

Tipps für Studentenarbeiten am IFS OST

Prof. Stefan Keller

IFS Institut für Software am OST Campus Rapperswil

Ausgabe 15, September/Oktober 2021

Inhaltsüberblick

1	Einleitung: Über dieses Dokument	2
1.1	Zweck dieses Dokuments.....	2
1.2	Zielpublikum	2
1.3	Ziele und Aufgaben	3
1.4	Vorgehenskonzept (Planung)	3
1.5	Lieferdokumente	4
1.6	Rechte an den Arbeiten	5
1.7	Diskussion und Tipps zur Projektabwicklung	5
2	Inhalt und Form der abzugebenden Arbeit	8
2.1	Kapitel ohne Kapitelnummerierung	8
2.2	Teil I: Bericht.....	10
2.3	Teil II: SW-Projektdokumentation	11
2.4	Teil II (ff.): Projektmanagement und Projektmonitoring	13
2.5	Anhänge (evtl. ohne Kapitelnummerierung)	13
	Anhang: Mindmap	15

1 Einleitung: Über dieses Dokument

1.1 Zweck dieses Dokuments

Dieses Dokument beschreibt die Dokumentation und einige organisatorische Aspekte einer Studentenarbeit ("Thesis") auf Stufe Fachhochschule, d.h. einer Studienarbeit, Bachelorarbeit, Vertiefungsprojekt oder einer Masterarbeit aus der Erfahrung des Autors und Betreuers bzw. Advisors. Es soll die **verbindlichen Leitfäden** ergänzen, die September 2021 vom den Studiengängen Bachelor Informatik und Masters MSE herausgegeben worden sind.

Es kann keine allgemeingültigen Regeln für Projektarbeiten geben, denn jedes Thema und jeder Standpunkt gegenüber und Blickwinkel auf dieses Thema führt zu Anpassungen. Es bleibt also den Autoren überlassen, wie die genaue **Kapitelstruktur** gestaltet sein soll.

Für die dafür nötigen Prinzipien des wissenschaftlichen Arbeitens und des Verfassens von Texten verweisen wir auf andere Literatur, namentlich:

- "Technical Writing and Research Advice" by Prof. Olaf Zimmermann: <https://www.ifs.hsr.ch/Technical-Writing-Advice.13194.0.html?&L=4> .
- Buch Justin Zobel "Writing for Computer Science", 3rd Ed., Chapter 5 "Writing a Paper", especially "A "Writing Up" Checklist" page 2 in the text (page 81 in PDF).

1.2 Zielpublikum

Dieses Dokument richtet sich primär an Studenten.

Die Studentenarbeit selber richtet sich an verschiedene Zielgruppen (variiert nach Projekt):

- *Alle* (also sowohl potentielle Auftraggeber als auch SW-Entwickler): Lesen und Beurteilen vor allem das Abstract (nur Text, genau eine Seite) und das Management-Summary (3-5 Seiten mit Grafik). Bei grösserem Interesse, lesen diese den Technischen Bericht, dann vielleicht - nach erfolgreicher Installation - das Tutorial.
- *Betreuer*: Er gibt Note in SA/DAs und achtet "auf alles" (... :->).
- *Gegenleser*: Er beurteilt das Endresultat (auf Papier), beeinflusst die Note. Er hält sich an die Gewichtung gemäss Aufgabenstellung; achtet aber in erster Linie meistens a) auf Formales und b) auf Inhalt.
- *Das Sekretariat*: verlangt vorab „Abstract“ - einen Beitrag zur Diplomarbeitsbroschüre der Abteilung. Das ist etwas Ähnliches wie das Management Summary aber auf zwei Seiten verkürzt (beschränktes Zielpublikum).
- Der Web-Besucher, d.h. Technische "Halblaien" oder allgemein Interessierte: Liest das fürs Web aufbereitete Management Summary.
- *Externe Partner* (eigentliche Auftraggeber): Diese schauen a) wie gut ihr Problem gelöst worden ist und b) können beurteilen, wie der Projektfortschritt war und wie auf Korrekturen reagiert wurde.
- *SW-Entwickler*: Ihn interessiert die Installation und folgende SW-Projektdokumente: Anforderungsspezifikation, Analyse, Resultate und Weiterentwicklung.

Projektmanagement und Projekt-Monitoring sind für den Auftraggeber interessant - aber nur im aktuellen Projekt. Design, Implementation (inkl. z.B. javadoc) und Test sind für SW-Entwickler interessant - aber nur im Folgeprojekt.

1.3 Ziele und Aufgaben

Die Arbeit verfolgt typischerweise folgende Ziele:

- Ein Software-Engineering-Projekt an einem realen Beispiel realisieren.
- Teamarbeit üben.
- Projekt-/Zeitmanagement üben.
- Erstellen einer vollständigen und hinreichenden Dokumentation üben.

Dies sind typische Aufgabenstellungen:

- Nach einem iterativen Prozess ein umfangreicheres, objektorientiertes Programm entwickeln.
- Durchführung der Arbeit in Teams (typischerweise 2er-Teams).
- Als Haupt-Programmiersprache ist in der Regel Java zu verwenden (weitere nach Absprache).
- Die Arbeit ist in mindestens zwei Iterationen durchzuführen, an deren Ende ein lauffähiger Prototyp vorliegen muss.
- Die Arbeit ist gemäss Vorgaben (siehe Dokumentationsvorgaben) zu dokumentieren.
- Die Dokumentation und der Programmcode sind auf dem neusten Stand zu halten und müssen für Projekt-Reviews mit den Dozenten zur Verfügung stehen. Die vollständige Dokumentation und der Programmcode sind an der Schlussabgabe abzugeben.
- Sie dürfen selbstverständlich von fremden Quellen profitieren, z.B. darf frei verfügbarer Code in das Programm eingebunden werden. Sie müssen allerdings ganz klar solche Fremdleistungen von Ihren Eigenleistungen abgrenzen, sowohl im Code (durch entsprechende Kommentare) als auch in der Dokumentation (kurze Strecken mit Anführungszeichen und Quellenangabe als Zitat kennzeichnen, längere Passagen durch die Wahl einer anderen Schrift, ebenfalls mit Quellenangabe). Vgl. dazu auch das entsprechende Kapitel unten.

1.4 Vorgehenskonzept (Planung)

Das Vorgehen bei Softwareprojekten geschieht grob in folgenden Phasen:

- Ausgangslage, Problemstellung. Idee und Konzepte festhalten:
- Einarbeitung in das Thema, Umfeld, Kick-Off-Meeting mit Partner
- Anforderungsspezifikation
- Realisierung vorbereiten:
- Projektmanagementplan, Projektmonitoring
- Technischer Bericht beginnen (Evaluation , "State-of-the-Art")
- Realisierung durchführen:
- Anforderungsspezifikation und Detailspezifikation (Anpassung / Ergänzung)
- Analyse
- Entwurf (Design)
- Entwicklung (Implementation) und Testplan
- Projektmonitoring, Entscheide
- Realisierung abschliessen:
- Schlussdokumentation (Technischer Bericht fertig stellen, Weiterentwicklung, Summary, Abstract, Pers. Bericht, Soll-Ist-Vergleich, Codeanalyse)
- Transition Vorbereitung: Poster, Präsentation, Benutzerdokumentation: Installation, Tutorial, Referenzhandbuch.
- Transition Durchführung: Präsentation der Arbeit (je nach den Vorgaben der Schule),

Abschlussmeeting mit Partner.

Laufend (ab Anforderungsspezifikation): Dokumentation.

Die Arbeit ist in mindestens zwei Iterationen durchzuführen, an deren Ende ein lauffähiger Prototyp vorliegen muss.

1.5 Lieferdokumente

Folgende Lieferdokumente umfasst üblicherweise eine Arbeit:

- Arbeit mit Titelblatt (gebunden oder "geringt")
- mit beigelegtem USB-Stick

1.5.1 Zu den Abstracts

Es werden drei Abstracts verlangt. Alle drei sollten inhaltlich natürlich aktuell sein und übereinstimmen:

- Das Abstract-Tool (sollte besser Broschüren-Abstract-Tool heissen), typischerweise 3 Abschnitte und 3 Bilder; nicht Teil der Doku. (von der HSR verlangt).
- Das Text-Abstract, nur Text max. eine Seite als Teil der Doku. (von der HSR verlangt).
- Das Management-Summary, ca. 3 Seiten mit 3 Bildern und u.a. Ausblick (von mir verlangt).

Ich empfehle mit dem Management-Summary zu beginnen. Die Abstracts werden erst ganz am Schluss der Arbeit erarbeitet bzw. vervollständigt.

1.5.2 Zur Arbeit

Total zählt das Inhaltsverzeichnis ca. 15 Haupt-Kapitel: (vgl. dort): Teil I Technischer Bericht (Einführung; Vision; "Stand der Technik", Umsetzungskonzept und Resultate) und Teil II SW-Projektdokumentation (Anforderungsspezifikation, Analyse, Design und Implementation, Projektmanagement und -Monitoring, Test, Resultate und Weiterentwicklung (!), Benutzerdokumentation, Anhänge). Das Fehlen des Unterkapitels Weiterentwicklung (im Teil II) ist als Mangel der SW-Dokumentation zu bewerten. Bewertet wird Umfang und Inhalt (Brauchbarkeit der Angaben, etc.).

1.5.3 Zum USB-Stick

Dieser enthält alle oben erwähnten Dokumente (vgl. Verzeichnis diplomarbeit/ oder thesis/, falls englisch). und mehr (vgl. Verzeichnis extras/, welches allfällige weitere externe Quellen oder Software enthält)!

Die Dokumente in diplomarbeit/ sind z.T. bereits einzeln vorgegeben (z.B. aufgabenstellung.pdf) oder sollten z.T. aufgeteilt werden, damit auch die Dokumente besser weiterverwendet werden können (z.B. referenzhandbuch.pdf. Darin enthalten sind auch die nicht zu druckenden Sitzungsprotokolle.

Zusätzlich zu den digitalen Dateien des gedruckten Dokuments, kommen u.a. folgende Dokumente hinzu (vgl. Verzeichnis last_minute/): Poster und Präsentations-Folien sowie allenfalls eine allerletzte Version der Applikation inkl. Source etc. (delivery/).

Der USB-Stick widerspiegelt natürlich auch die verwendete Software (Verzeichnis delivery/). Der USB-Stick könnte demnach folgende Dokumenten- und Verzeichnisstruktur enthalten:

```
+ myapp_1.1/
+ CHANGES.txt           (Change-Log mit Aenderungen zuoberst)
+ INSTALL.txt
+ LICENSE.txt            (z.B. mit Verweis auf BSD-License)
+ README.txt             (ASCII-Text oder Plain-HTML)
+---bin/                 (u.a. mit .bat, .sh, startup.exe, ...)
+---doc/                 (Helpfiles, html, pdf, ...)
+---lib/                 (u.a. mit .jar, ...)
+---plugins/             (z.B. ch/integis/export.jar, ...)
+---src/                 (z.B. build.xml und ch/integis/... etc.)
+---workspace/          (mit den Daten etc.)
+ diplomarbeit/
+ README.txt
```

```
+ abstract.pdf
+ aufgabenstellung.pdf
+ diplomarbeit.pdf
+ referenzhandbuch.pdf
+---doc_src/           (Dokumente und -Abbildungen im Originalformat)
+---project/           ( Projektdokumente, z.B. Sitzungsprotokolle)
+---tutorial/
+ extras/
+  README.txt
+ last_minute/
+  README.txt
+---poster/
+---presentation/
+---myapp_1.2/         (nach >2 Iterationen muss die App. ja >1.1 sein...)
```

1.6 Rechte an den Arbeiten

Wichtig: Dieser Abschnitt gilt als Ergänzung für entsprechenden Regelungen der Schule.

Die Urheberrechte bleiben ganz bei den Autoren. Die Nutzungs- und Vertriebsrechte hingegen müssen möglichst vorgängig bestimmt werden und zwar betreffend folgende Artefakte

1. Dokument (Thesis) – hier empfiehlt der Autor "CC BY 3.0 ch"
2. Software – hier empfiehlt der Autor typischerweise MIT (BSD oder ISC wären auch "nicht-viral" wie GPL sind aber weniger verbreitet). Die gilt natürlich nur, wenn bestimmte Lizenzen nicht von der bestehenden Software vorgegeben sind.
3. Daten - hier empfiehlt der Autor typischerweise Public Domain (CC0?) oder ODbL.

1.7 Diskussion und Tipps zur Projektabwicklung

1.7.1 Projektmanagement ist alles...

Hier einige einfache Hilfestellungen und Faustregeln:

Einige sagen, dass jedes Projekt anders sei und es daher kaum brauchbare Schätz- und Kontrollmethoden gäbe. **Aber.** Jedes Projekt lässt sich mindestens in Einheiten unterteilen, die Sie abschätzen und kontrollieren können!

Sie haben sicher schon einiges über Projektmanagement gehört: Auf welcher Stufe schätzen Sie sich selber ein? Wenn Sie auf Level 3 sind, dann sind sie schon gut. Mal sehen, ob Sie's weiter schaffen...

Erläuterung zur Abbildung unten:

- Niveau 3
 - Der Software-Prozess ist vollständig dokumentiert,
 - Verwaltungs- und technische Aspekte sind klar definiert,
 - Qualität wird kontinuierlich verbessert,
 - Es werden Reviews durchgeführt
 - Case Tools sind erst jetzt anwendbar (bei Level 1 oder 2 noch nicht!).
- Niveau 4
 - Qualitäts- und Produktivitätsziele sind vorgegeben,
 - Qualität und Produktivität werden fortwährend überwacht,
 - Es gibt statistische Qualitätskontrollen.
- Niveau 5
 - Kontinuierliche Verbesserung des Prozesses,
 - Es gibt dokumentierten Feedback,
 - Das Wissen wird von jedem Projekt zu anderen weitergereicht.

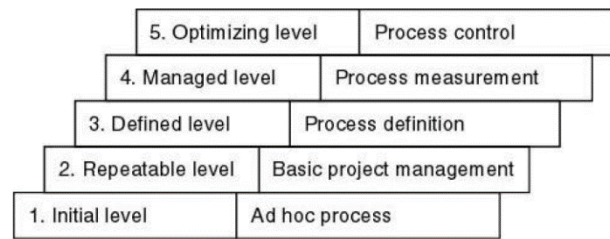


Abbildung: Levels (Niveaus) der Projektabwicklung.

1.7.2 Probleme

Bei Studien- und Abschlussarbeiten gibt es erfahrungsgemäss folgende typischen Probleme:

- Die Zeit ist meistens knapp und kann nie gut genug genutzt werden. Erfahrungsgemäss hat man zu Beginn der Arbeit den grössten Spielraum, der oft zuwenig genutzt wird.
- Dokumentation wird an den Schluss verschoben.
- Die Aspekte der Weiterentwicklung werden vernachlässigt... (siehe sep. Kapitel dazu unten).
- Oft wird zu lange alleine gegrübelt. Nutzen Sie das Internet und Fragen Sie den/die Kollegen und Betreuer nach spätestens mehr als einem halben Tag am gleichen Problem.
- Der Auftraggeber ändert die Anforderungen oder kann (will?) sie nicht festlegen. Ein geflügeltes Wort sagt, „Man gebe (spezifiziere) dem Anwender nicht was er will, sondern was er braucht“.
- Ein typisches Dilemma ist „Make or Buy“. Natürlich ist Selber-Machen oft interessanter. Und klar ist, dass die Einarbeitung von unbekannten Programmvorlagen und Bibliotheken Zeit kosten kann und mit Unsicherheiten verbunden ist (Dokumentation, Komplexität, zuwenig ausbaufähig). Andererseits bietet bestehende Software sicherlich den grösseren Funktionsumfang, der erst noch von anderen weiterentwickelt wird und (meistens) ausgetestet ist. Die Entscheidung „Make or Buy“ wird typischerweise mit einer Evaluation und wohldefinierten Kriterien gefällt.
- Sinnvollerweise baut man oft auf externe, vorhandene Software-Bibliotheken (Buy-Option oben). Dabei kann es vorkommen, dass die Dokumentation unvollständig oder die Software fehlerhaft ist. Aus eigener Erfahrung gilt dies nicht nur für Open Source Software.

1.7.3 Stolpersteine

Bei Studien- und Abschlussarbeiten gibt es erfahrungsgemäss folgende typischen Stolpersteine:

- Die (oft nachträgliche!) Dokumentation von (Design-)Entscheiden ist schwierig. Hilfe bietet sicher mal der Y-Approach bei grösseren Architektur-Entscheiden (vgl. unten).
- Es werden oft keine Alternativen angegeben und es wird zu wenig evaluiert (mit nachvollziehbaren Kriterien und Bewertung). Dies vor allem in den Dokumenten "Stand der Technik", Umsetzungskonzept, Design, Architektur oder Implementation.
- Die Einarbeitung in das Werkzeug (und ev. die neue "Sprache") kann nicht genug früh geschehen. Die Einarbeitung geschieht typischerweise direkt anhand des Themas, jedoch mit dem Ziel ein Prototyp zu erstellen
- Beim Beginn der Diplomarbeit, d.h. bei der definitiven Festlegung des Themas, soll der Prototyp neu programmiert oder mindestens komplett "ge-reviewt" werden.
- Sich möglichst bewusst machen, was die Möglichkeiten und Einschränkungen der eingesetzten Werkzeuge sind.
- Man kann sich bei der Arbeit einerseits auf Funktionalität (Funktionen der Applikation, Algorithmen) oder auf die Benutzerschnittstelle konzentrieren; selten für beides.

Dokumentation von (Design-)Entscheiden:

- Beispiel: "Architectural Decision #1 - In the context of 'UC1: fetch missions', facing 'NFR2', we

decided for PostGIS spatial queries to achieve a quick response. This approach is compatible with the KISS principle, which was also applied in scope of this work.”

- Quellen:
 - Olaf Zimmermann (ABB Corporate Research). Making Architectural Knowledge Sustainable–Industrial Practice Report and Outlook. Invited IEEE Software Speaker, SEI SATURN, 2012. [Online; Accessed 9 July 2017]. URL: <https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=31345>
 - AKM Homepage. [Online; Accessed 9 July 2017] <https://www.ifs.hsr.ch/index.php?id=13191&L=4>

1.7.4 Rechtschreibung und Quellenangaben

- Im Zeitalter des Internets können fremde Texte und Bilder (Medien, Code) noch einfacher als früher kopiert werden. Nutzen Sie die Arbeiten anderer aus (falls dies rechtlich geregelt ist), aber zitieren Sie die Quellen nach den Regeln der Kunst (vgl. unten).
- Es gilt die neue deutsche Rechtschreibung. Oft wird am Ende vergessen, die Rechtschreibe-Funktion einzuschalten oder zu beachten!
- Eine der häufigsten Fehler, welche bisher von kaum einem Rechtschreibe-Programm (und Deutschlehrer?!) korrigiert wird, ist die Regel 28 „Bindestrich zur Aneinanderreihung“. Das Deutsche kennt – im Gegensatz zum Englischen – Bindestriche, dies vor allem auch zur Vermeidung von Doppeldeutungen. Also: Bitte verwenden Sie **Bindestriche**, dort wo sie hingehören.

2 Inhalt und Form der abzugebenden Arbeit

Bei der Abgabe muss jede Arbeit folgende Inhalte haben:

2.1 Kapitel ohne Kapitelnummerierung

2.1.1 Impressum und Revision

Das Hauptdokument sollte auf der Titelblatt-Rückseite (oder Seite 2) je eine Angabe zur Erstellung und letzten Aktualisierung des Dokuments haben sowie ein Impressum (Kontakt-/Autoren- bzw. Dokumenten-Bezugsadresse im Internet, z.B. www.integis.ch > Projekte).

Die SW-Engineering-Dokumente selber können eigene Tabellen mit Historie-Informationen enthalten.

2.1.2 Abstract

Ein Abstract ist eine rein textuelle kurze Zusammenfassung der Arbeit. Der Abstract ist für die Recherche in grossen Dokumentensammlungen geeignet. Er umfasst nie mehr als eine Seite, typisch sogar nur etwa 200 Worte (etwa 20 Zeilen).

Der Begriff ‚**Kurzfassung**‘ ist zuwenig genau definiert; er soll wenn möglich vermieden werden.

2.1.3 Management Summary und Web-Publikation

Das Management Summary soll 2-5 Seiten umfassen sowie eine bis zwei Figuren enthalten. Es richtet sich an den „gebildete Laien“ auf dem Gebiet und beschreibt daher in erster Linie die (neuen und eigenen) Ergebnisse und Resultate der Arbeit. Die Sprache soll knapp, klar und stark untergliedert sein.

Grundlage für das Management Summary kann der Broschüren-Eintrag sein, den die Abteilung bei Diplomarbeiten jeweils früh verlangt, um eine Broschüre zu drucken. Das Management Summary dient als Vorlage für eine allfällige Web-Publikation.

Das Abstract und das Management Summary werden - zeitlich gesehen - gegen Schluss der Arbeit geschrieben und bilden zusammen mit den Schlussfolgerungen im technischen Bericht den am häufigsten gelesenen Teil der Arbeit. Diese Dokumente sollen daher am Sorgfältigsten ausgearbeitet sein.

Die folgenden Stichworte sollen die typische Struktur illustrieren, wobei die genaue Ausführung jeweils auf die spezifischen Bedürfnisse und Randbedingungen eines Projekts anzupassen ist. Diese Struktur kann auch für die Präsentation der Arbeit als "Richtschnur" dienen.

1. Ausgangslage
 - Warum machen wir das Projekt?
 - Welche Ziele wurden gesteckt (Kann-Ziele, Muss-Ziele)
 - Was machen andere / welche ähnlichen Arbeiten gibt es zum Thema?
 - Vorgehen: Was wurde gemacht? In welchen Teilschritten?
 - Risiken der Arbeit?
 - Wer war involviert (Durchführung, Entscheide usw.)?
 - Was konnte von anderen verwendet werden?
2. Ergebnisse
 - Was ist das Resultat?
 - Bewertung der Resultate, was ist Neuartig an der Arbeit?
 - Zielerreichung bezüglich Kann-/Muss-Zielen
 - Abweichungen (positiv und negativ) und kurze Begründung dafür
 - (Externe) Kosten der Arbeit?
 - Was ist der Nutzen (quantifizierbar/nicht quantifizierbar)?
3. Ausblick
 - Was hat man mit Durchführung des Projekts gelernt?
 - Verbleibende Probleme, (zukünftige) Gegenmassnahmen bez. Risiken
 - Was würde man anders machen, was ist weiter zu tun

Überschriften (Unterkapitel) ohne Nummerierung einsetzen!

2.1.4 Inhaltsverzeichnis

Die Abgabe ist so zu gliedern, dass die Kapitel und Inhalte klar erkenntlich und auffindbar sind. Es muss ein dokumentübergreifendes Inhaltsverzeichnis existieren.

Alle Dokumente müssen mit einer Seitennummerierung versehen sein, die eine eindeutige Referenzierung erlaubt, d.h., fortlaufende Zahlen oder Dokumentidentifikation plus Seitennummer (z.B. 47, oder Kapitel 1-47). Die Aufteilung der Kapitel in Dokumente (=Dateien) ist den Autoren überlassen.

Hier ein Beispiel eines Inhaltsverzeichnisses im Sinne einer Checkliste und kurz kommentiert:

- 1 Abstract
- 2 Management Summary
- 3 Inhaltsverzeichnis, evtl. Abbildungsverzeichnis
- 4 Aufgabenstellung (bei SA/DA vorgegeben)
- I *** Teil I Technischer Bericht**
- I 1 Einführung**
- I 1.1 Problemstellung, Vision
- I 1.2 Ziele und Unterziele
- I 1.3 Rahmenbedingungen, Umfeld, Definitionen, Abgrenzungen
- I 1.4 Vorgehen, Aufbau der Arbeit
- I 2 "Stand der Technik" (Was gibt es schon?)**
- I 2.1 Bestehende Lösungsansätze und Normen
- I 2.2 Kurzbeschreibung und Charakterisierung...
- I 2.3 Defizite (Hinweise auf Weiterentwicklung-, bzw. Verbesserungspotential)
- I 3 Bewertung (Evaluation)**
- I 3.1 Kriterien (Wie wird bewertet?)
- I 3.2 Schlussfolgerungen, eigener Lösungsansatz
- I 4 Umsetzungskonzept (des eigenen Lösungsansatzes)**
- I 4.1 Grobe Beschreibung des eigenen Lösungskonzepts
- I 4.x z.T. Wiederholung im Groben, z.T. Verweise auf Teil II-Kapitel
- I 5 Resultate, Bewertung und Ausblick**
- I 5.1 Zielerreichung
- I 5.2 Ausblick: Weiterentwicklung (nur wichtigste Punkte)
- I 5.3 Persönliche Berichte
- I 5.4 Dank (...)
- II *** Teil II SW-Projektdokumentation ("klassisches" RUP 2/HSR)**
- II 1 Vision (evtl. direkt auf Kap. 1.1 verweisen)**
- II 2 Anforderungsspezifikation**
- II 2.1 Anforderung an die Arbeit (bei Studien- und Diplomarbeiten)
- II 2.2 Use Cases (Success Scenario / Success Diagramm)
- II 2.3 System-Sequenzdiagramme
- II 2.4 Weitere Funktionen, die nicht erfasst wurden
- II 2.5 Nicht-funktionale Anforderungen (Rahmenbedingungen, evtl. Verweis auf 1.3)
- II 2.6 Detailspezifikation
- II 3 Analyse (Business Modell)**
- II 3.1 Domain Modell, Klassendiagramme (konzeptionell)
- II 3.2 Objektkatalog (Beschreibung der Konzepte, bzw. Entitätsmengen)
- II 4 Design (Entwurf)**
- II 4.1 Architektur
- II 4.2 Objektkatalog (Klassenkonzepte, Verantwortlichkeiten und Konsistenzbed.),
- II 4.3 Package- und Klassendiagramme (konzeptionell)
- II 4.4 Sequenzdiagramm, UI Design
- II 5 Implementation (Entwicklung) und Test**
- II 5.1 Implementation: Erläuterungen wichtiger konkreter Klassen
- II 5.2 Automatische Testverfahren,
- II 5.3 Manuelle Testverfahren, etc.
- II 6 Resultate und Weiterentwicklung**
- II 6.1 Resultate (evtl. nach oben in Teil I Kap. 5 eingliedern)
- II 6.2 Möglichkeiten der Weiterentwicklung
- II 6.3 Vorgehen (welche Mögl. würde man nun wie weiterentwickeln?)
- II 7 Projektmanagement (Planung, Soll) und**
- II 7.1 Prototypen, Releases, Meilensteine
- II 7.2 Team, Rollen und Verantwortlichkeiten
- II 7.3 Aufwandschätzung, Zeitplan, Projektplan
- II 7.4 Risiken
- II 7.5 Prozessmodell
- II 8 Projektmonitoring (Ist-Beschreibung, so ist es passiert)**
- II 8.1 Soll-Ist-Zeit-Vergleich
- II 8.2 Codestatistik (Zeilen: Kommentare, Klassen, Packages)
- II 8.x (Beschlussprotokolle nur auf CD)

II 9 **Softwaredokumentation** (evtl. ganz separat)
II 9.1 Installation
II 9.2 Tutorial / Benutzerhandbuch
II 9.3 Referenzhandbuch (Funktionen, Bedeutung von/Abhilfe bei Fehlermeldungen)
II ANHANG A **Inhalt der CD**
II ANHANG A **Glossar und Abkürzungsverzeichnis**
II ANHANG B **Literatur- und Quellenverzeichnis**

2.2 Teil I: Bericht

2.2.1 Problemstellung, Ziele, Einführung, Einleitung

Warum machen wir das Projekt? Wieso haben wir dieses Projekt bearbeitet?

2.2.2 Aufgabenstellung

Welche Ziele wurden gesteckt (Kann-Ziele, Muss-Ziele)

Die vom Betreuer abgegebene und unterschriebene Aufgabenstellung.

Bei Fortsetzungsarbeiten geht der Aufgabenstellung eine tabellarische Aufstellung voran (Einleitung), welche klar aufzeigt, was während der ersten Arbeit wurde.

2.2.3 Rahmenbedingungen

Meist vorgegeben.

2.2.4 Vorgehen, Aufbau der Arbeit

Vorgehen: Was wurde gemacht? In welchen Teilschritten? Risiken der Arbeit? Wer war involviert (Durchführung, Entscheide usw.)? Details in anderen Kapiteln.

Einführung in die Problem- und Aufgabenstellung. Übersicht über die übrigen Teile der Abgabe.

2.2.5 "Stand der Technik"

Zweck und Inhalt dieses Kapitels.

Was machen andere / welche ähnlichen Arbeiten gibt es zum Thema? Was kann von anderen verwendet werden?

Diese Einleitung soll für den Ingenieur irgendeiner Fachrichtung verständlich sein. Sie stellt die Aufgabe in einen grösseren Zusammenhang und liefert eine genaue Beschreibung der Problemstellung. Allfällige Vorarbeiten oder ähnlich gelagerte Arbeiten werden diskutiert.

Theoretische Grundlagen sind nur so weit auszuarbeiten, als dies für die Lösung der Aufgabe nötig ist (keine Lehrbücher schreiben). Die Erkenntnisse aus den theoretischen Untersuchungen sind wenn immer möglich direkt mit der Problemlösung zu verknüpfen.

2.2.6 Bewertung (Evaluation)

Evaluierten heisst Bewerten. Objektives Bewerten geschieht immer über Kriterien, qualitative (evtl. quantifiziert) und quantitative.

2.2.7 Vision, Umsetzungskonzept

Da drin steckt den Kern der Lösungs idee und des Konzeptes.

2.2.8 Resultate: Bewertung und Zielerreichung

Bewertung der Resultate. Vergleich mit anderen Lösungen.

Was ist Neuartig an der Arbeit? Was ist der Nutzen der Arbeit (quantifizierbar/nicht quantifizierbar)? Was sind die (externen) Kosten der Arbeit?

Zielerreichung: was wurde erreicht, was wurde nicht erreicht bezüglich Kann-/Muss-Zielen? Abweichungen (positiv und negativ) und kurze Begründung dafür.

2.2.9 Schlussfolgerungen und Ausblick

Die Schlussfolgerungen bilden zusammen mit der Zusammenfassung die wichtigsten Abschnitte eines Berichts und sollen daher am sorgfältigsten ausgearbeitet sein. Die Schlussfolgerungen enthalten eine Zusammenfassung und Beurteilung der Resultate (Vergleich mit anderen Lösungen, was wurde erreicht, was nicht, was bleibt noch zu tun, was würde man nun anders tun).

In den Schlussfolgerungen soll auch ein Ausblick auf das weitere Vorgehen bzw. auf die Bedeutung der erreichten Ergebnisse gegeben werden.

Weiteres zum Ausblick siehe Kapitel 2.3.9 Weiterentwicklung.

2.2.10 Persönliche Berichte und Dank (fakultativ)

Persönlicher Bericht der Studierenden zu ihren Erfahrungen bei der Arbeit. Bei Vorträgen ein sicher interessanter Teil; in der Diplomarbeit je nach dem.

2.3 Teil II: SW-Projektdokumentation

Hier folgen die Dokumente gemäss Software-Engineering-Vorgehen.

2.3.1 Überblick

Zweck und Inhalt dieses Kapitels

2.3.2 Vision

Verweis auf Teil I.

2.3.3 Anforderungsspezifikation

Enthält folgende mögliche Unterkapitel:

- Rahmenbedingungen (wenn nicht schon oben abgehandelt)
- Anwendungsfalldiagramm
- Hauptanwendungsfall
- Funktionale Anforderungen
- Nicht-Funktionale Anforderungen
- Weitere: Aktivitätsdiagramme, Fallbeispiele, Szenarien, Prototypen...

2.3.4 Analyse

Klassen-, bzw. Domainmodell (UML-Klassendiagramm).

2.3.5 Design (Entwurf)

Die Architektur soll eine (objektorientierte) Problem domain umfassen. Eine allfällig eingesetzte Datenbank darf diese Problem domain permanent speichern, nicht aber ersetzen.

Enthält folgende mögliche Inhalt und/oder Unterkapitel:

- Architekturentscheide: Siehe „Architectural Decisions“ von <https://www.ifs.hsr.ch/index.php?id=13195&L=4> v.a. die „Y-statements“. Weitere Quellen auf erwähnter Webseite sowie auf dem Skripteserver im Modul „Application Architecture“ von Prof. Olaf Zimmermann.
- Klassenverantwortlichkeiten (Klassenname, Verantwortlichkeit, Wissen, Tun, Abhängigkeiten); CRC-Diagramme?
- Sequenzdiagramm / Kollaborationsdiagramm

2.3.6 Implementation (Entwicklung)

Objektkatalog: könnte mit einem Thesaurus verwaltet werden!

Rest individuell.

2.3.7 Test

Enthält folgende mögliche Unterkapitel:

- Testverfahren automatisch (mit JUnit) und manuell (dokumentiert; z.B. GUI); durchnummeriert.
- Iterationen 1 ... X

2.3.8 Resultate

Resultate und Ergebnisse der Arbeit. Dieser Abschnitt richtet sich an den speziell für das entsprechende Fachgebiet interessierten Ingenieur. Er soll es ihm ermöglichen, die für die Problemlösung gemachten Überlegungen zu verstehen und nachzuvollziehen.

2.3.9 Weiterentwicklung

Möglichkeiten der Weiterentwicklung: Funktionen und mögliches weiteres Vorgehen. Weiterentwicklung scheint allgemein ein heikler Punkt zu sein in allen Projekten, besonders auch in SA/DA-Projekten:

In realen Projekten, im RUP-Prozess wird er kaum betont und auch für Sie und für die Auftraggeber scheint es ein Problem zu sein. Warum?

- Wenn man dort zuviel aufschreibt, dann könnte das als Manko aufgefasst werden; und Aufschreiben nimmt erst noch Zeit weg...
- Zudem ist aus Sicht Auftraggeber (ohne "Gegensteuer") eine zusätzliche Funktion besser bewertbar als der "Wert" eines robusten, sauberen Designs oder eines Refactorings (z.B. der Separierung der Objekt/Klassen-Zuständigkeiten).

Ein Test, wie "sauber" - namentlich: wie änderungsfreundlich - ein Design ist, zeigt sich beispielsweise in der Antwort auf folgende Frage: "Welche Interfaces muss man implementieren und/oder welche Komponenten/Klassen muss man erweitern (schlimmstenfalls anpassen), um eine weitere Funktionalität einzubauen? Z.B. ein weiterer Exportfilter?".

Nun ist es aber so, dass Weiterentwicklung den Auftraggeber sehr stark interessieren sollte, da es sich dabei um wiederkehrende Kosten (Wartung und funktionaler Ausbau) handelt. Denn:
Wer weiss besser und hat viele Ideen, was noch zu tun wäre, wenn nicht die Autoren des SA/DA-Projekts?

Daher gelten folgende Regelungen zur Weiterentwicklung (gilt auch für andere Projektarten):

- Weiterentwicklung ist obligatorisch und erscheint in zwei separaten Kapitel (ggf. Dokument): Im Technischen Bericht, Unterkapitel Resultate/Ausblick, und in der SW-Projektdokumentation.
- Weiterentwicklung im Technischen Bericht ist allgemein gehalten und daher weniger heikel. Wichtig ist die Aufzählung der hauptsächlichlichen weiteren möglichen funktionalen oder nicht-funktionalen Anforderungen.
- Weiterentwicklung in der SW-Projektdokumentation ist an Architekten / SW-Entwickler gerichtet wie jedes andere SW-Dokument.
- Gewichtung: Es wird separat bewertet und zusätzlich erst spät (SA/DA bei der Präsentation) vollständig abgegeben wird. Spät heisst bei DAs auf der Dokumentation der Präsentation (inkl. CD). Sein ungewichtetes Gewicht gegenüber der Gesamtdokumentation ist umfangmässig ca. 1/15. Zur Erinnerung siehe Unterkapitel Lieferdokumente mit ca. 15 Hauptkapitel-/Dokumenten.

2.3.10 Benutzerdokumentation

Installationsanleitung(en), Bedienungsanleitung(en) und Tutorien (evtl. in den Anhang)!

Vergessen Sie a) nicht den CD-Inhalt zu notieren und auch in die Doku. zu nehmen.

Testen sie die Installation mit realistischen Vorgaben!!

2.4 Teil II (ff.): Projektmanagement und Projektmonitoring

2.4.1 Allgemeines

- Normen
- Konfigurationsmanagement (Entwicklungs-Werkzeuge, Eingesetzte Software)

2.4.2 Projektmanagement

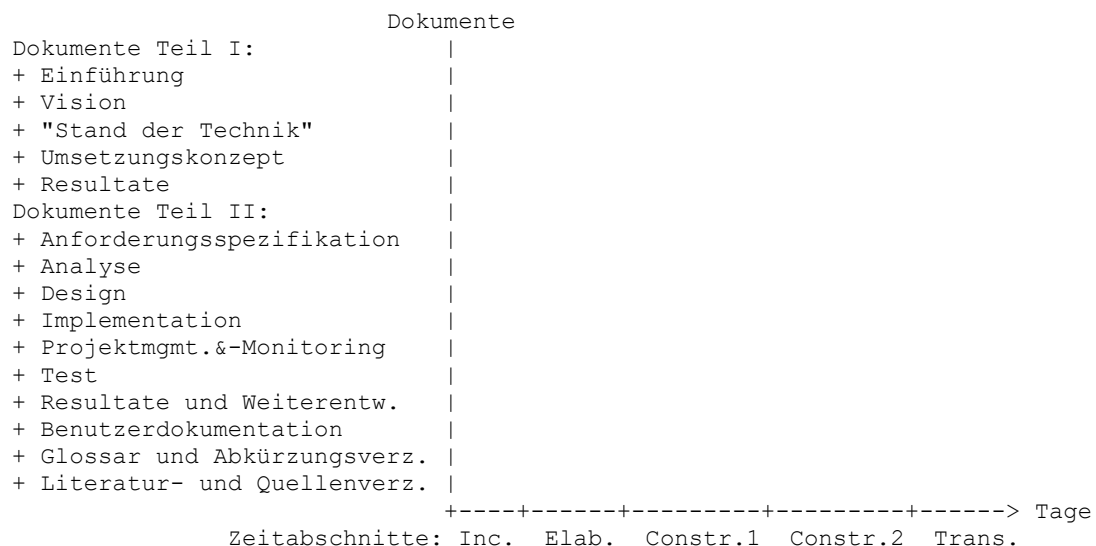
Enthält folgende mögliche Unterkapitel:

- Vorgehen
- Zeitplanung
- (Erreichen der Ziele siehe sep. Kapitel "Bewertung und Ausblick").

Zeitplanung: Die Zeitplanung orientiert sich an den Meilensteinen und ist nach folgender Idee strukturiert:

Inception => Elaboration => Construction 1 => Construction 2 => ... => Transition

Das ergibt folgendes Zeitdiagramm:



2.4.3 Projektmonitoring

Code-Analyse (Metriken).

2.5 Anhänge (evtl. ohne Kapitelnummerierung)

2.5.1 Glossar (fakultativ)

Je nach Umfang Teil des technischen Berichts oder als eigenes Dokument. Besonders bei Grundlagearbeiten ist das Glossar und/oder ein Verzeichnis der wichtigsten Abkürzungen sehr hilfreich.

2.5.2 Literatur- und Quellenverzeichnis

Im Literaturverzeichnis sind alle verwendeten Quellen (Bücher, Publikationen, URL, evtl. auch Hinweise auf Gespräche und Information, welche über die Computernetze beschafft wurde) aufgeführt. Typischerweise werden die Quellenangaben nummeriert (z.B. [1], [2], ...) und in der Reihenfolge geordnet, wie sie im Bericht vorkommen. Man kann die Quellen auch mit den (abgekürzten) Namen der Autoren und

dem Erscheinungsjahr bezeichnen (z.B. [Schueli90], [Shannon49], ...), wobei man die Einträge dann alphabetisch ordnet. Jede Referenz ist so anzugeben, dass sie möglichst einfach auffindbar ist (evtl. inklusive Seitenzahl, Bibliothek Bestellnummer). Eine Referenz, welche die allgemeine Grundlage für ein ganzes Kapitel bildet, wird im Titel bzw. in einer Fussnote aufgeführt. Für Referenzen aus dem Internet soll ein kommentierter URL angegeben werden.

Vgl. auch APA Style Guide to Electronic References, wie er u.a. von Google Scholar (<https://scholar.google.ch>) angeboten wird. Hier noch ein Tipp zum Zitieren von "simplen" Webadressen und "Direct quotes in APA Style" <https://www.scribbr.com/apa-style/direct-quotes/>.

Beispiele zu Quellenangaben:

Keller, S.F. (1994): Sustained spatial data management in real-world projects - a research focus. In: Nievergelt et. al (Hrsg.), International Workshop on Advanced Research in GIS (IGIS'94), Monte Verità, Ascona, Switzerland, february 28 - march 4 1994; Lecture Notes on Computer Science (LNCS) no. 884, International Series, Springer, 155-167.

Keller, S.F. (2000): INTERLIS und OpenGIS. In: Vermessung - Photogrammetrie - Kulturtechnik, 4/2000. [<http://www.vpk.ch>]

L+T - Bundesamt für Landestopographie (2000): "INTERLIS 2.0-Referenzhandbuch", Eidg. Vermessungsdirektion, Wabern. [<http://www.interlis.ch>]

Zadeh, L.A. (1965): Fuzzy sets. Information and Control. 8, 338-353.

Zimmermann, H.-J., J. Angstenberger, K. Lieven, R. Weber (1993): Fuzzy-Technologien: Prinzipien, Werkzeuge, Potentiale. VDI-Verlag GmbH, Düsseldorf.

Persönliche Kommunikation (2001) Telefonat vom 22.5.92 mit Herrn XY, Firma, Adresse, Telefonnummer.

Alternative Darstellung:

[1] C.E.Shannon, "Communication in the Presence of Noise", Proceedings IRE, Vol. 37, January, 1949, pp. 10-21.

2.5.3 Weitere Dokumente des Projekts (z.B. Normen, Programmierer-Dokumentation/javadoc)

Enthält eine Übersicht über die übrigen Dokumente gemäss Aufgabenstellung bzw. Software-Engineering-Vorgehen. Dies sind z.B., die Protokolle der Sitzungen.

Anhang: Mindmap

Dies ist eine Mindmap einer Studien-/Bachelorarbeits-Dokumentation (Vorschlag A. Rinkel/S.F. Keller vom Okt. 2001). Dieser ist noch nicht ganz abgeglichen mit dem vorliegenden Dokument. Das Mindmap beginnt rechts mit dem Bericht (roter Hintergrund) und ist zu verstehen, als Auswahl zu verstehen: wichtigste Punkte (die mit Begründung leer sein können) und erweiterbar (d.h. adaptierbar an individuelle Situationen).

