qualitätssicherung während der Impplementierungsphase
  - Erstellung von Design-Dokumenten
  - ...

- Stellvertretender technischer Architekt: dient als Backup für den Architekten und hat, bei dessen Abwesenheit, die Aufgabe den Architekten in allen Angelegenheiten zu vertreten.

- Dokumentenbeauftragter: ist für alle Bereiche der Dokumentation zuständig und hat unter anderem folgende Aufgaben:
  - Erstellung der Dokumentationsrichtlinien
  - Einhaltung der Dokumentationsrichtlinien prüfen
  - Vollständigkeit der Dokumententagebücher prüfen
  - ...

- Testbeauftragter: ist für die Erstellung der Testpläne bzw. auch der Testfälle zuständig. Gemeinsam mit den Testern wird an der Einhaltung der Testpläne bzw an der Durchführung der Tests und an der Erstellung der Testberichte gearbeitet.

Am besten lässt sich die Rollenverteilung in einer Tabelle vermerken:

Rolle	Name	Matr.-Nr.	Kennz.	Email	Telefon
TK	Max Muster	01020455	526	m.m@aon.at	01 546 345
TA	Hannes Muster	04022344	526	h.m@chello.at	
...					

2.5 Lieferkomponenten

Dem Kunden wird bei Projektabschluss Folgendes übermittelt:

- voll funktionale Software
- Designprotokolle
- Benutzerhandbuch
- Anforderungsdokumente

Hier kann auch eine Priorisierung der einzelnen Module vorgenommen werden (Must be, Would be good, Luxury)

2.6 Weitere Komponenten (nicht im Lieferumfang)

- Dokumente, welche zur Erstellung der Software benötigt wurden (ER-Diagramme, Klassendiagramme, usw.)
- Tools, die zur Erstellung der Software notwendig war (Eclipse, Rational XDE, ...)
- Teaminterne Dokumente

2.7 Work-Breakdown-Structure

Die Work Breakdown Structure (WBS), d.h. die Organisation der einzelnen Arbeitsinhalte und -pakete, wird mit Hilfe eines Projektstrukturplanes (PSP) dargestellt. Der Projektstrukturplan (PSP) entsteht bei der Planung eines Projektes. Der Projektstrukturplan enthält die Struktur der im Projekt zu bewältigenden Aktivitäten. Der Projektstrukturplan soll die Komplexität des Projektes in überschaubare Arbeitspakete aufteilen und dient somit der aufwands- und termingerechten Abwicklung eines Projektes.

Hier wird das Problem hierarchisch in Haupt- und Teilaufgaben gegliedert. Die Teilaufgaben werden mit einer Kosten-, bzw. Aufwandschätzung versehen. Damit erhält man rasch einen Überblick über den Status des Projektes, indem man einen Soll/Ist-Vergleich erstellt.

Bsp:

Nr.	Struktur	Aufgaben	Beginn	Ende	Aufw./Kosten in Tagen
1	**Projektstart**		01.11.2005	01.11.2005	**0.50**
1.1		Projekt-Kickoff-Meeting	01.11.2005	01.11.2005	0.50
1.2					
2	**Analysephase**		**02.11.2005**	**10.12.2005**	**44.50**
2.1		Anforderungen erheben	02.11.2005	25.11.2005	15.00
2.2		Use-Case erstellen	10.11.2005	05.12.2005	15.50
2.3		Anforderungen reviewen	05.11.2005	01.12.2005	5.00
2.4		Use-Case reviewen	15.11.2005	06.12.2005	5.00
2.5		Änderungen aus Review dokumentieren	06.12.2005	09.12.2005	4.00
2.6		Anforderungsdokument präsentieren	10.12.2005	10.12.2005	0.50
3	**Designphase**		10.12.2005	31.12.2005	
3.1		....	...	...	

2.8 Ecktermine

Hier werden die relevanten Termine für den Kunden festgelegt und beschrieben.

Bsp:

Tätigkeit/Phase	Kurzbeschreibung	Termin
Anforderungsanalyse	Intern und beim Kunden	02.10.2005 – 01.12.2005
Abstimmung der Anforderungs-analyse	Mit dem Kunden	05.12.2005
Entwurfsphase		Bis 15.01.2006
Implementierungs- und Testphase	Erstellen der Applikation und durchführen des Integrations- und Systemtests	Bis 31.03.2006
Auslieferung der Betaversion an den Kunden	Inkl. des Installationshandbuches	01.04.2006
Feinschliff der Applikation	Beheben der vom Kunden gemeldeten Fehler	01.05.2006
Projektabschluss	Auslieferung der Software	10.05.2006

2.9 Aufwandsschätzung

Diese Schätzung wird in 2 Schätzungen unterteilt:

- **Personalaufwand:** z.B. wieviele Personentage braucht man zum Durchführen des Projektes
- **Sachmittelaufwand:** z.B. was kosten die notwendigen Tools, die notwendige Hardware, ...

2.10 Grobe Datenbankbeschreibung

Hier wird ein erstes Konzept Datenbank grob beschrieben um einen Überblick über den Umfang des Projektes zu bekommen. Es wird aufgeschlüsselt welche Tabellen verwendet werden und auch einen ersten Entwurf, welche Attribute zur Problemlösung benötigt werden.

3 Anwendungsfallbeschreibung

Dieses Dokument spiegelt das Gesamtdokument zur Anforderungsbeschreibung wieder, sprich von der Aktorenliste bis zur Anwendungsfallbeschreibung im Ereignis/Aktion-Stil.

3.1 Aktorenliste

In diesem Teil des Dokuments wird genauestens beschrieben,

- welche Aktoren auftreten,
- was die Aktoren für Rechte haben, sprich welche Anwendungsfälle darf er durchführen und welche nicht,

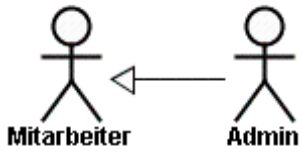
Bsp.:

Name	Rechte	Anmerkungen
Admin	Uneingeschränkten Systemzugriff	Dieser Aktor ist für die Verwaltung und Konfiguration des Systems zuständig
Mitarbeiter	Darf die Anwendungsfälle „Kunde anlegen“ und „Kunde bearbeiten“ durchführen.	Dieser Aktor ist für die Kundenverwaltung zuständig, und hat sonst keine Rechte auf andere Anwendungsfälle.

3.2 Aktorenhierarchie

Ist ein Aktor eine Spezialisierung/Generalisierung eines anderen Aktors, dann muss das hier beschrieben werden.

Bsp.:



Ein Admin ist ein Mitarbeiter, welcher zusätzlich zu den Rechten des Mitarbeiters noch spezielle Rechte hat.

3.3 Übersicht über alle Anwendungsfälle

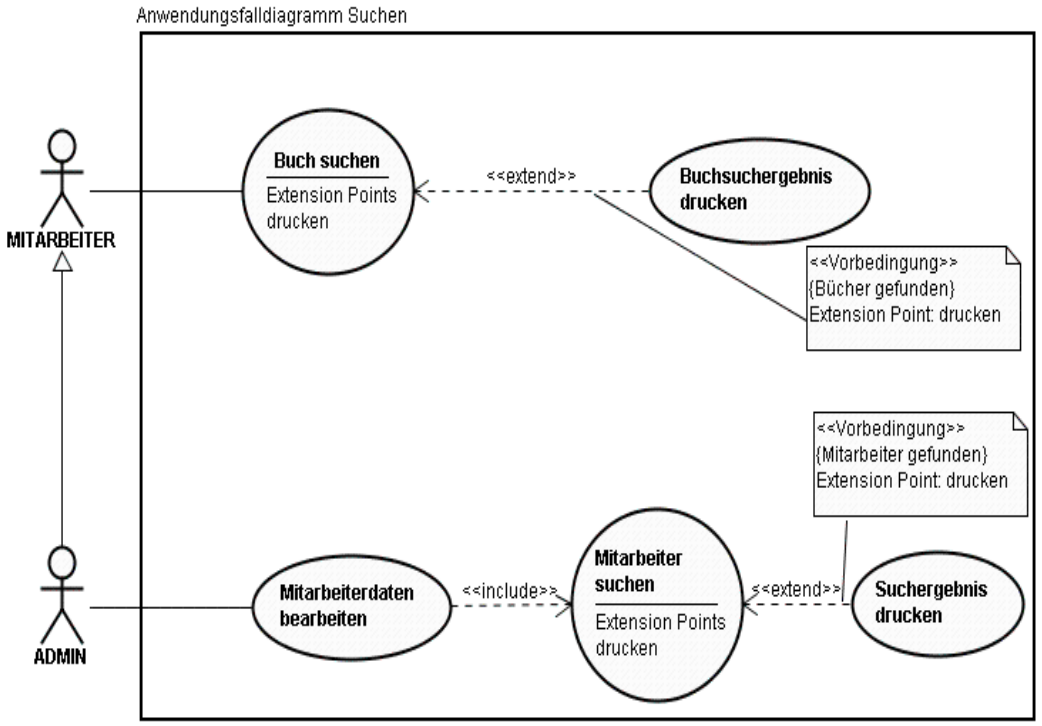
Hier werden alle Anwendungsfälle in einer Übersichtstabelle dargestellt. Weiters sollte man sich hier schon überlegen, welche Anwendungsfälle funktional zusammengehören und diese Gemeinsamkeiten in einem Paket zusammenfassen.

Paket	Kürzel	Titel	Priorität
User	Create_U	Benutzer anlegen	1
User	Search_U	Benutzer suchen	1
User	Del_U	Benutzer löschen	2

Die Priorität gibt an, wie wichtig der Anwendungsfall für das System, bzw. für den Kunden ist (z.B.: 1- Hoch, 2-Mittel, 3-Niedrig). So lassen sich die Hauptanwendungsfälle herauskristallisieren.

3.4 Anwendungsfalldiagramme

In diesem Teil des Dokuments, werden die UML Anwendungsfall-Diagramme gezeichnet.



Die Vorbedingungen und die entsprechenden extension Points werden als Notiz an die <<extends>> Beziehung angehängt

3.5 Anwendungsfall Beschreibung

Hier werden die einzelnen Anwendungsfälle textuell beschrieben. Diese Beschreibung muss nach der folgenden Struktur erarbeitet werden.

- **„Identification summary“:**
  Titel des Anwendungsfalles
  Kurzbeschreibung des Anwendungsfalles
  Welche **Aktoren** sind beteiligt
  Erstellungsdaten des Anwendungsfalls (Ersteller, Datum, Version)

- **„Flow of events“:**
  Hauptszenario – was soll abgebildet werden
  Alternativszenario - gibt es Alternativabläufe, wo sind die Abweichungen?
  Fehlersituationen – was passiert im Fehlerfall, Systemzustand
  Vorbedingung - Voraussetzung für erfolgreiche Ausführung
  Nachbedingung - Systemzustand nach erfolgreicher Ausführung

- **„Non-functional constraints“**
  Bemerkungen, Angaben über Häufigkeit, usw.

Bsp.:

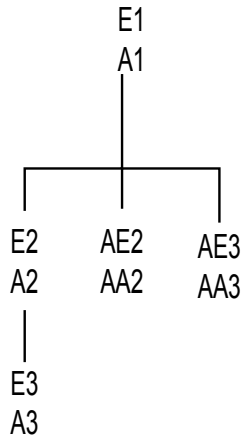
- **Identification summary“**
  Titel: Buch suchen
  Kurzbeschreibung: Der User sucht nach einem bestimmten Buch
  Aktoren: Der Anwender
  Erstellungsdaten: Max Mustermann, 2. 10. 2004, V 1.0
- **„Flow of Events“:**
  Beschreibung: Der Anwender gibt die Suchkriterien für ein Buch ein (Titel, Autor, Verlag, Kategorie). Das System zeigt die zu den Kriterien gefundenen Bücher an. Der Anwender wählt einen Datensatz aus und die Liste mit den Suchergebnissen wird geschlossen.
  Alternativszenario: Kann das System kein Buch zu den angegebenen Kriterien finden, es wird eine Fehlermeldung ausgegeben.
  Fehlersituation: Kann der Connect zur Datenbank nicht aufgebaut werden, dann wird noch 2x versucht den Connect zur DB herzustellen. Sind alle 3 Versuche erfolglos wird eine Fehlermeldung ausgegeben.
  Vorbedingung: Das System läuft
  Nachbedingung: Die Daten des gefundenen Buches werden am Bildschirm angezeigt

- **„Nonfunctional constraints“:**
  Der User sollte nicht zu lange auf das Suchergebnis warten müssen (max. 10 sek)
  Der Anwendungsfall „Buch suchen“ wird sehr häufig in einer Bibliothek Verwendung finden.

3.6 E/A Diagramm pro Anwendungsfall

Dies ist eine eigene Art der Anwendungsfallbeschreibung. Sie ist der textuellen Beschreibung ähnlich, jedoch werden hier auch die Ereignisse (E), welche Aktionen (A) auslösen, angegeben und beschrieben.

Bsp.



E1) Der Anwender gibt Suchkriterien für ein Buch ein. (Titel, Autor, Kategorie)
A1) Das System sucht nach Büchern, die den eingegebenen Suchkriterien entsprechen.
E2) Es existieren Bücher, die den angegeben Suchkriterien entsprechen
A2) Die Daten der Bücher werden angezeigt
AE2) Es wurde kein Buch mit den angegeben Suchkriterien gefunden
AA2) Fehlermeldung ausgeben
AE3) Mit diesen Kriterien wurden zu viele Bücher gefunden
AA3) Der Benutzer wird aufgefordert, die Suchkriterien zu verfeinern
E3) Der Anwender wählt einen der angezeigten Datensätze aus
A3) Die Liste mit den Suchergebnissen wird geschlossen und der ausgewählte Datensatz eventuell als Parameter an einen weiteren Anwendungsfall übergeben

4 Projektstatusbericht

In diesem Dokument wird ein Überblick über den aktuellen Projektstatus erstellt. Hier wird beschrieben welche Aktivitäten durchgeführt wurden, ob das Projekt im Zeitplan ist oder nicht, welche Probleme aufgetreten sind und wie diese behoben wurden. Auch können hier eventuell offene Punkte beschrieben werden und auch angegeben werden warum es hier zu Verzögerungen gekommen ist.

Bsp:

Projektstatusbericht vom 15.10.2005
Bericht zur Phase 1 für das Projekt <Name des Projektes>

1. **Erledigte Punkte**
   Folgende Punkte wurden bis zum 14.10.2005 erledigt
   - Teamfindung
   - Rollenverteilung
   - Anforderungsdokument erstellen
   - Domänenmodell erstellen
   - UI-Prototyp erstellen

2. **Zeitplan**
   Derzeit bewegt sich das Projekt voll im Zeitplan.

3. **Probleme**
   Es wurde erkannt, dass die gestellten Anforderungen nicht im gewünschten Zeitplan durchzuführen sind. Nach Absprache mit dem Tutor, wurden 3 Anwendungsfälle gestrichen, die Dokumentation dementsprechend angepasst, und somit das Problem gelöst.

5 Projektplan grob

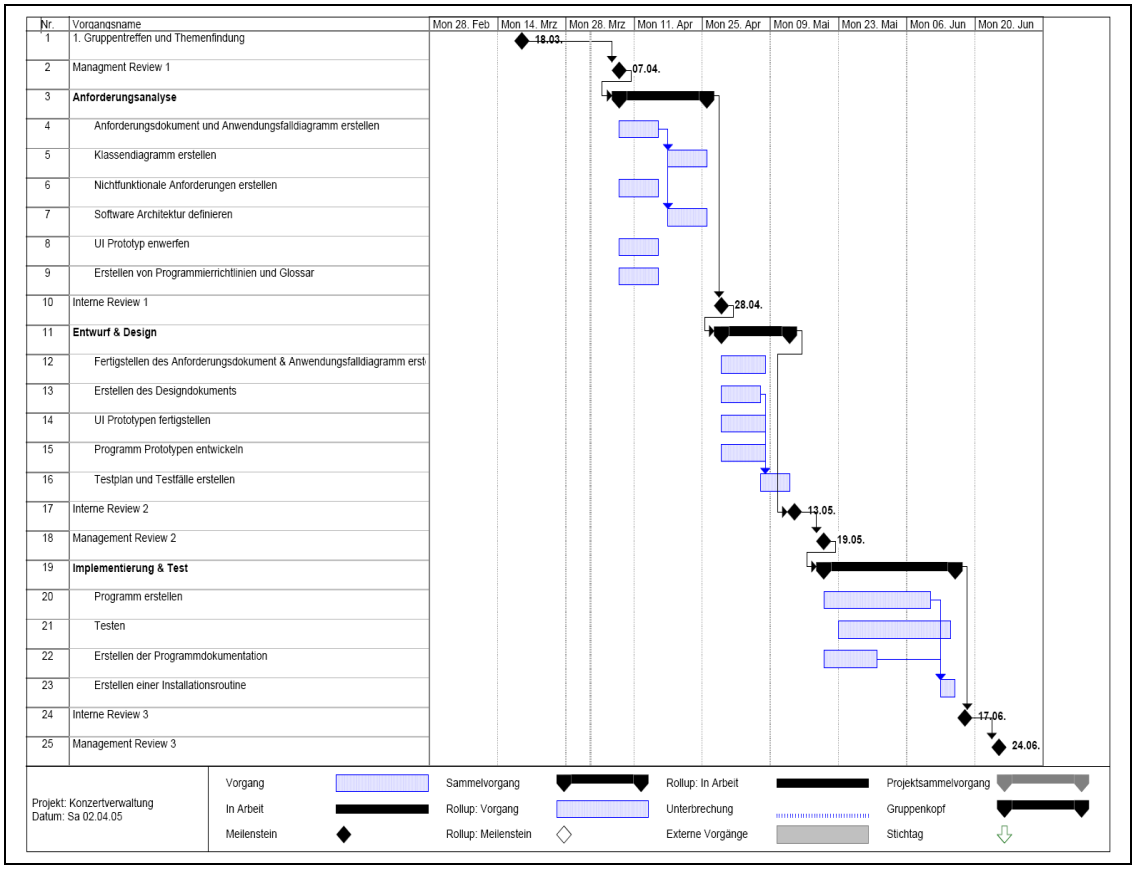
Es werden hier die Ecktermine des Projektes grob abgeschätzt. Weiters erfolgen hier auch bereits erste grobe Abschätzungen des Personal- und Ressourcenaufwandes.

Bezeichnung	Termin	Personalaufwand	Ressourcenaufwand
Projektstart	10.10.2005	120 h	
Anforderungsdokument fertig	17.11.2005	100 h	
Designdokument fertig	25.11.2005	200 h	
Implementierung fertig	05.01.2006	300 h	
Testphase fertig	24.01.2006	200 h	
Programmdokumentation fertig	18.01.2006	50 h	
Projektende	28.01.2006	20 h	

6 Projektplan fein

Hier wird für alle Aufgaben eine Aufwandschätzung abgegeben. Danach werden die Aufgaben in einer Taskliste zusammengefasst. Jeder Task wird einer oder mehreren Personen zugewiesen. Weiters muss man sich hier überlegen, in welcher Reihenfolge die Aufgaben abgearbeitet werden können, bzw. ob Aufgaben parallel durchgeführt werden können.

Hat man diese Aufgaben und die Ressourcenzuordung durchgeführt und in der Task-Liste erfasst, so kann man dies mittels GANTT-Diagramm darstellen, welches in diesem Beispiel in MS-Projekt realisiert wurde. Werden im GANTT-Diagramm Prozentangaben verwendet, so ist genau zu beschreiben, wofür sie verwendet werden und welchen Zweck sie erfüllen.



7 Nichtfunktionale Anforderungen

Hier werden zusätzliche Anforderungen beschrieben, die nicht direkt an die Funktionalität des Systems gerichtet sind.

Bsp.:

- **Qualitätsanforderungen**
  - *Anwenderfreundlichkeit*
    Die Dialoge der Applikation müssen so gestaltet sein, dass die Mitarbeiter nur kurze Einarbeitungszeiten benötigen, um mit der Applikation arbeiten zu können.
  - *Korrektheit*
    In den math. Berechnungen, muss sichergestellt werden, dass diese 100% fehlerfrei sind, da diese die Grundlagen für das Berichtswesen sind. Berechnungen werden auf die 4te Kommastelle genau berechnet.
- **Leistungsanforderungen**
  - *Antwortenzeiten*
    Das Anlegen eines Kunden darf 30 sec nicht überschreiten. Die 30 sec beinhalten die Eingabe der Kundendaten, das Speicher der Daten und das Anzeigen der gespeicherten Daten.
- **Wirtschaftliche Anforderungen**
  - Einsparungspotential
- …

8 Risikoabschätzung

In diesem Dokument werden alle möglichen Risiken, die dem Projekt, von der Planung bis zur Fertigstellung, widerfahren könnten, aufgelistet, bewertet und Gegenmaßnahmen beschrieben.

1. **Nummer**
2. **Name des Risikos**
   Im Namen sollte bereits eindeutig erkennbar sein welches Risiko beschrieben wird.
3. **Kurzbeschreibung**
   Das Risiko wird kurz textuell und, wenn möglich, graphisch beschrieben. Um welches Risiko handelt es sich?
4. **Art des Risiko**
   Hier wird die Art des Risikos angegeben, z.B.: 1-Allgemeines Risiko, 2-Projektspezifisches Risiko
5. **Risiko-Typisierung**

Wo könnte das Risiko auftreten (Planung, Entwicklung, Management, Umfeld, …)?

6. **Priorisierung**
   Hier wird eine Kategorisierung des Risikos vorgenommen, z.B.: 1-Hoch, 2-Mittel, 3-Niedrig.
7. **Eintrittswahrscheinlichkeit**
   Wie wahrscheinlich ist es, dass dieses Risiko eintritt, z.B.: 1-Hoch, 2-Mittel, 3-Niedrig.
8. **Folgewirkung**
   Es muss ermittelt werden, welche Folgewirkungen das eintretende Risiko auslöst. Diese Folgewirkungen, zum Beispiel weitere Risiken, werden dann hier vermerkt. Risiken werden hier mit ihrer Nummer eindeutig zugewiesen.
9. **Gegenmaßnahmen**
   Was kann unternommen werden,
   a) um das Risiko zu vermindern
   b) wenn das Risiko eintritt
10. **Verantwortlichkeit**
    Wer ist für diesen Risikobereich und wer für die Durchführung der Gegenmaßnahmen verantwortlich.
11. **Auftreten**
    Wann ist dieses Risiko aufgetreten (Angabe Zeitspanne bis Behebung)
12. **Eingetretene Folgewirkungen**
    Hier muss angegeben werden welche Folgewirkungen bzw. Folgerisiken eingetreten sind. Falls neue, das heißt bisher noch nicht bedachte, Folgewirkungen aufgetreten sind, müssen diese hier vermerkt werden und als neue Punkte zur Risikoabschätzung hinzugefügt werden.

Es gibt zwei Arten von Risiken: Allgemeine und projektspezifische Risiken. Allgemeine Risiken können in jedem Projekt, unabhängig von Thema oder Art, auftreten. Ein Beispiel für ein allgemeines Risiko ist, wenn ein Teammitglied erkranken bzw. für eine gewisse Zeit verhindert sein sollte.
Projektspezifische Risiken, sind solche, die nur bei derselben Kategorie von Projekten auftreten. Das heißt Risiken, die mit der Planung und Umsetzung bzw. Implementierung verbunden sind. Ein Beispiel für ein projektspezifisches Risiko wäre die Lieferung eines unvollständigen bzw. nicht funktionstüchtigen Moduls oder Klasse, auf die andere Module oder Klassen aufbauen. Handelt es sich hierbei noch dazu um einen zentralen Baustein des Projekts, würden in diesem Fall bei der Priorisierung ’1-Hoch’, bei der Art des Risikos ’Projektspezifisches Risiko’ und bei der Typisierung ’Implementierung; Klasse … in Modul …’ stehen.
In der Spalte Auftreten wird in diesem Beispiel die Zeitspanne der Verzögerung angegeben, bis zu dem Datum, an dem die Fehler behoben wurden und die Weiterarbeit an den darauf aufbauenden Modulen bzw. Klassen aufgenommen werden kann.
Die eingetretenen Folgewirkungen wäre in diesem Fall, unter anderem, die Verzögerung der Weiterarbeit an darauf aufbauenden Modulen bzw. Klassen (Auflistung der betroffenen Teile).

9 Projekttagebuch

Dieses Dokument enthält eine Übersicht des Ablaufs des Projektes aus der Sicht der Gruppentreffen und Reviews.

Bsp.:

Datum	Beginn	Dauer	Ereignis	Verweis
12.10.2005	16:00	1,5h	Erstes Gruppentreffen	Protokoll_20051012.doc
13.10.2005	17:00	0,5h	Skype-Konferenz	Protokoll_20051013.doc
30.10.2005	15:30	2,5h	MR1	Protokoll_20051030.doc
…				

10 Besprechungsprotokolle

In diesem Dokument werden die Inhalte der Besprechungen zusammengefasst. Die Besprechungen sollten folgende Inhalte abdecken:

- Aufgabenverteilung
- Terminplanung (keine Aktivität ohne Termin)
- Besprechen und Lösen von Problemen
- Erstellen von ToDo-Listen
- Weitere Vorgehensweise festlegen

Sollte die Besprechung mit einem Tutor oder einem Assistenten stattgefunden haben (IR oder MR), muss ergänzt werden, wann das Protokoll vom Tutor freigegeben wurde.

Weiters muss in diesem Dokument ersichtlich sein, wo die Besprechung stattgefunden, wer daran teilgenommen, und wie lange die Besprechung gedauert hat.

Bsp.:

BESPRECHUNGSPROTOKOLL

Datum: 10.10.2005
Ort: Mensa
Beginn: 17:00
Ende: 18:00

Anwesende

Name	Email
Max Mustermann	m.m@aon.at
Karin Meier	karin.meier@chello.at
…	

Agenda

1. Aufgabenverteilung
2. Terminplanung
3. Entwicklungsumgebung
4. …

Beschreibung	Verantwortlich
Ad 1) In diesem Meeting wurde die initiale Aufgabenverteilung durchgeführt. Die Ergebnisse der Verteilung werden in der ToDo-Liste (ToDo_20051001_v1.xls) zusammengefasst	Mustermann, bis 10.10.2005
Ad 2) Bei der Terminplanung wurde festgestellt, dass sich der Termin für das MR2 mit einem anderen Abgabegespräch überschneidet. Es muss am Institut nachgefragt werden, ob ein anderer Termin möglich ist.	Teamkoordinator, bis 01.12.2005
Ad 3) Als Entwicklungseditor entschied sich das Team für Eclipse. Als Entwicklungsumgebung wurde für den Client Windows gewählt, für den Server Linux. Die Entwicklungsumgebungen aller Entwickler müssen nun dementsprechend angepasst werden.	ALLE, bis 10.10.2005
…	

11 Meilenstein-Trendanalyse

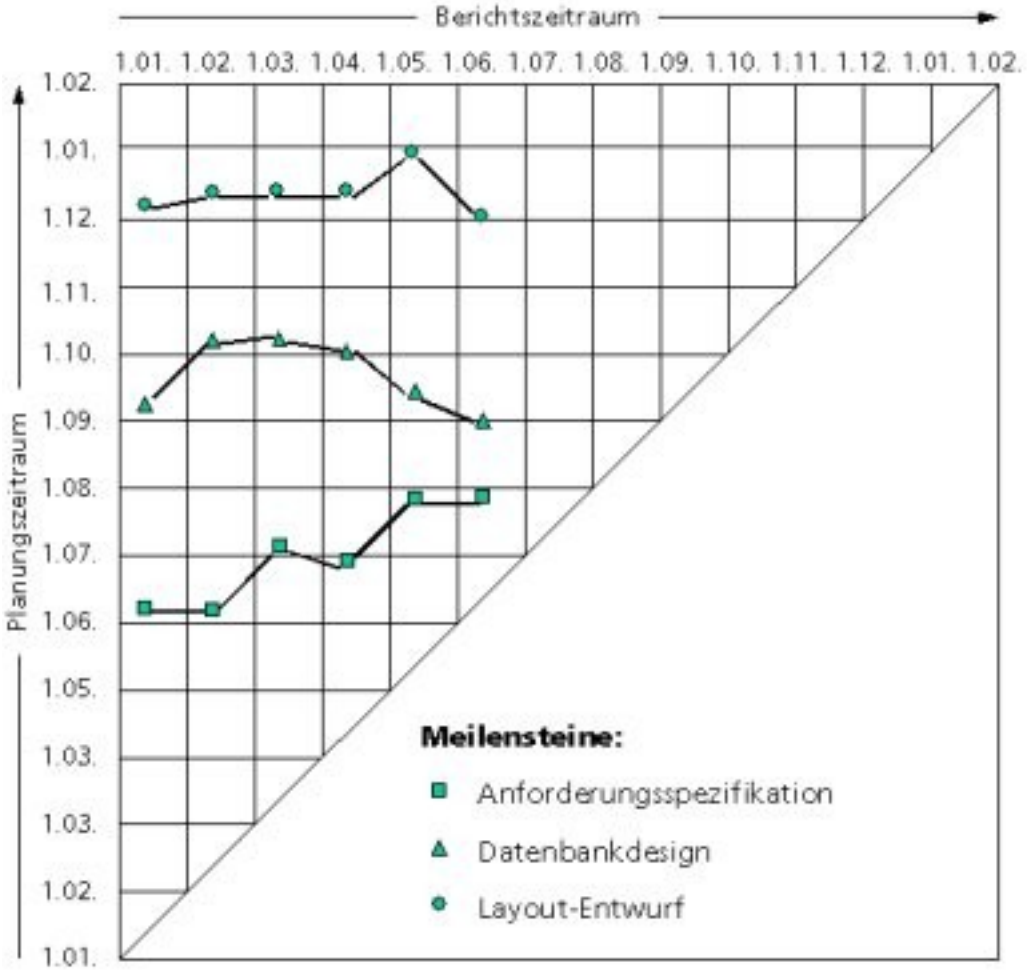
Dies ist eine Methode des Projektmanagements, um das Projekt zu überwachen. Hier werden die SOLL- und IST-Pläne verglichen. Dies dient dazu, Terminverzögerungen so früh als möglich zu erkennen, und sich damit die Möglichkeit zu schaffen, rechtzeitig reagieren zu können.

Interpretation:

- Können die eingetragenen Markierungen mit einer annähernd waagrechten Linie verbunden werden, so wird das Projekt planmäßig abgewickelt.

- Erhält man eine Kurve die nach oben zeigt, so ist ersichtlich, dass das Projekt verzögert ist.

- Erhält man eine Kurve die noch unten zeigt, so ist ersichtlich, dass das Projekt schneller als geplant abgewickelt werden kann.



12 Stundenliste

In diesem Dokument fasst jedes Gruppenmitglied seine erbrachten Aufwände, für die ihm zugeteilten Aktivitäten, zusammen. Die Stundenlisten sollten bei allen Gruppenmitgliedern ein einheitliches Aussehen haben, weiters sollte der Gesamtaufwand der einzelnen Teammitglieder ersichtlich sein.

Bsp.

Datum	Von	Bis	Stunden	Tätigkeit
06.10.2005	16:30	18:30	2	Erstellen des Anforderungsdokumentes
06.10.2005	19:00	20:00	1	Erstellen des Meetingprotokolls
…				
Summe			***25***	

13 UI-Prototyp und Beschreibung

In diesem Dokument wird beschrieben wie die Implementierung des Projektes aussehen könnte. Hier müssen alle notwendigen Eingabe-, Such-, und Anzeigeoberflächen (Bildschirmmasken) präsentiert und beschrieben werden.

Für die einzelnen Masken wird folgendes beschrieben:

- Welche Eingabefelder gibt es
- Sind diese Felder „Muss-Felder“ bei der Eingabe
- Was wird mit dieser Bildschirmmaske abgebildet (Beschreibung)
- Wie wird durch die Applikation navigiert (welches UI wird nach dem Klicken welches Buttons aufgerufen?)
- Welche Menü-Einträge gibt es und was bewirken diese
    - Wie funktionieren die Suchen
    - Welche Ergebnisse liefern die Suchen

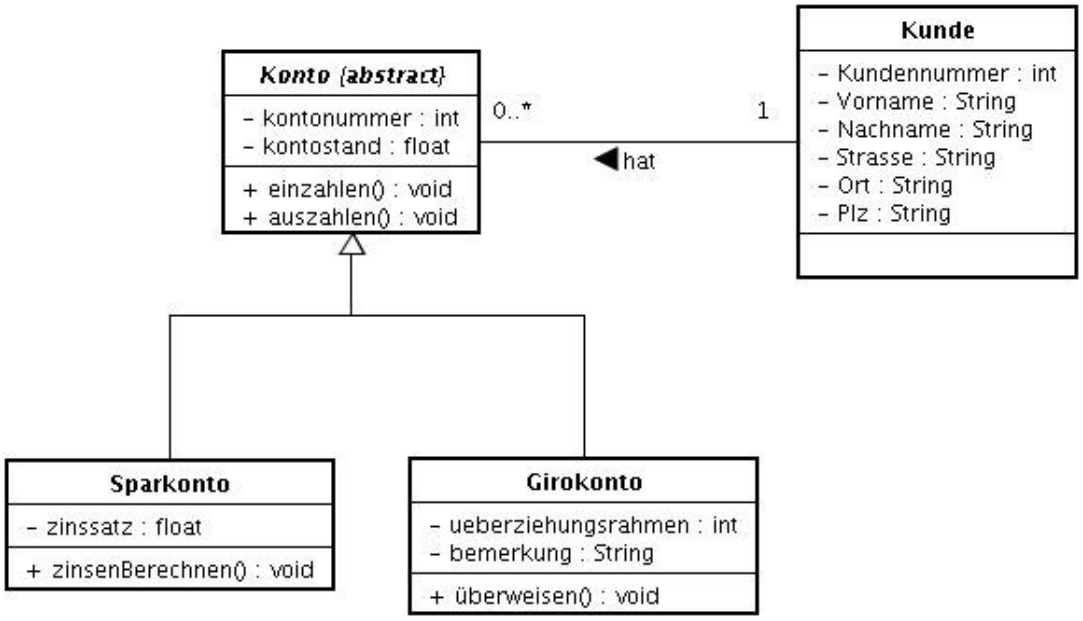
Alle Bildschirmmasken sind zusätzlich auch mittels Screenshot darzustellen.

14 Architekturdokument

Dieses Dokument bildet die Grundlage für die Implementierung des Projektes und enthält alle dafür notwendigen Informationen. Die unten angeführten Themengebiete werden in diesem Dokument zusammengefasst.

14.1 Klassendiagramm

In diesem Dokument werden die Klassen, die sich aus den Anforderungen ergeben, beschrieben, und mittels UML-Klassendiagramm dargestellt.



Beim Erstellen des Klassendiagramms muss auf folgendes geachtet werden:

- Attribute und Methoden müssen mit den richtigen Sichtbarkeitssymbolen gekennzeichnet werden.
- Die Assoziationen zwischen den Klassen müssen beschriftet und mit Kardinalitäten versehen sein.
- Lassen sich Assoziationen als Aggregation oder Komposition darstellen, so muss dies gemacht werden

Sichtbarkeit von Attributen oder Methoden	
Public (+)	Jedes andere Element hat uneingeschränkten Zugriff.
Private (-)	Nur Instanzen der Klasse, die das Attribut definiert, dürfen zugreifen.
Protected (#)	Nur Instanzen der Klasse, die das Attribut definiert, und Instanzen abgeleiteter Klassen dürfen zugreifen.
Package (~)	Das Attribut ist für alle Elemente, die sich im selben Paket wie die definierte Klasse befinden, sicht- und zugreifbar.

Das Klassendiagramm muss dann noch um eine textuelle Beschreibung erweitert werden.

14.2 Datenbankbeschreibung

Hier wird jede Datenbanktabelle genauestens beschrieben:

- Welche Felder beinhaltet die Tabelle
- Welchen Datentyp haben diese Felder
- Welche Information wird in diesem Feld gespeichert
- Angabe der Schlüssel (Foreign Key, Primary Key)
- Mit welcher Option werden diese Felder angelegt

Folgende Symbole können in der Spalte Option verwendet werden:

Kürzel	Bedeutung	Beschreibung
1	Eindeutig	Dieses Feld kennzeichnet den Datensatz eindeutig, d.h. dieser Wert kommt in der gesamten DB-Tabelle nur 1x vor.
[1]	Eindeutig über mehrere Felder	Alle Felder die diese Option haben bilden gemeinsam einen eindeutigen Schlüssel.
!	Nicht leer	Wenn ein Feld keine NULL-Values enthalten darf, so muss dieses Feld mit dieser Option gekennzeichnet werden
+	Autoincrement	Dieses Feld wird beim Einfügen eines Datensatzes automatisch erhöht – dies ist ideal bei künstlichen Schlüsseln wie z.B.: lfd. Nummer
P	Primary Key	Primärschlüssel
F	Foreign Key	Fremdschlüssel

Bsp. Tabelle PERSON

PERSON			
Feldname	**Typ**	**Option**	**Bemerkung**
Pid	INTEGER	!1+P	Künstlicher primary Key dieser Tabelle. Jeder neue Datensatz wird automatischer mit einer eindeutigen ID gespeichert.
Vorname	VARCHAR(40)	!	Vorname der Person, max. 40 Zeichen lang
Nachname	VARCHAR(60)	!	Nachname der Person, max. 60 Zeichen lang
GebDat	DATE		Geburtsdatum der Person, da dies nicht immer bekannt ist, kann das Feld auch leer sein.
AdrNr	INTEGER	!F	Dieses Feld referenziert auf die Tabelle Adresse, in der die genaue Adresse der Person gespeichert ist.

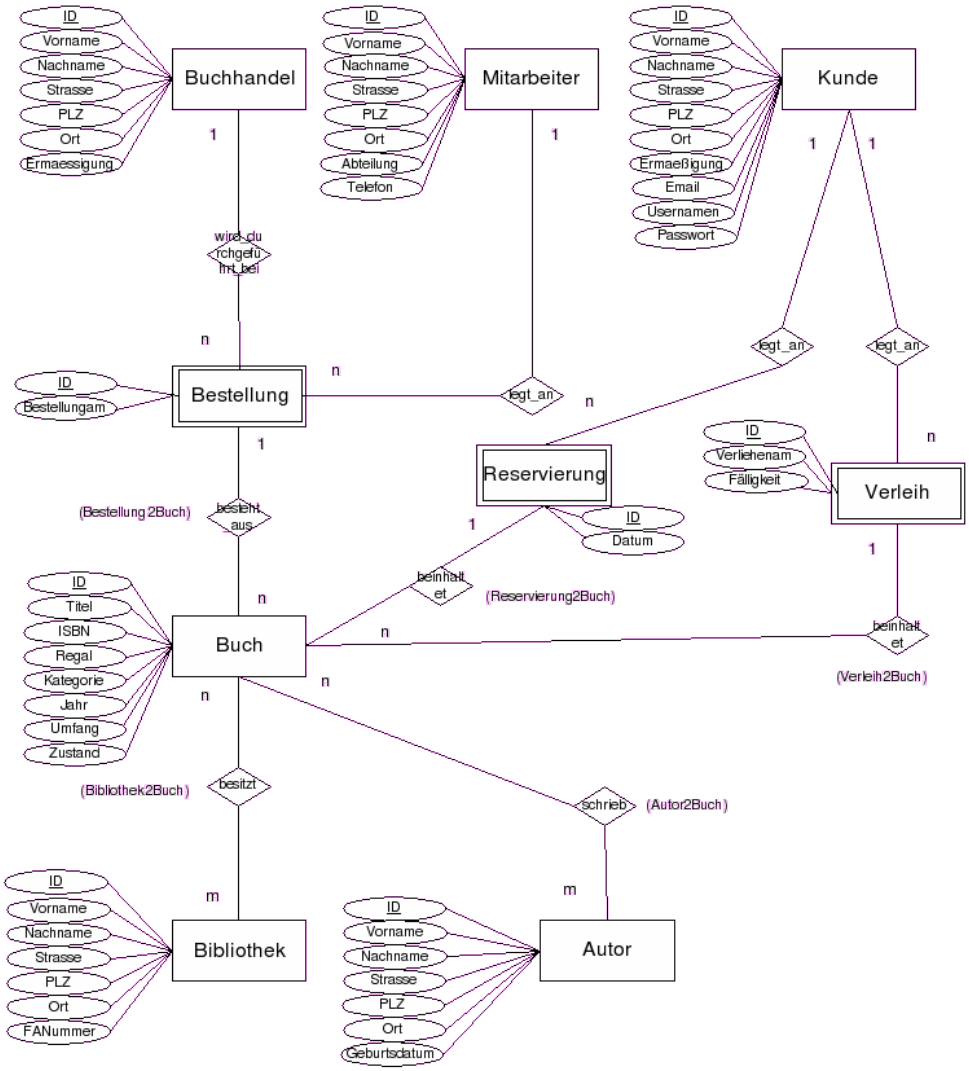
14.3 EER-Diagramm

In diesem Dokument soll der konzeptionelle Entwurf des Datenbankschemas mit Hilfe des Entity-Relationship-Diagramm erstellt werden. Hier müssen die Entität und deren Relationen beschrieben werden.

Hier sind folgende Punkte zu berücksichtigen:

- Entitäten, Relationen und Attribute müssen einen Namen haben
- Schwache Entitäten müssen sichtbar gekennzeichnet werden
- Kardinalitäten müssen angegeben werden
- Die Schlüsselattribute müssen gekennzeichnet werden

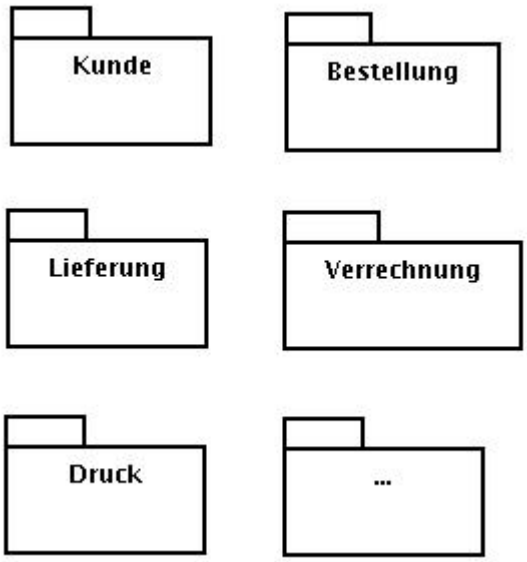
Bsp.



14.4 Paket-Diagramm

Um einen Überblick über die Klassen im Klassendiagramm zu erhalten, versucht man diese über technische bzw. fachliche Gesichtspunkte zu Packages zusammenzufassen. Dies wird dann grafisch dargestellt und textuell beschrieben, welches Package welche Funktionaliäten enthält.

Bsp.:

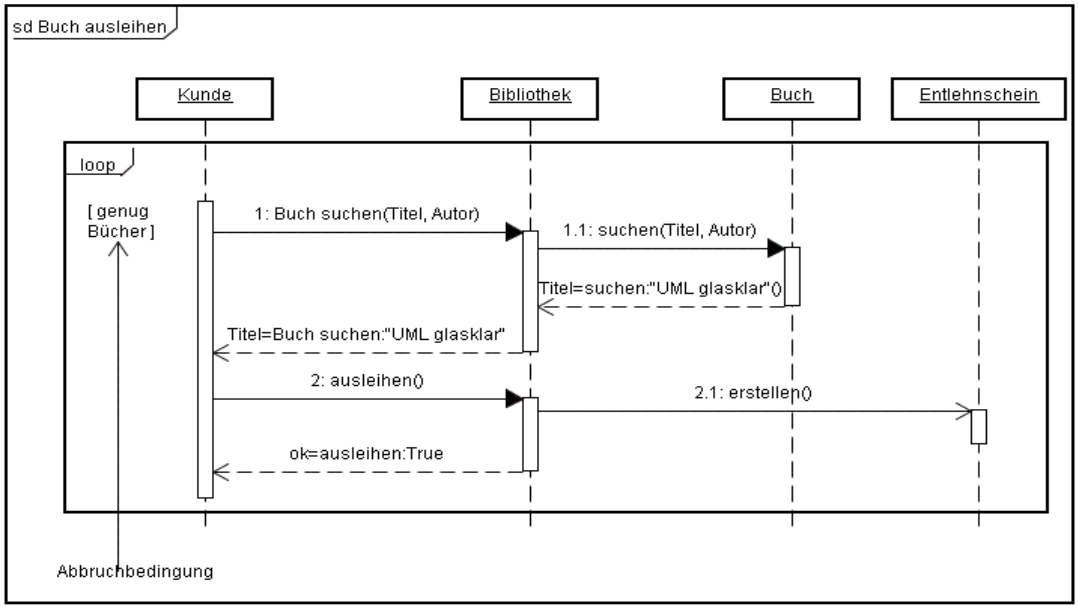


Package	Funktionalität
Kunde	Enthält die Klassen Person und Kunde
	Weiters sind hier folgende Funktionalitäten implementiert: • Kunde anlegen • Kunde löschen • Kunde suchen • ...
Druck	In diesem Package sind alle notwendigen Klassen für die Druckaufbereitung enhalten (CSerienbrief, CDokument, CDruckAuf, CEinzeldruck, CMassendruck, …)
	Weiters sind hier folgende Funktionalitäten implementiert: • Serienbrief erstellen • Einzeldruck • Massendruck • Druckaufbereitung für den Einzel- und Massendruck • Erstellen der Druckdokumente in XML
…	

14.5 Sequenzdiagramm

Das Erstellen des Sequenzdiagramms ist erst nach der Erstellung des Klassendiagramms möglich. Das Sequenzdiagramm beschreibt, wie in einem bestimmten Ablauf (z.B. eine bestimmte Methode) die notwendigen Objekte miteinander kommunizieren, bzw. in welcher Reihenfolge welche Objekte verwendet werden bzw. welche Methoden aufgerufen werden.

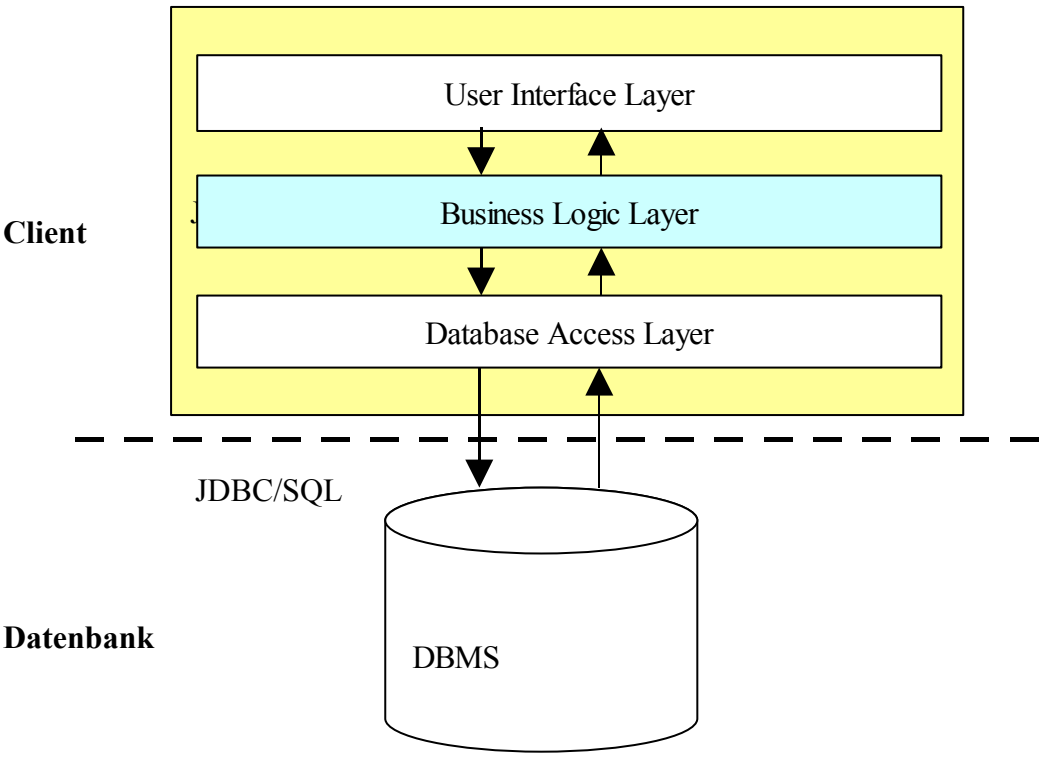
Bsp.:



14.6 Systemarchitektur

Hier soll kurz (textuell und graphisch) beschrieben werden wie die Systemarchitektur aufgebaut ist, bzw. wie diese funktioniert (z.B.: für eine Webapplikation oder Client/Server Programm)

Bsp.:



14.7 Domänenmodell

Das Domänenmodell zeigt die wichtigsten Objekte der Anwendungsdomäne und ihre Attribute. Diese Darstellung soll der Identifizierung und dem Verständnis aller Objekte der Anwendungsdomäne dienen. Typischerweise handelt es sich dabei um folgendes:

- an Funktionen des Systems beteiligte oder davon betroffene Personen bzw. deren Rollen (z.B. Kunde, Ansprechpartner, Verkäufer, usw.)
- Objekte, welche Zustände eines Prozesses wiedergeben (z.B. Buchung, Reparatur, usw.)
- Objekte, welche wichtige Sachgegenstände für den Prozess beschreiben (z.B. Vertrag, Rechnung, usw.)
- Objekte des Alltags der Anwendungsdomäne (z.B. Auto bei Autohändlern, Buch bei Büchereien)
- Objekte, die Infrastruktur beschreiben (z.B. Abteilungshierarchie)

Zur Darstellung des Domänenmodells wird ein UML-Klassendiagramm verwendet, in dem die Namen der Objekte und ihre Attribute angegeben werden, die Methoden werden frei gelassen.

15 Testdokument

Dieses Dokument beinhaltet alle Aspekte um die Applikation einem allumfassenden Test zu unterziehen.

15.1 Begriffsbestimmungen und Abkürzungen

Werden für die Erstellung des Dokumentes bestimmte spezifische Begriffe bzw. Abkürzungen verwendet, so müssen diese hier beschrieben werden.

Bsp.:
Testobjekt
Unter dem Begriff Testobjekt versteht man die zu testende Einheit.

15.2 Testanforderungen

Hier wird beschrieben, welche Testkomponenten mit welcher Priorität getestet werden sollen.

Bsp.:
Alle Standardfunktionen mit höchster Prioriät

15.3 Testkomponenten

Hier werden jene Komponenten beschrieben, welche dem Test unterzogen werden (entspricht in etwa den Packages, welche man im Paket-Diagram beschrieben hat)
Bsp.:
KUNDE
KONTO
VERTRAG

15.4 Welche Funktionen werden getestet

Hat man die Testkomponenten definiert, so werden hier noch die zu testenden Funktionen der Komponenten aufgelistet:

Komponente	Funktion	Priorität
KUNDE	Kunde anlegen	Sehr hoch
KUNDE	Kunde suchen	Sehr hoch
KUNDE	Kunde löschen	Hoch
KONTO	Konto anlegen	Sehr hoch
KONTO	Konto löschen	Sehr hoch
…		

16 Testfälle

In diesem Dokument wird jeder einzelne Testfall beschrieben. Die Beschreibung des Testfalls soll wie folgt aufgebaut sein:

1. Testkomponente
2. Vorbedingungen
3. Nummer des Testfalls
4. Typ des Testfalls (FF..Fehlerfall, NF..Normalfall, SF..Sonderfall, …)
5. Kurzbeschreibung
6. Erwartetes Ergebnis
7. Eingaben
8. Erreichtes Ergebnis
9. Beschreibung bei Fehlerfall
10. Bestanden Ja/Nein

Bsp.:

1. Kunde

1.1 Kunde anlegen

Vorbedingung: Das Programm läuft, der Benutzer hat sich korrekt eingeloggt und hat die dafür notwendigen Rechte.

Nr	Typ	Beschreibung	Erwartetes Ergebnis	Eingaben	Erreichtes Ergebnis	Beschreibung bei Fehlerfall	Bestanden Ja/Nein
1	NF	Anlegen des Kunden durch vollständiges Befüllen des Eingabedialogs	Kunde wird angelegt	Name: Maier GebDat: 01.10.1981			
2	FF	Der Eingabedialog wird nicht vollständig befüllt	Fehlermeldung: Kunde konnte nicht angelegt werden! Bitte befüllen Sie das Datenfeld „GebDat.:“	Name: Maier GebDat: „“			
…							

17 Testbericht

Dieses Dokument enthält die Ergebnisse der durchgeführten Tests. Für jede Testiteration ist eine neue Version des Testberichtes anzulegen. Hier werden die letzten drei Spalten der Testfälle befüllt. Man erkennt daher, welcher Test bestanden hat und welcher nicht. Zusätzlich zur Tabelle müssen folgende Punkte textuell beschrieben werden:

- Welche Fehlerfälle konnten bei dieser Testiteration beobachtet werden?
- Welche Maßnahmen werden gesetzt?
- Welche Änderungen (im Projekt, zeitlicher Ablauf, etc) ergeben sich durch diese Testiteration?
- Welche neuen Testfälle haben sich durch diese Testiteration ergeben (z.B. durch einen Fehlerfall müssen Teile detaillierter getestet werden, oder es hat sich ergeben, dass Teile nicht ausreichend abgedeckt sind)?

In dieser Beschreibung muss aber nicht noch einmal der komplette Testfall angegeben werden, sondern man kann einfach dessen Nummer angeben.

Für die Übungsreihe aus Software Engineering sind in der Regel zwei Testiterationen ausreichend, die dritte soll nur mehr angedeutet werden, d.h. es muss beim dritten Mal nicht mehr komplett durchgetestet werden, sondern nur mehr bestehende Fehler behoben und dann ausgetestet werden.

18 Benutzerhandbuch

Dieses Dokument soll dem Benutzer behilflich sein um die Applikation zu bedienen. Hier werden die einzelnen Bildschirmmasken mittels Screenshot in das Dokument aufgenommen. Danach soll erklärt werden, welche Aktivitäten der Benutzer durchführen muss, um bestimmte Funktionalitäten der Applikation erfolgreich durchführen zu können.

19 Installationsleitfaden

Dieses Dokument soll festlegen, welche Hard- bzw. Software-Komponenten notwendig sind, damit es möglich ist die Applikation zu starten. Weiters müssen hier auch, falls notwendig, diverse Konfigurationen beschrieben werden.

Bsp.

Datenbank

- Download der erforderlichen Komponenten => Link zur Downloadadresse
  Installation der erforderlichen Komponenten => Link zur Downloadadresse

Webserver

- Download des Apache => Link zur Downloadadresse

- Installation des Apache
  Beschreibung wie der Apache installiert wurde
- Konfiguration des Apache
  Beschreiben der Änderungen in den Konfigurationsfiles
- **...**

20 Projektende Bericht

Hier soll kurz zusammengefasst werden, wie das Projekt verlaufen ist. Man betrachtet hier das gesamte Projekt, vom Projektstart bis zum Projektende um unter folgenden Gesichtspunkten ein Resümee über das Projekt ziehen zu können:

1. **Planung**
   a. Wurden die geschätzten Aufwände über- bzw. unterschritten.
   b. Warum wurden die geschätzen Aufwände über- bzw. unterschritten.

2. **Probleme**
   a. Wo traten Probleme auf?
   b. Wie konnten diese Probleme behoben werden?
   c. Mit welchen Maßnahmen konnte man diese Probleme beheben?

3. **Team**
   a. Wie hat das Team funktioniert?
   b. Welche Stärken, bzw. welche Schwächen konnte man erkennen.

4. **…**

Anhang – Der Rational Unified Process (RUP)

Der Rational Unified Process (RUP) wurde von der Firma Rational Software (nun IBM) entwickelt. Der RUP ist ein Vorgehensmodell zur Softwareentwicklung.

Was ist ein Prozess?

Ein Prozess beschreibt immer wer, wann wie (et)was zu erledigen hat. Diese vier Kategorien werden wie folgt unterteilt:

- Wer: Mitarbeiter
- Wann: Workflow
- Was: Artefakt
- Wie: Tätigkeit

Der RUP gliedert sich in 4 Phasen:

1. **Inception (Konzeptionsphase):** In dieser Phase wird das Konzept für das weitere Vorgehen entwickelt. Diese Phase besteht in den meisten Fällen aus nur einer Iteration. Sinn dieser Phase ist eine „Vision“ zu erschaffen. Diese „Vision“ beinhaltet die simple Frage: „Wo wollen wir eigentlich hin?“. Hier werden die Anforderungen des Kunden so weit analysiert, dass Kunde und Entwickler die gleiche Sicht auf die Anforderungen erhalten.

   Dies wird unter anderem mit Workflow Business Modelling und dem Anforderungsmanagement erreicht. Alle gewonnenen Informationen über Umfang, Funktion, Zielgruppe, usw. fließen dabei in die „Vision“ ein. Danach kann das Projektmanagement anhand des Projektumfangs und der Vision einen groben Zeit- und Kostenplan bzw. auch eine Risikoabschätzung erstellen.

   Sind sich Kunde, Projektmanagement und Entwickler über die Anforderungen, den Kosten-, den Zeitplan und die Risiken einig, dann kann die Inceptionphase abgeschlossen werden. Ist dies nicht der Fall, so muss eine weitere Iteration eingeführt werden, in der die fehlenden Informationen eingeholt, in den Plänen (Kosten-, Zeit, ...) ergänzt, und dem Kunden wieder vorgelegt werden.

2. **Elaboration (Entwurfsphase):** In dieser Phase beginnt die Ausarbeitung des Projektes. Nun ist zu klären „Was wie hergestellt werden soll?“. Hier werden aus den Anforderungen Use-Cases erstellt und gegen die Anforderungen überprüft, damit sichergestellt ist, dass die Anforderungen in die richtige Richtung entwickelt, und nichts vergessen wurde.

   Sind die Use-Cases vollständig, dann nimmt der Systemarchitekt die Arbeit auf, um eine solide Architektur für das System zu erstellen, welche allen

funktionellen und nichtfunktionellen Anforderungen des Kunden gerecht wird.

Dieser Architektur-Prototyp soll zeigen, dass alle Anforderungen auch realisierbar sind. Die wichtigsten Komponenten und Use-Cases werden hier schon grob implementiert. Ab diesem Zeitpunkt ist eine detaillierte Planung von Zeit, Aufwand und Budget durchzuführen.

Im Budget sollten auch Kosten für Tools und Prozesse, die im Environment Workflow organisiert und angeschafft werden, enthalten sein. Die Eigenschaften des Environments können sich aber bis zur Construction Phase ändern, da Tools oder Prozesse je nach ihrer Effizienz wieder ausgemustert und durch andere ersetzt werden können.

Die Elaboration Phase ist beendet, wenn klar ist, dass das System den fixierten Zeit- und Budgetplan einhält und ein stabiler Architektur-Protoyp vorhanden ist, an dem sich der weitere Prozess orientiert. Sind diese Anforderungen alle geklärt, wird zum Abschluss der Elaboration Phase die Planung der Construction Phase erstellt.

3. **Construction (Realisierungs-/Implementierungsphase:** Hier findet die eigentliche Realisierung des Projektes statt. Nun werden alle Komponenten implementiert, getestet und die Architektur weiter ausgebaut. Ausgiebiges regelmäßiges Testen stellt sicher, dass sich keine Fehler einschleichen. Es ist offensichtlich, dass Use-Cases anders getestet werden müssen als die Zuverlässigkeit des Systems oder einzelne Quellcodes. Hier kann man sich verschiedener Testarten bedienen.

   Es müssen daher bereits Testpläne und Testfälle vorhanden sein. Treten hier Fehler auf, so müssen diese analysiert und behoben werden. Dadurch sind auch hier Iterationen dieser Phase notwendig, da die behobenen Fehler auch dokumentiert werden müssen.

   Während das System ausgebaut wird, überwacht das Architekturteam das System und überprüft, ob alle Richtlinen (Schnittstellenspezifikationen, Codierungsrichtlinien, Architektur) eingehalten werden. Damit wird sichergestellt, dass die Kommunikation und Zusammenarbeit der Komponenten funktioniert.

4. **Transition (Einführungsphase):** Die Transition Phase ist die letzte Phase des Entwicklungsprozesses. Hier wird dem Kunden eine Beta-Version in einer Testumgebung zur Verfügung gestellt. Damit dies funktioniert, muss das Configuration Management schon viel Vorarbeit geleistet haben (Installationsleitfaden, Server- , Client- und Datenbankkonfigurationen, ...).

   Ist die Installation abgeschlossen und sind die wichtigsten Komponenten konfiguriert, startet ein ausgiebiger Test, in dem das Feedback des Kunden eine der wichtigsten Komponenten darstellt. Hier wird dem System der Feinschliff verpasst. Während des Tests ist es weiters notwendig, dass Begleitunterlagen erstellt werden (Benutzerhandbuch, Trainingsmaterial,

Erstellen von Prozessen für den Support, ...).

Die Entwicklung ist abgeschlossen, wenn der Kunde zufrieden ist und alle Grundlagen der „Vision“ erfüllt sind. Ist das Produkt fertig steht den Entwicklern noch ein weiterer Arbeitsgang bevor: RUP empfiehlt ein internes Meeting aller Projektbeteiligten, um das ganze Projekt Revue passieren zu lassen, damit man aus den gemachten Fehlern lernen kann.