Projekthandbuch

Praktikum Software Engineering 1999

SCG, IAM, Universität Bern

1. Einführung

> -- ***Dieses Dokument sollte eine Hilfestellung sein und die Studenten nicht verwirren. Man sollte Feedback welches auf Unklarheit hindeutet aufnehmen und das Dokument dahingehend verbessern.***

Das Projekthandbuch enthält eine nähere Beschreibung aller Projektphasen und der Deliverables, welche in diesen Phase erstellt (und abgeliefert werden) müssen.

1.1 Informationsquellen

Das Projekthandbuch enthält nicht alle Informationen die für die Teilnehmer des PSE wichtig sind. Weitere Unterlagen werden im Laufe des Seminars verteilt oder auf der PSE Homepage veröffentlicht. Die Homepage befindet sich unter

```
http://www.iam.unibe.ch/~scg/Teaching/PSE
```

Auf der Homepage werden Literaturlisten veröffentlicht, sowie Hinweise und Dokumentation zu Tools und Techniken.

Für Diskussionen von allgemeinen Interesse steht den Teilnehmern ein WikiWiki Web (ein WWW-basiertes Blackboard) zur Verfügung:

```
http://www.iam.unibe.ch/~scg/cgi-bin/PSE99.cgi
```

Auf dem Wiki können Fragen an den Auftraggeber, technische Probleme (und ihre Lösungen) und Beiträge zu anderen Themen angeschlagen werden. Die Anleitung zur Bedienung des Wikis befindet sich im Wiki selbst.

Falls sich Fragen aus keiner der besprochenen Quellen beantworten lassen, sind die Assistenten auszuquetschen.

1.2 Deliverables

Als *Deliverables* werden alle Artefakte bezeichnet, die im Lauf des Softwareprozesses erstellt werden. Das beinhaltet das Pflichtenheft, die Designdokumente, die Prototypen, die Arbeitspläne, das Softwareprodukt selbst. Wie schon in der Einführungsvorlesung betont sind alle Deliverables über die Gruppenwebseite zugänglich zu machen.

Für formelle Darstellungen sollte vor allem UML benutzt werden. Für besondere Fälle können auch Petri Netze, endliche Automaten, usw. verwendet werden. Falls eine formelle Darstellung für bestimmte Sachverhalte gewählt wird, dann muss dazu immer noch ein erk-

lärender Text verfasst werden. Der Grund dafür ist unter anderem, dass die Spezifikationen auch für Personen lesbar sein sollen welche die formelle Notation nicht kennen.

1.3 Kooperation zwischen den Gruppen

Die Kooperation zwischen den einzelnen Gruppen wird zuallererst dadurch erzwungen ,dass jedes Deliverable einer Gruppe von einer anderen Gruppe nachgeprüft wird (*Review*). Dokumnente sollen sorgfältig gelesen und auf Verständlichkeit, Eindeutigkeit, Vollständigkeit, Konsistenz, und Überprüfbarkeit hin untersucht werden. Beim Review von lauffähigen Programmen ist darauf zu achten, ob die Installation klar und vollständig beschrieben ist, ob sie sich der Beschreibung entspechend durchführen lässt, ob die Funktionalität des Programmes beschrieben ist und ob das Programm diese Funktionalität auch liefert. Ein schriftlicher Review-Bericht ist das Resultat eines Reviews. Vorlagen für Reviewprotokolle und -berichte werden von den Assistenten verteilt werden.

Die Zusammenarbeit zwischen Gruppen ist auch in anderen Bereichen möglich:

- Weitergabe von Spezialwissen. Falls sich eine Gruppe in einen bestimmten Themenbereich speziell tief eingearbeitet hat, ist sie eingeladen, dies auf dem PSE Wiki kundzutun und Wissenstransfer in Form von Vorträgen oder Dokumenten anzubieten.
- Komponentenmarkt. Falls eine Gruppe besonders viel Arbeit in eine bestimmtes Software-Modul investiert hat, kann sie diese auf dem PSE Wiki feilbieten.
  Hilfswerkzeuge, die während des Projektverlaufs entwickelt werden, sind ebenfalls ein mögliches Handelsobjekt.

2. Phaseneinteilung des Software-Projektes

Die Erstellung eines Software-Produktes unterteilt sich in mehrere Phasen. Diese Phasen beinhalten eine abgegrenzte Aufgabe und am Ende jeder Phase ist ein bestimmtes Ergebnis zu erreichen (Dokument oder Programmcode). In einem Softwareprojekt macht der Anteil der Codierungsphase nur rund 20% des Gesamtaufwandes der Projektrealisation aus. Dieser Aufwand wird umso geringer, je besser die anderen Phasen bearbeitet werden. Zu Abschätzung des Aufwandes für die einzelnen Phasen soll die folgende Tabelle dienen:

	Phase	**Anteil am Gesamtaufwand**
Analyse	Vorstudie (*pilot study*)	5-10%
	Plichtenhefterstellung (*requirement specification*)	10-20%
Design	Grobdesign	5-15%
	Feindesign	10-20%

	Phase	Anteil am Gesamtaufwand
Implementierung	Codierung	10-20%
	Tests und Debugging	30-50%

Die hier angegebenen Werte dienen nur zur Orientierung und können je nach Art des zu bearbeitetenden Projektes variieren.

Es ist allerdings festzuhalten, dass die einzelnen Phasen nicht notwendigerweise sequentiell bearbeitet werden. Wie in der einführenden Vorlesung betont, verläuft der Software Prozess iterativ und alle Phasen sind zu jeder Zeit aktuell. So ist es z.B. bei der Erstellung eines Userinterfaces üblich, dass man einen Prototypen des Userinterfaces bereits in der Phase Analyse II (Requirementspezifikation) erstellt. Weiterhin wird man bereits codieren, bevor man die Designphase abgeschlossen hat. Diese Überschneidungen haben zum Ziel, dass man sich die z.T. fehlenden Informationen über funktionale Abhängigkeiten, Abhängigkeiten von der Umgebung usw. durch das Vorziehen einzelner Arbeitsschritte erarbeiten kann.

Dadurch, dass ständig Neubearbeitungen älterer Phasen erfolgen müssen natürlich auch bereits einmal abgeschlossene Dokumente (auch freigegeben, oder *released*) immer wieder neu geöffnet und geändert werden. Eine Anforderung die an die Teams gestellt wird, ist dass bis zum Schluss des Projekts alle (freigegebenen) Versionen aller Dokumente von der Gruppen-Webseite abrufbar bleiben, sodass der Prozess verfolgt werden kann.

2.1 Zusätzliche Phasen

Obwohl das Praktikum Software Engineering die Realität der Softwareerstellung im grösseren Rahmen und industriellen Kontext möglichst nahe simulieren soll, bleibt es ein Sandkastenspiel. Das grösste Handicap der Simulation ist dabei nicht, dass man kein Geld kriegt (das wird dadurch aufgehoben, dass man den Job nicht verlieren kann, wenn's schiefgeht), sondern dass das Praktikum nach drei Monaten beendet ist. Der grösste Teil des Software-Lebenszyklus, die Produktions- und Wartungsphase (*Maintenance*), kann also nicht simuliert werden. Die grosse Wichtigkeit dieser Phase schlägt sich aber in "echten" Projekten schon in den frühen Phasen nieder, wo Wartbarkeit undTechnologieentwicklung in Betracht gezogen werden müssen.

Nun ist es im diesjährigen Praktikum zum erstenmal der Fall, dass die erstellte Software wirklich zum Einsatz gelangen wird und ihre Lebenszeit sich weit über das SS99 hinaus erstrecken wird. Es wird deshalb von den Teilnehmern erwartet, dass sie die oben besprochenen Aspekte in Betracht ziehen.

2.2 Beschreibung der Prozessschritte

In den folgenden Sektionen werden die einzelnen Phasen des Projektes näher beschrieben. Für jeden Prozessschritt werden die technischen Tätigkeiten und die Deliverables angegeben. Um die Erstellung der Deliverables zu erleichtern, werden für beinahe jedes Dokument beispielhafte Dokument-Strukturen aufgefuehrt. Diese sollen als Ideengeber oder Checklisten dienen, und nicht als dogmatische Vorgabe verstanden werden. Die Listen sind bezüglich des

konkreten Projektes einerseits unvollständig und anderseits zu ausführlich.

2.3 Phasenunabhängige Tätigkeiten

Eine Reihe von Tätigkeiten müssen während des gesamten Prozesses durchgeführt werden, unabhängig von den einzelnen Phasen. Sie werden im folgenden kurz beschrieben.

-- the section is not yet fully edited from this point on

- technische Abstimmung mit dem Auftraggeber bei Anforderungsänderungen oder bei Unklarheiten

  -- what is a better word for technische Abstimmung?

- qualitätssichernde Massnahmen: Reviews

  -- we should actually talk individually about the different reviews

Test und Debugging sind im eigentlichen Sinne keine einzelne Phasen. Tests beginnen bereits in der Analysephase. Jeder Schritt muss getestet werden. Der Programmierer ist für seinen Teil des Projektes verantwortlich. Jeder Teilschritt muss von geeigneten Tests begleitet werden. Dabei kommen sogenannten Regressionstests eine besondere Bedeutung zu, da diese garantieren, dass Änderungen in der Applikation nicht dazu führen, dass bereits existierende und getestete Teile der Applikation jetzt fehlerhaft arbeiten. Es ist also wichtig, bereits am Beginn der Projektbearbeitung eine geeignete Teststrategie zu entwickeln. Im PSE wird der Gebrauch des Jave Testing Frameworks **JUnit** nahegelegt (siehe PSE Homepage).

3. Analyse

Die Analyse-Phase verläuft von der Problemstellung bis zum fertigestellten Pflichtenheft. Bei iterativer oder paralleler Entwicklung ist die Analysephase natürlich nie abgeschlossen und das Pflichtenheft wird immer wieder angepasst werden müssen.

Wichtig ist, das während der Analyse noch kein Design gemacht wird!

3.1 Problemstellung

Die Problemstellung im allgemeinen

- umfasst grob das Ziel des Projektes und eventuelle die derzeit benutzten Lösungen,
- charakterisiert gegenwärtige Schwachstellen, z.B. hinsichtlich Funktionalität, Zuverlässigkeit,
- beschreibt die Ziele für eine neue Lösung,
- und nennt die Randbedingungen für die Lösung, z.B. Anwendungsumgebung, Zeitrahmen.

Die Problemstellung im konkreten wurde in der Einführungsvorlesung vorgestellt. Detaillierte Beschreibungen können in Form von zwei Dokumenten von der Homepage heruntergeladen werden.

Um sich mit der Problemstellung vertraut zu machen und sich auf das Sammeln von Requirements vorzubreiten, ist eine Vorstudie anzustellen.

-- we should put pilot study into the schedule on the introductory foils

Die Vorstudie wird in zwei Teilphasen zerlegt: Ist-Analyse und Sollkonzept.

3.2 Ist-Analyse

Diese Phase dient dazu das Problem, welches vom Kunden aus seiner, der Benutzersicht, geschildert wird, in die Sprache und Sichtweise des Informatikers zu übersetzen.

Tätigkeiten

- Ermittlung der Ist-Situation
- ggf. strukturierte Darstellung der Ist-Situation (UML)

Deliverables

- Ist-Analyse

Beispielhafte Dokumentstruktur für die Ist-Analyse

Die vorgestellte Dokumentstruktur zieht Fälle in Betracht, wo ein altes Software-System bereits besteht und durch ein neues abgelöst werden soll. Beachten Sie auch, dass ein System total nicht-automatisiert sein kann.

0 Übersicht
1 Derzeitige Lösung
  1.1 Funktionen
  1.2 Zuverlässigkeit
    Welchen Zuverlässigkeitsanforderungen genügt das System?
  1.3 Benutzer
    Wieviele Benutzer brauchen das System? Im Durchschnitt? Maximal? Gleichzeitig?
  1.4 Zeiten und Verbrauch
    Verbrauch von Resourcen. Lauf- und Einsatzzeiten.
  1.5 Schwachstellen
    Welche Schwachstrellen werden vom Auftraggeber identifiziert? Warum wird ein neue Lösung angestrebt?
2 Rahmenbedingungen
    Bestehende Hard- und Softwarebasis, Schnittstellen zu anderen Softwareprodukten.
3 Bewertung des Ist-Zustandes
    Wo liegt das Problem aus der Sicht des Informatikers? Inwiefern kann man die alte Lösung wiederverwenden, inwiefern muss man ein neues System bauen?

3.3 Soll-Konzept

Basierend auf den aus der Problemstellung verfügbaren Informationen und der in der Ist-Analyse festgestellten Probleme wird ein Soll-Konzept erstellt. Im Soll-Konzept wird unter anderem auch die Aufgabenstellung strukturiert wiedergegeben. Die Beschäftigung mit dem Soll-Konzept soll den Software Engineeer befähigen, die Informationslücken in der Problemstellung zu identifizieren. Das dient als Vorbereitung auf das Sammeln der exakten Requirements.

Tätigkeiten

- Erarbeitung einer neuen Lösung für die gestellte Aufgabe. Beschreibung der Eigenschaften der Lösung
- Beschreibung der Eigenschaften, die die Lösung aus Benutzersicht haben sollte. Keine Angaben zur Implementierung!
- ggf. Darstellung in strukturierter Form
- ggf. Erstellung von Benutzerszenarios (UML Use Cases)

— *How can we make clear what is expected? What is the difference with the next phase?*

— *Actually, we do not ask the students to deliver documents for both! Can lead to confusion!*

— *Why not just join together the first two under Analysis?*

Deliverables

- Soll-Konzept

Beispielhafte Dokumentstruktur für das Soll-Konzept

0 Übersicht
1 Lösungskonzept
  1.1 Funktionen
  1.2 Zuverlässigkeit
    Welche Zuverlässigkeit ist gefordert?
  1.3 Benutzer
    Wieviele Benutzer sollen das zukünftige System brauchen? Maximal? Im Durchschnitt?
  1.4 Zeiten und Verbrauch
  1.5 Änderungen
    Annahmen über Art, Umfang und Häufigkeit zu erwartender Änderungen
  1.6 Einführung der neuen Lösung
    Was muss bei der Einführung beachtet werden? Muss die Hard- und Software-Plattform angepasst werden? In welchen Schritten kann eingeführt werden?
2 Rahmenbedingungen
3 Anforderungskatalog für das Produkt
  3.1 Umgebungsbedingte Eigenschaften
  3.2 Qualitätsbedingte Eigenschaften
  3.3 Ausbaustufen
    Was ist die minimale Lösung, welche bereits einen wichtigen Teil des Gesamtlösung realisiert (und dem Auftraggeber bereits ausgeliefert werden könnte)? In welchen Schritten soll die Gesamtlösung angestrebt werden?

3.4 Requirementspezifikation - Pflichtenheft

Die Spezifiaktion der Anforderungen ist ein iterativer Prozess. Erstens wird man mit fortschreitendem Verständnis immer mehr Details und damit neue Fragestellungen erkennen. Zweitens werden von der Benutzerseite her immer neue Anforderungen gestellt werden

Tätigkeiten

- Sammeln der Anforderungen durch Interviews mit dem Kunden
- Identifizierung der kritischen Anforderungen, d.h. der Anforderungen von deren Erfüllung das Gelingen des gesamten Projektes abhängt. Untersuchung der Realisierbarkeit dieser Anforderungen.
- Erstellung von Prototypen um kritische und technologisch unbekannte oder besonders schwierige Aspekte des Systems besser zu verstehen und absch:atzen zu können.
- Erstellen von Protoypen für die Benutzeroberfläche. Der Kunde soll möglichst früh ein Bild vom zukünftigen System bekommen. Das erlaubt zu prüfen, ob die Kommunikation Benutzer-Ingenieur klappt. Auch lassen sich Benutzer von konkreten Beispielen meist zu weiteren Anforderungen inspirieren.
- Festlegung der verbindlichen Anforderungen an die Systembasis
- Festlegung der verbindlichen Anforderungen an Produktfunktionen und -qualität, sowie an Test und Anwenderdokumentation
- Priorisierung der Anforderungen

Besipielhafte Dokumentstruktur für das Pflichtenheft

0 Übersicht
Zusammenfassung wesentlicher Aussagen zum Produkt
1 Umgebungsbedingte Produkteigenschaften
1.1 Anforderungen durch die Schnittstellen der Systembasis
1.2 Anforderungen an die Austauschbarkeit
Welche Teile der Software- und Hardwarebasis werden voraussichtlich während der Lebenszeit des Systems ausgetauscht werden?
1.3 Berücksichtigung zu verwendender Bausteine
Werden bereits bestehende Module wiederverwendet? Werden Module eingekauft?
1.4 Anforderungen aus Vorschriften
Falls die Aufgabenstellung in einen Rahmen von gesetzlichen oder technischen Vorschriften fällt, müssen diese aufgeführt werden.
2 Qualitätsbedingte Produkteigenschaften
2.1 Anforderungen an Funktionen und Informationen
2.2 Anforderungen an die Zuverlässigkeit
Die Anforderungen an die Zuverlässigkeit müssen spezifisch erfolgen. ”Das System muss zu 99,99% zuverlässig sein” ist zu vage. “Das System muss zu 99,99% der Zeit eines beliebigen Kalenderjahres laufen” ist so genau, das Messungen zur Überprüfung dieser Anforderung planbar werden.
2.3 Anforderung an die Benutzbarkeit
Eine präzise Beschreibung der Eigenschaften wie z.B. Benutzerinterface, Speichermedien
Es reicht nicht aus, nur zu verlangen: Die Applikation muss benutzerfreundlich sein.
2.4 Anforderungen an das Zeit- und Verbrauchsverhalten
2.5 Anforderungen an die Wartbarkeit
2.6 Anforderungen für die Erweiterbarkeit

Dies ist wichtig falls das System auf lange Sicht geplant wird. In welchen Stufen wird das System vorraussichtlich erweitert werden und wie muss man sich darauf vorbereiten?

3 Test und Messungen
  3.1 Anforderungen an den Test
  3.2 Anforderungen an die Messungen
4 Einsatz
  4.1 Anforderungen an die Logistik der Installation
  4.2 Anforderungen an die Unterstützung der Migration
  4.3 Anforderung an die Logistik der Betreuung
5 Sonstige Leistungen

Das Ergebnis dieser Phase ist das Pflichtenheft. Dieses Dokument muss in schriftlicher Form vorliegen. Es stellt einen Vertrag zwischen dem Kunden und dem Softwarehersteller dar (vgl. ISO9001). Dieses Dokument muss mit dem Kunden diskutiert werden. Erst wenn der Kunde sein Einverständnis erklärt, kann mit der nächsten Phase begonnen werden. Das Pflichtenheft ist ein lebendiges Dokument. Während der Projektbearbeitung kann es zu Änderungen kommen. Diese müssen im Pflichtenheft festgehalten werden, sowie mit dem Kunden verhandelt werden (vgl. ISO9001).

4. Designspezifikation

Die Designspezifikation, aufgeteilt in die Phasen Grobdesign und Feindesign, wird in einem separtem Dokument niedergelegt. Es ist wichtig, dieses Dokument einem exakten Reviewprozess zu unterziehen. Das Dokument stellt wiederum einen Vertrag zwischen dem Kunden und dem Softwarehersteller dar. Es muss mit dem Kunden diskutiert werden. Erst wenn der Kunde sein Einverständnis erklärt, kann mit der Implementation begonnen werden. Ebenso wie das Pflichtenheft ist die Designspezifikation ein lebendiges Dokument. Während der Projektbearbeitung kann es zu Änderungen kommen. Diese müssen in der Designspezifikation festgehalten werden, sowie mit dem Kunden verhandelt werden.

Begriffe

Ein System wird in folgende hierarchische Teile aufgespalten (bottom-up):

- *Unit*: eine Klasse oder eine kleine Zahl von von sehr eng zusammenarbeitenden Klassen die eine abgeschlossene Einheit bilden.
- *Modul*: eine Reihe von Komponenten die zusammen einen oder mehrere verwandte Dienste erbringen.
- *Subsystem*: In grösseren Projekten kann es sinnvoll sein, Module zu Subsystemen zusammenzufassen. Bei kleineren Projekten ist die zusätzliche Hierarchiestufe nicht unbedingt notwendig.
- *System*: implementiert die gesamte geforderte Funktionalität

Deliverables

- Designspezifikation, bestehend aus Grob- und Feindesign

4.1 Grobdesign

Auf Basis des Pflichtenheftes werden in diesem Prozessabschnitt die fachlichen[1] Funktionen, die grobe technische Lösung für das Produkt, sowie die Konzepte für die Anwenderdokumentation, Test, Betreuung und Lieferung erarbeitet.

Tätigkeiten

- Entwurf der fachlichen Funktionen, des Datenmodells, der Oberfläche und der Schnittstellen des Produktes
- ggf. Darstellung in strukturierter Form
- Spezifikation der Eigenschaften zur Erfüllung der Qualitätsanforderungen
- Erstellung des Konzeptes für die Anwenderdokumentation (Arten von Dokumenten, Umfang, Zielgruppen)
- Erstellung des Test- und Messkonzepts für den Systemtest
  Tests müssen das gesamte Spektrum der Applikationsentwicklung abdecken. Es müssen auch Regressionstests vorgesehen werden, da diese garantieren, dass Änderungen am System nicht zu neuen Fehlern führen. In der Regel muss weitere Software entwickelt werden, die das Testen ermöglicht.
  Messkonzepte sind notwendig in Fällen wo ein System genau bestimmte Echtzeitanferoderungen erfüllen muss (Bsp. Telekommunikation).
- Spezifikation der DV-Schnittstellen zur festgelegten Systembasis, zu den verwendeten Bausteinen sowie zu anderen Produkten
- grobe DV-technische Strukturierung in Teilprodukte und Komponenten
- ggf. strukturierte Darstellung
  Verwendung von UML und erläuterndem Text
- Beschreibung der betrachteten Lösungsalternativen mit Begründung für die gewählte Lösung

Beispielhafte Dokumentstruktur für das Grobdesign

0 Übersicht
  Zusammenfassung wesentlicher Aussagen zur Lösung
1 Umgebungsbedingte Produkteigenschaften
  1.1 Austauschbarkeit
  1.2 Erfüllung von Vorschriften
    Welche Massnahmen werden getroffen, um die Vorschriften einzuhalten.
  1.3 Spezifikation der Schnittstellen zur Systembasis
  1.4 Zu verwendende Bausteine
    Welche bereits vorhandenen Bausteine werden verwendet? Wird ein Teil des Systems eingekauft?
2 Grobe DV-technische Lösung

1. “Fachliche” Funktion meint hier eine direkt auf die Aufgabenstellung bezogene Komponente des Systems. “Technische” Funktionen im Gegensatz dazu sind informatik-spezifische Funktionen oder Komponenten welche die Bereitstellung der fachlichen Funktion ermöglichen oder implementieren. Eine `AuftragsListe` wäre in diesem Sinn eine fachliche, ein `Stack` hingegen eine technische Klasse.

2.1 DV-technische Struktur des Produktes
Welche Architektur wurde gewählt?
2.2 Teilprodukte und Komponenten
In welche Subsysteme oder Module wird das System aufgeteilt?
2.3 Schnittstellen zu anderen Produkten
Wie werden die Schnittstellen zu anderen Systemen realisiert?
3 Qualitätsbedingte Produkteigenschaften
3.1 Vorgaben für Programmierung und Test
3.2 Vorgaben für die Dokumentation
3.3 Spezifikation der Funktionen und Daten
3.4 Zuverlässigkeit
3.5 Benutzbarkeit
3.6 Zeit- und Verbrauchsverhalten
4 Test und Messung
4.1 Testkonzept für den Systemtest
Wie wird sichergestellt, dass ein erfolgreich verlaufender Systemtest bedeutet, dass das System die Anforderungen erfüllt (Testen kann nur das Bestehen von Fehlern beweisen, nicht deren Fehlen!).
4.2 Messkonzept
5 Einsatz
5.1 Installationskonzept
5.2 Migrationskonzept
5.3 Betreuungskonzept
6 Sonstige Leistungen

4.2 Feindesign

In diesem Prozessabschnitt wird das DV-technische Grobkonzept bis auf Komponentenebene verfeinert. Die Testsysteme für Integrations- und Systemtest werden spezifiziert.

Tätigkeiten

- Verfeinerung des DV-technischen Grobkonzepts in mehreren Schritten bis zu den Komponenten; für jede Verfeinerungsebene Spezifikation der Anforderungen an die Teilprodukte/Komponenten, Spezifikation der Schnittstellen und Abläufe
- Spezifikation der Datenbasis des Produktes
- ggf. strukturierte Darstellung von Produkt und Datenbasis
- Spezifikation des Testsystems für Integrationstest
- Spezifikation des Testsystems für den Systemtest
- Spezifikation der Anwenderdokumentation
- ggf. Spezifikation von Installations-, Migrations- und Betreuungs-Tools
- Feindesign einschliesslich Spezifikation für
  - *Testsystem für Integrations- und Systemtest*
  - *Installations-, Migrations- und Betreuungs-Tools*
- Spezifikation der Anwenderdokumentation

Beispielhafte Dokumentstruktur für das Feindesign

0 Übersicht
Zusammenfassung wesentlicher Aussagen zur DV-Ausführung
1 DV-technische Lösung
1.1 Gesamtprodukt
Übersicht aller Module des Systems
1.2 Teilprodukte
Detaillierung der einzelnen Module bis auf Stufe Units.
2 Test und Messung
2.1 Testspezifikation für Integrations- und Systemtest
2.2 Testfälle
2.3 Spezifikation des Messsystems
3 Einsatz

5. Implementation

Nach Abschluss der Spezifikation des Designs kann mit der Implementation begonnen werden. Die Implementationsphase stützt sich dabei auf alle Dokumente, die bisher erstellt wurden, auch auf eventuell erstellte Protoypen.

Während der Implementation wird mindestens ein Codereview eines Modules verlangt. Dieser Review muss von einer anderen Gruppe durchgeführt werden.