Bookman

Bookman
http://bookman.sf.net/
a1roelli@hsr.ch, michael.naef@switzerland.org

27. Oktober 2003 - 3. Dezember 2003

Inhaltsverzeichnis

Kapitel 1

Projekt Management Plan

1.1 Administratives

1.1.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	29. 10. 2003	adi
0.1.1	in LaTeXüberführt	31. 10. 2003	cal
0.2	Überarbeitung	3. 11. 2003	cal
0.2.1	Neue Sektion Dokumente, Tools überarbeitet	5. 11. 2003	cal
0.2.2	Überarbeitet, Zeitplan erstellt	5. 11. 2003	adi
0.2.2.a	Einige Fehler Korrigiert	7. 11. 2003	cal
0.2.3	Ins aktuelle Template integriert	11. 11. 2003	cal
0.3	Vollständig überarbeitet (siehe Protokoll vom 10. 11. 2003)	12. 11. 2003	cal
1	Entspricht der Verion 0.3 welche nach der Sitzung vom 17. 11 2003 zur Version 1.0 erklärt wurde		
1.0.1	UC - Dokumnet hinzugefügt	27. 11. 2003	cal
1.0.2	DA - Dokument hinzugefügt	3. 12. 2003	cal
1.1	Überarbeitetung für Fach PmQm Eingeflossen	3. 12. 2003	cal

1.1.2 TODO

was	bis wann	wer
NIL		

1.2 Einführung

1.2.1 Projektübersicht

Die Studienarbeit unter dem Titel “Bookman” ist Teil des gleichnamigen Opensource Projektes. Sie dauert vom 27. Oktober 2003 bis am 6. Februar 2004.

Für weitere Deatails siehe Aufgabenstellung.

1.2.2 Ziele

Ziel ist es, am Ende des Projektes eine gut funktionierende, durchdachte und für Erweiterungen offene Lösung für das verteilte Management von Bookmarks zu haben. Sie soll als Basis für eine weitere Entwicklung in der Opensource Community dienen können.

Am Ende des Projektes soll der Erfolg anhand einer Testumgebung demonstriert werden können.

1.2.3 Entwicklung des Projekt Management Plans

Zu Begin des Projektes wird der PMP erstellt und zu einer ersten Version ausgearbeitet. Diese deckt den Ablauf des Projetes bis zum Ende der ersten Phase (siehe 1.4.1) ab. Dies wurde mit Version 0.3 erreicht. Aufgrund der Resultate dieser Phase wird ein Erweiterter (vollständiger) PMP (PMP++) erstellt, der die Entwiklung bis zum Ende des Projektes beinhaltet.

Der PMP ist ein Working-Documnet und kann jederzeit verändert werden.

1.3 Oraganisatorisches

Hier werden die Organisation, Verantworlichkeiten und Abläufe in Zusammenhang mit der Organisation (Meetings) definiert.

1.3.1 Beteiligte Personen

Name	E-Mail	Kürzel	Typ
Adrian Rölli	a1roelli@hsr.ch	adi	Student / HSR
Michael Naef	mnaef@hsr.ch	cal	Student / HSR
Thomas Letsch	tletsch@hsr.ch	ThL	Betreuer / HSR

1.3.2 Zeitmanagement

Die geleistete Arbeit wird erfasst und nach Kategorien goerdnet abgelegt.

Von der Schule sind acht Lektionen pro Person und Woche Budgetiert. Dies beläuft sich total auf $8 * 1.5h * 15W * 2P = 360Ph$ Soll-Arbeitszeit.

1.3.3 Projekt Strktur

Da das Team während der SA aus nur zwei Personen besteht, verzichten wir auf eine Hierarchische Struktur. Folgende Verantwortlichkeinten werden zugeordnet:

Tätigkeit	Verantwortlichkeit
Dokumentation	adi, cal
CVS	cal
Projekt WebPage	cal
Sourceforge admin	cal
Zeiterfassung	adi
Implementation	adi, cal
Testen	adi
Qualitäts Management	adi

zu erweitern...

1.3.4 Wöchentliche Meetings

In der Regel finden Meetings der Projekt-Teilnehmer und des Betreuers einmal wöchentlich statt. Dabei werden der Projektfortschritt, das weitere Vorgehen und allfällige Probleme besprochen.

Über die Sitzungen wird Protokoll geführt (für Details siehe Protokoll vom 27. 10. 2003).

1.3.5 Referenzliste

Richtlinien Studienarbeiten
http://informatik.hsr.ch/themen.cfm?link=themen.cfm
&gruppe=2&kategorie=24&page=Content/Gruppen/Stud/

Faecherkatalog/Studarb/aktuell.html&unterMenu1=43&count=2&
news=news/news.html

Inhalt und Form der abzugebenden Arbeit
http://informatik.hsr.ch/Content/Gruppen/Stud/Faecherkatalog/
Studarb/DokuAnleitung.pdf

Termine
http://informatik.hsr.ch/Content/Gruppen/Stud/Faecherkatalog/
Studarb/termine.html

1.3.6 Mittel

Das Projekt besitzt kein eigenes Budget. Die verwendete Hard- und Software wird von der Hochschule Rapperswil oder den Teammitgliedern zur Verfügung gestellt.

1.4 Ablauf

Hier wird der Ablauf mit allen Faktorem (Doku, Milestones etc...) soweit definiert, wie es in PMP 1.0 möglich sein wird.

1.4.1 Phasen

Das Projekt kann grob in drei Pasen eingeteilt werden:

1. Initiierung
2. Entwicklung
3. Konsolidierung

Initiierung

Diese Pahse am Anfang des Projektes beinhaltet im wesentlichen das Projekt-Setup, die Analyse und die System-Architektur.

Aus dem Setup resultiert der *Projekt Management Plan* (PMP) in der Version 1, der das Projekt Management bis zum Ende der erste Phase abdeckt.

Die nachfolgende Analyse dient zur Spezifikation der Anforderungen an das Projekt. Daraus resultiet die *Anforderungs-Spezifikation*.

Die *System-Architektur* beendet die erste Pahse. Aufgrund der Anfordeungen können die Mittel zum erreichen des Ziels festegelgt werden (Technologien, Grobdesign, Teile sind identifiziert etc.). Dies ermöglicht eine genauere Planung des Rests des Projektes. Es resultiert nebst dem SA-Dokumnet der erweiterte PMP (PMP++).

Die Phase wird durch einen internen Review der geleisteten Arbeit (insbesondere der Dokumentation abgeschlosen).

Entwicklung

Diese Phase wird genauer im PMP++ definiert.

Grundsätzlich findet hier die eigentliche Entwicklungsarbeit, auf die sich auch die SE Modelle (XP, RUP, UCD etc.) beziehen, statt.

Konsolidierung

Diese Phase wird genauer im PMP++ definiert.

Diese sehr kurze Phase am Schluss des Projektes dient zur geordneten Beendigung der Arbeit (Organisatoriesches, Doku, Reviews, Verifikation etc.). Sie kann auch als Puffer für liegengebliebenes dienen.

1.4.2 Meilensteine

Meilensteine definieren Fixpunkte im Projekt, anhand dessen wir den Fortschritt messen können. Sie werden im Folegenden in der Form Mn definiert.

M1: Offerte

Dieser Meielenstein gilt als erreicht wenn die Anfspez., die SA und der PMP++ stehen. Sind diese Voraussetzungen erfüllt, wäre es möglich, einem fiktiven Kunden *eine Offerte zu machen.*

M$n-1$ Entwicklung beendet

Dieser Meilenstein gilt als erreicht, sobald die Entwicklungsphase abgeschlossen ist (was nicht heist, dass die Entwicklung im Rahmen der Konsolidierungsphase nicht weitergeht).

Mn Projekt beendet

Das Projekt gilt als beendet und somit dieser Meilenstein als erreicht, wenn die Abgabe abgeschlossen ist, dh. alle Produkte der Arbeit an den richtigen Stellen eingetroffen sind.

1.5 Dokumentation

Alle Aspekte des Projekts werden schriftlich Dokumentiert. Redundanzen in der Dokumentation sind zu vermeiden um die Gefahr von Inkosistenzen zu verringern.

1.5.1 Dokumente

Hier werden die genereiten Dokumente und ihr Inhalt definiert. Die Beschreibung hier dient gleichzeitig als Zusammenfassung (*abstract*) der Dokumente.

Projekt Management Plan (PMP)

Der PMP behandelt das Projektmanagement. Er dokumentiert die Organisation, Organisatorisches, Projektphasen, Meilensteine und Resultate.

Anforderungs-Spezifikation (AnfSpez)

Die AnfSpez definiert welche Anforderungen an das Projekt (bzw. dessen Output) gestellt werden. Sie definiert das Minimum welches erreicht werden muss um das Projekt erfolgreich abzuschliessen.

Konventionen (SG)

Definiert die Richtlinien und Konvetionen welche nötig sind, für eine gemeinsame Arbeit mehrerer Personen an Bookman. Das Dokument beinhaltet:

- Coding Style Guides
- Doku Style Guide
- Versioning Richtlinien
- Datei Hierarchien
- cvs Organistion und Regeln

Glossar (GLO)

Im Glossar werden die verwendeten Terme definiert und erklärt.

Typographische Konventionen (Typo)

Das Typo Dokument versteht sich als Kurzfassung des Doku Style Guide's spezifisch für den Leser. Es erklärt die Typografischen Konventionen.

Lizenzen (Lic)

Definiert, was welchen Lizenzen unterliegt und enthält alle für Bookman relevanten Lizenzen.

Protokolle (Proto)

Beinhaltet alle Protokolle der wöchentlichen Meetings und Protokolle allfälliger zusätzlicher Sitzungen.

Use Cases (UC)

Definiert die wichtigsten Use Cases des Dienstes. Für die Anforderungsspezifikation werden nur die Wichtigsten aus der Sicht eines Users auf eine Blackbox, sehr abstrakt und grob in einer Form ähnlich "Brief" gehalten. Während der System Analyse können sie weiter zu "Casual" oder "Fully Dressed" entwickelt werden (ev. in einem eigenen Dokument).

Domain Analyse (DA)

Die Domain Analyse Beinhaltet eine genaue Untersuchung der Problem Domain. Für die Anforderungs-Spezifikation dient sie als input Dokumnet.

1.5.2 Produktdokumentation

Teil des Projektes ist auch die Dokumentation der Arbeit für den Anwender.

Da es sich um ein weiterzuführendes Opensource Projekt handelt, gehört der Dokumentation des Codes und insbesondere der Schnittstellen und Libraries besonderen Augenmerk. (siehe auch Richtlinien::Coding Style Guide).

Anwender Dokumentation

→PMP++

Programmier Dokumentation

→PMP++

1.6 Qualitäts Management

Diese Kapitel behandelt Belange der Qualitätssicherung, Verifikation und Validation.

1.6.1 Qualitätssicherung

Um durchweg konstante, konsistente und saubere Arbeit sicherzustellen, existieren verschiedene Mechanismen zum Vorgehen und der Überprüfung der Resultate.

Konventionen sollen Konstanz und Konsistenz innerhalb des Projektes ermöglichen (siehe 1.5.1). Die für die Phase benötigten Richtlinien müssen vorher oder zu Begin der entsprechenden Phase definiert werden. Sie können bei Bedarf während den Arbeiten erweitert oder geändert werden.

Reviews sichern die Konsistenz und Konformität der geleisteten Arbeiten. Sie werden entweder intern oder zusammen mit nicht beteiligten Personen durchgeführt. Ein Review wird mit einem Protokoll abgeschlossen, Korrekturen werden während des Reviews angebracht oder zur späteren Korrektur im Protokoll vermerkt.

Tests der erstellten SW und Module stellen deren Funktion und Konformität mit den betreffenden Spezifikationen sicher. Der Inhalt der Tests wird in den entsprechnden Dokumenten definiert (→PMP++). Über die Tests wird Protokoll geführt.

1.7 Risikomanagement

Die Durchführung des Projektes unterliegt gewissen Risiken. Das folgende Kapitel befasst sich mit diesen und deren Minimierung.

1.7.1 Risiken

Im folgenden definieren wir die identifizierten Risiken. Dabei gilt:

Auswirkungen:	was passiert bei Entritt des Ereignisses?
Massnahmen:	Was kann getan werden um • dem Eintreten vorzubeugen • die Auswirkungen bei Eintritt zu minimieren.
Risiko:	Ausmass des Schadens ohne Einsatz der Massnamen. (engl. "harm")
Wahrscheinlichkeit:	Wahrscheinlichkeit des Eintritts.

Längerer Ausfall eines Teammitgliedes

Auswirkungen:	Verzögerung im Zeitplan
Massnahmen:	Absenzen vorausplanen; Reserven einplanen
Risiko:	mittel
Wahrscheinlichkeit:	klein - mittel (Militärdienst)

Kürzerer Ausfall eines Teammitgliedes

Auswirkungen:	Verzögerung im Zeitplan (kurzfristig)
Massnahmen:	Absenzen vorausplanen; Reserven einplanen; Alle Teammitglieder sind über aktuellen Stand Informiert.
Risiko:	klein
Wahrscheinlichkeit:	mittel

Falsche Einschätzung der Komplexität

Auswirkungen:	Nichterfüllen der Anforderungen
Massnahmen:	gründliche Einarbeitung in die entsprechenden Gebiete
Risiko:	mittel
Wahrscheinlichkeit:	klein

Fehlende Infrastruktur (HW, SW, Lokalitäten, etc.)

Auswirkungen:	Verzögerung im Zeitplan; Nichterfüllen der Anforderungen
Massnahmen:	Infrastruktur frühzeitig organisieren
Risiko:	mittel
Wahrscheinlichkeit:	klein

Ausfall von HW

Auswirkungen:	Verzögerung im Zeitplan; Verlust von Daten
Massnahmen:	Daten verteilen (cvs, Backup)
Risiko:	mittel
Wahrscheinlichkeit:	klein

Ausfall des Daten-Repository's

Auswirkungen:	Verzögerung im Zeitplan; Verlust von Daten
Massnahmen:	regelmässiges Backup der Daten.
Risiko:	Gross
Wahrscheinlichkeit:	klein

Kapitel 2

Anforderungs Spezifikation

2.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	19. 11. 2003	cal
0.2	Anforderungen hinzugefügt	20. 11. 2003	adi
0.2.1	Use Cases in separates Dokumnet ausgelagert	27. 11. 2003	cal
0.3	komplett überarbeitet	03. 12. 2003	adi

2.1.1 TODO

was	bis wann	wer
Inhalt hinzufügen	-	-

2.2 Allgemeine Beschreibung

Aufbauend auf der Aufgabestellung werden hier die allgemeinen Anforderungen für das Produkt festgelegt.

2.2.1 Umfeld

Das Projekt Bookman dreht sich um die Verwaltung von Bookmarks in einem browserunabhängien Format. Diese Bookmarks können als Sets auf den Dienst herauf, und vom Dinst heruntergeladen werden.

2.2.2 Funktionen des Dienstes

Jedes Bookmark gehöhrt zu einem Set. Sets können mehrere Bookmarks beinhalten. Ein Benutzer soll mehrere Sets erstellen und verwalten können.

Die verschiedenen Sets sollen wiederum zu Super-Sets gemerget werden können.

herunterladen

Die Bookmarks können als ganze Sets vom Dienst lokal heruntergeladen und ebenfalls wieder auf den Dienst heraufgeladen werden.

2.2.3 Benutzer

Potentielle Anwender von Bookman sind primär durchschnittliche oder fortgeschrittene Benutzer, welche

- mit mehreren Browser Applikationen
- auf mehreren Maschinen oder Platformen

arbeiten.

2.2.4 Einschränkungen und Besonderes

2.2.5 Voraussetzungen und Abhängigkeiten

Bookman wird für Linux und Windows entwickelt. Das Endprodukt muss auf beiden Systemen lauffähig sein. Dies gilt nicht für den „Server seitigen“ Dienst, welcher auf mindestens einem System lauffähig sein muss.

2.3 Anforderungen

2.3.1 Funktionale Anforderungen

Die Use Cases zu den funktionalen Anforderungen befinden sich im Dokument →UC.

Die folgenden Scenarien müssen mit Bookman bewältigt werden können:

- Bookmarks eines Browsers als Set auf den Dienst heraufladen.
- Ein Set von Bookmarks vom Dienst zum Browser herunterladen, damit mit diesen Bookmarks gearbeitet werden kann.
- Aus mehreren Sets von Bookmarks ein neues Set erstellen.
- Sets löschen.

2.3.2 Nichtfunktionale Anforderungen

Beim Händling der Bookmarks mit Bookman darf kein Datenverlust auftreten. Das heisst, dass zB Opera-Bookmarks auch nach dem x-ten mal Mergen immer noch alle Informationen enthalten, welche sie beim ersten Übergang in ein Bookman-Set enthielten.

Da mehrere Sets zu einem neuen Set zusammengefasst werden können, können Mehrfacheiträgen vorkommen. Die Funktionalität des Dienstes darf dadurch nicht beeinträchtigt werden.

Performance Anforderungen

Bezüglich der Performance und dem verursachten Netzwerk-Traffic werden keine Anforderungen gestellt.

Bedienung

Das Bookmark-Tool soll mit Maus und Tastatur zu bedienung sein.

Umgebung

Das Bookmark-Tool muss auf den HSR standart Installationen lauffähig sein. Dies ist für Windows „Windwos 2000“ und für Linux „RedHat 6.0 ???????????“.

Der „Server seitige“ Dienst muss auf mindestens einer der beiden Umbebungen lauffähig sein.

Kapitel 3

Use Cases

3.1 Administratives

3.1.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	26. 11. 2003	cal
0.1.1	Viele Use Cases Hinzugefügt und beschrieben	26. 11. 2003	cal
0.1.2	Use Case Diagram hinzugefügt	27. 11. 2003	cal

3.1.2 TODO

was	bis wann	wer
UC komplettieren und Überarbeiten	W48	cal, adi
UC Diagramm erstellen und einfügen	ASAP	cal

3.2 Use Case Diagramm

Abbildung 3.1: Übersicht der Use Cases

ma

3.3 Use Cases

3.3.1 Beispiel Use Case

Ziel

Was soll durch/in diesem Use Case erreicht werden.

Beschreibung

Wie erreichen wir das? Bemerkungen.

Preconditions

Postconditions

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte tun
2	System	reagiert

Erweiterungen

Welche zusätzlichen sub – Use Cases können/dürfen/müssen durchlaufen werden?

3.3.2 Bookmark Hinzufügen Simple

Ziel

Einen neuen Eintrag in der Bookmarkliste ohne besondere Angaben schnell hinzufügen.

Beschreibung

Dieser UC dient dazu ein Bookmark schnell und ohne viel Aufwand in das Set aufzunehmen. Insbesondere werden keine zusätzlichen Meta-Informationen wie Kategorie(en) oder eine Beschreibung aufgenommen sondern es werden (konfigurierbare?) default Werte angenommen.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, [Netzverbindung]).

Postconditions

Eintrag ist hinzugefügt.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Bookmark zu einer bestimmten (aktuellen) URL hinzufügen.
2	System	fügt URL mit default werten in seine Liste ein.

Erweiterungen

Keine. Allenfalls UC 3.3.3

3.3.3 Bookmark Hinzufügen Complex

Ziel

Einen neuen Eintrag in der Bookmarkliste mit zusätzlichen Meta-Informationen hinzufügen.

Beschreibung

Dies sollte der Standard–UC zum Hinzufügen von Bookmarks sein. Der Benutzer gibt zusätzlich Informationen zum Bookmark welche anstelle der default Werten aus Use Case 3.3.2 verwendet werden.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, [Netzverbindung]).

Postconditions

Eintrag ist mit Entsrechenden Werten hinzugefügt.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Bookmark zu einer bestimmten (aktuellen) URL mit *speziellen* Meta-Informationen hinzufügen.
2	System	System bietet dem User die Möglichkeit die Meta-Information zu spezifizieren.
3	User	gibt diese an soweit er will.
4	System	übernimmt die Daten und legt sie entsprechend ab.

Erweiterungen

Die Synchronisations UCs, insbesondere UC 3.3.13 können allenfalls folgen.

3.3.4 Bookmark ändern

Ziel

Eine oder mehrere Eigenschaften eines bestehenden Bookmark-Eintrages sollen geändert werden.

Beschreibung

Der User möchte Meta-Informationen eines Eintrages im Dienst ändern oder Hinzufügen (nachfolgend nur "ändern" genannt).

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, [Netzverbindung]). sowie einen Bestehenden Bookmark-Eintrag.

Postconditions

Die Eintrag ist gemäss den Vorstellungen des Benutzers geändert.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Eintrag ändern.
2	System	gibt Ihm eine Auswahlwöglichkeit.
3	User	Wählt den zu änderneden Eintrag
4	System	bietet User Änderungsmöglickeit.
5	User	bringt seine Änderungen an.
6	System	übernimmt die Änderungen in den Datenbestand.

Erweiterungen

UC 3.3.7 kann benutzt werden um einen nicht vorhandene Kategorie zu erstellen.

3.3.5 Bookmark löschen

Ziel

Das Bookmark soll vom Set entfernt werden.

Beschreibung

Das Bookmark muss soll aus dem Set entfernt werden.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, [Netzverbindung]). sowie einen Bestehenden Bookmark-Eintrag.

Postconditions

Der Eintrag ist entfernt.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Bookmark entfernen.
2	System	gibt Ihm eine Auswahlwöglichkeit.
3	User	Wählt den zu entfernenden Eintrag
4	System	entfernt den Eintrag im Datenbestand.

Erweiterungen

Keine?

3.3.6 Passwort ändern

Ziel

Der User will ein neues Zugangspasswort für seinen Bookman Account setzen.

Beschreibung

Der User ändert sein Account Passwort nachdem er seine Identität und Authorisation nachgewiesen hat. Diese Operation erfordert zwingend eine Verbindung mit dem Server.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung).

Postconditions

Das Passwort ist auf den vom User neu definierten Wert gesetzt.

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte sein Account Passwort ändern.
2	System	System erbittet neues Passwort
3	User	teilt neues Passwort mit.
4	System	System ersetzt altes mit neuem Passwort.

Erweiterungen

UC 3.3.16 ist zwingend erforderlich.

3.3.7 Kategorie Hinzufügen

Ziel

Eine neue Kategorie soll im System registriert werden.

Beschreibung

Wie erreichen wir das? Bemerkungen.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung).

Postconditions

Die neue Kategorie ist vorhanden.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte eine neue Kategorie hinzufügen
2	System	erbittet Spezifikation der Kategorie
3	User	teilt Daten mit
4	System	erstellt neue Kategorie

Erweiterungen

Keine?

3.3.8 Kategorie Bearbeiten

Ziel

Eigenschaften einer Kategorie sollen geändert werden.

Beschreibung

Eine Kategorie kann sich ändern (zB. der Name).

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung). Eine Kategorie ist vorhanden.

Postconditions

Änderungen sind wirksam.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Eigenschaften einer Kategorie ändern
2	System	bietet Auswahl an
3	User	wählt zu ändernde Kategorie
4	System	bietet Änderungsmöglichkeit
5	User	ändert was er möchte
6	System	verarbeitet und Speichert Änderung

Erweiterungen

Keine?

3.3.9 Subset Definieren

Ziel

Aus dem Set soll mittels Mengenoperationen mit den Kategorien ein Subset definiert werden.

Beschreibung

Ein Subset ist eine Auswahl der zur verfügung stehenden Bookmarks. Es wir definiert indem auf die Kategorien Menegnoperationen angewandt werden. Die Definition umfasst die Mathematische Regel (Menegenoperation) zur Bildung eines Subsets.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung). Eine Kategorie ist vorhanden (vorzugsweise mehrere...).

Postconditions

Ein zusätzliches Subset ist definiert.

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte ein Subset definieren.
2	System	System erbitet Regel zur Bildung des Subsets. sowie MetaInformation dazu (Name?).
3	User	teilt dem System Regel und Metainfo mit.
4	System	speichert die Regel und Metainfo.

Erweiterungen

UC 3.3.7 ??

3.3.10 Subset Bearbeiten

Ziel

Die Definition des Subsets soll geändert werden.

Beschreibung

Die Regeln oder die Metainfo kann angepasst werden.

Preconditions

Subset ist vorhanden.

Postconditions

Subset ist geändert.

Primary Actor

User

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Subset ändern.
2	System	bietet Subset Auswahl an.
3	User	teilt System mit welches Subset geändert werden soll
4	System	bietet Änderungsmöglichkeit.
5	User	Ändert Informationen
6	System	speichert Änderungen

Erweiterungen

UC 3.3.7 ??

3.3.11 Konflikt lösen

Ziel

Eine Konfliktsituation kann sauber gelöst werden.

Beschreibung

Eine Konfliktsituation kann auftreten, wenn Beim Synchronisieren nicht automatisch festgestellt werden kann welche von zwei Informationen die gültige(re) ist. Hier sind wir auf die Hilfe biologischer Algorithmen und Inteligenz angewiesen.

Preconditions

Bein Synchronisieren (UC 3.3.12, 3.3.13, 3.3.14, 3.3.15,) ist ein Konflikt aufgetreten.

Postconditions

Das System weis wie es den Konflikt lösen kann.

Primary Actor

User? System?

Main Success Scenario

#	Actor	User Intention or System Reaction
1	System	informiert den User über den Konflikt (Zeigt ua. die Konflizierenden Datensätze).
2	System	bietet dem User verschiedene Lösungsmöglichkeiten an (1st winns, 2nd winns, manual etc..).
3	User	entscheidet wie er den Konflikt lösen möchte und teilt dem System die dazu nötigen Informationen mit.

Erweiterungen

Keine? (Speziallfälle für BM, Set und Kat?)

3.3.12 System Synchronisieren

Ziel

Der Client und der Server gleichen Ihre Datenbestände ab.

Beschreibung

Ziel ist es, die Daten auf dem Server und dem Client zu synchronisieren. Die Systemsynchronisation beinhaltet den abgleich aller gespeicherten Daten: Bookmarks (UC 3.3.13), Subsets (UC 3.3.14) und Kategorien (UC 3.3.15). Dabei wird bei sich unterscheidenden Einträgen versucht automatisch den Gültige(re)n zu finden und den Ungültige(re)n damit zu aktualisieren.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung) sowie Bookmarks zum synchronisieren.

Postconditions

Der server und der Client haben konsistente Bookark–Sets oder Subsets.

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte Informationen synchronisieren.
2	System	überprüft Informationen auf dem Server und Client
3	System	überschriebt auf beiden Seiten ungültige mit gültigen Informationen.
4	System	Informiert User übder den Erfolg

Erweiterungen

Wenn nicht entschieden werden kann welche Informtion die gültige ist tritt eine Konfliktsituation ein (UC 3.3.11).

3.3.13 Bookmarks Synchronisieren

Ziel

Was soll durch/in diesem Use Case erreicht werden.

Beschreibung

Wie erreichen wir das? Bemerkungen.

Preconditions

Postconditions

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte tun
2	System	reagiert

Erweiterungen

Welche zusätzlichen sub – Use Cases können/dürfen/müssen durchlaufen werden?

3.3.14 Set Synchronisieren

Ziel

Was soll durch/in diesem Use Case erreicht werden.

Beschreibung

Wie erreichen wir das? Bemerkungen.

Preconditions

Postconditions

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte tun
2	System	reagiert

Erweiterungen

Welche zusätzlichen sub – Use Cases können/dürfen/müssen durchlaufen werden?

3.3.15 Kategorien Synchronisieren

Ziel

Was soll durch/in diesem Use Case erreicht werden.

Beschreibung

Wie erreichen wir das? Bemerkungen.

Preconditions

Postconditions

Primary Actor

Main Success Scenario

#	Actor	User Intention or System Reaction
1	User	möchte tun
2	System	reagiert

Erweiterungen

Welche zusätzlichen sub – Use Cases können/dürfen/müssen durchlaufen werden?

3.3.16 Authentisieren

Ziel

Das System kann einen User sicher[1] Identifizieren.

Beschreibung

Zur wahrung der Datensicherheit und der Privatsphäre der Benutzer ist es notwendig die Identität eines Users zu verifizieren.

Preconditions

User braucht funktionsfähige Bookman Infrastuktur (Client, Server, Account, Netzverbindung).

Postconditions

System kann den Komunikationspartner Identifizieren.

Primary Actor

System

Main Success Scenario

#	Actor	User Intention or System Reaction
1	System	möchte sich über die Identität seines Komunikationspartners informieren.
2	System	Vordert den Buntzer auf sich anhand eines Benutzernamens und gemeinsamen Geheimnisses zu zu identifizieren.
3	User	Teilt dem System die gewünschte Information mit.
4	System	überprüft die Angaben

Erweiterungen

Keine?

[1] Den Grad der Sicherheit ist lassen wir hier erst einmal aussen vor.

Kapitel 4

Domain Analyse

4.1 Administratives

4.1.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	27. 11. 2003	cal
0.1.1	Abstract auf Titelblatt hinzugefügt	7. 11. 2003	cal

4.1.2 TODO

was	bis wann	wer
Inhalt hinzufügen	W44, 11. 11. 03, jetzt...	cal, adi, ??

4.2 Bookmarks

In diesem Abschnitt geht es darum die Informationen welche Browser mit Bookmarks abspeichern zu evaulieren sowie die Art wie sie diese tun.

4.2.1 Mozilla (Unix)

Hierarchie

→Mozilla unterstütz eine Baumartige Bookmark Struktur mit geschachtelten Ordnern.

Speicherung und User Verwaltung

Die Bookmarks werden in der Datei `$HOME/.mozilla/default/2sun0ugv.slt/bookmarks.html` abgelegt. Wie der Name des Directoreis, welches unter anderem die Bookmarks enthält, gewählt wird ist nicht klar. Allen untersuchten Installationen ist folgende schreibweise *<8 Zeichen>*`.slt` gemeinsam.

Aufgrund des Unix inhärenten Userkonzeptes und der entsprechenden Speicherung der Daten im jeweiligen Home Verzeichniss ist die Unterscheidung der User klar.

Speicher–Format

Das Format ist HTML. Die Informationen werden in einer Baumartig organisierten Definition–List (<DL>) gespeichert.

Ein Eintrag (bookmark) sieht etwa so aus:

4.2.2 Opera (Unix)

Die folgenden Informationen wurden durch reverse Engineering erarbeitet.

Hierarchie

Opera unterstütz eine Baumartige Bookmark Struktur mit geschachtelten Ordnern.

Speicherung und User Verwaltung

Die Bookmarks werden in der Daei `$HOME/.opera/opera6.adr` abgelegt. Woher der Bezeichner *6* kommt ist unklar.

Aufgrund des Unix inhärenten Userkonzeptes und der entsprechenden Speicherung der Daten im jeweiligen Home Verzeichniss ist auch bei Opera die Unterscheidung der User klar.

Speicher–Format

Die Daten werden in einer sehr klar und einfach aufgebauten Textdatei gespeichert.

Ein **Bookmark** Eintrag mit allen Features sieht wiefolg aus:

```
#URL
   NAME=Opera's language files
   URL=http://www.opera.com/download/languagefiles/
   CREATED=1069983090
   VISITED=1070011251
   DESCRIPTION=Ich bin die Descrption
   SHORT NAME=nick
   ACTIVE=YES
   ON PERSONALBAR=YES
   PERSONALBAR_POS=-1
   IN PANEL=YES
   PANEL_POS=8
```

Ein gültiger Minimaleintrag besteht aus den Feldern NAME und URL.

Feld	Beschreibung
#URL	Scheint der Identifier für den Eintragstyp "URL" zu sein.
NAME	Titel des Bookmarks
URL	URL des Bookmark Ziels.
CREATED	Unix Timestamp der Erzeugung.
VISITED	Unix Timestamp des letzten Visits.
DESCRIPTION	Beschreibung des Eintrages in Worten. Encoding und maximale länge unbekannt.
SHORT NAME	Nikmane des Bookmarks.
ACTIVE	Dieser Eintrag ist vorhanden und auf den Wert `YES` gesetzt wenn im das Bookmark im Bookmarkview ebefalls aktiv ist (blau hinterlegt).
ON PERSONALBAR	Dieses Feld ist vorhanden und auf den Wert `YES` gesetzt, wenn der User das Bookmark auf sein Persaonal bar anzeigen lässt.
PERSONALBAR_POS	Der Zweck ist unklar. Der Wert scheint aber immer auf `-1` gesetzt zu sein.
IN PANEL	Diesess Feld ist vorhanden und auf den Wert `YES` gesetzt, wenn der User das Bookmark in seinem Hotlist pannel anzeigen lässt.
PANEL_POS	Dises Feld enthält die Position des Eintrages im Hotlist pannel (sollte daher eindeutig sein).

Ordner werden wiefolgt definiert:

```
#FOLDER
   NAME=subsubfolder
   CREATED=1069982931
   ACTIVE=YES
   EXPANDED=YES
```

Other Folders or Bookmarks

-

Der Inhalt eines Ordenrs ist also zwischen seiner Deklaration und seines End–Striches zu finden. Das können sowohl Bookmarks wie auch weiter Ordner sein (Schatelungstiefe unbekannt). Ein gültiger Ordner braucht nur das Feld `NAME`.

Feld	Beschreibung
#FOLDER	Scheint der Identifier für den Eintragstyp "Ordner" zu sein.
NAME	Titel des Ordners
CREATED	Unix Timestamp der Erzeugung.
ACTIVE	Dieser Eintrag ist vorhanden und auf den Wert `YES` gesetzt wenn im das Bookmark im Bookmarkview ebefalls aktiv ist (blau hinterlegt).
EXPANDED	Dieser Eintrag ist vorhanden und auf den Wert `YES` gesetzt wenn der Ordner aufgeklappt ist.

Kapitel 5

Richtlinien

5.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	???	adi
0.1.1	Coding Style Guide überarbeitet	12. 11. 2003	cal
0.2	Kappitel hinzugefügt	12. 11. 2003	cal
0.2.1	Dokumente und Versioning hinzugefügt	19. 11. 2003	cal
0.2.2	Coding Style Guide um LaTeX erweitert und umstrukturiert	20. 11. 2003	cal

5.1.1 TODO

was	bis wann	wer
Restliche Regelen	W47	cal

5.2 Dateisystem

5.2.1 Dateinamen und Pfade

Unter UNIX ist das Dateisystem Case Sentsitive. Daher müssen wir auch unter Windows die Konventionen beachten.

Der erste Buchstabe wird generell klein geschrieben. Zusammengestzte Wörter beginnen mit einem kleinen Buchstaben. Jedes angehängte Wort beginnt mit einem grossen Buchstaben[1] (zB dasIstEinBeispiel). Wir verwenden keine Umlaute und Leerschläge im Dateisystem ('ä'-¿ 'ae' etc, m̈y House.bsp-̈¿ m̈yHouse.bsp̈). Dateinamen bei Codefiles auf Englisch, dadurch Konsitent mit Code und gar keine Umlaute nötig.
Beispiele:

- aStupidName
- UFO (reine Abkürzung)
- aUFOPilot (erster Bstb. klein)

5.2.2 Dateinamenserweiterungen

Nachfolgend definieren wir die wichtigsten Dateinamenserweiterungen. Falls nicht anders erwähnt kann vom etablierten Standard ausgegangen werden.

Datei Typ	Erweiterung
C Source	c
C++ Source	cpp
C++ Header	h
C# Source	cs
Java Source	java
Java Script	js
ASCII Text	txt
LaTeXcode	tex
HTML code	html
XML code	xml
Tape Archive (GNU)	tar
GNU–Zip	gz
GNU–Zip Tape Archive	tar.gz

(zu erweitern...)

[1] CamelCase

5.3 Datei Hierarchien

5.3.1 CVS

Zur Verwaltung von Code steht auf Sourceforge für unser Projekt ein CVS-Repository zur Verfügung.

Ins CVS gehören nur Inhalte die selber geschreiebn worden sind; insbesondere gehören Dateien, welche generiert werden *nicht* ins CVS. Beispiele:

gehört ins CVS	gehört nicht ins CVS
TEX-Datein	dvi, toc, aux, glo etc.
Makefile	Projekt-Metafiles
Code	generierter Code

Module

Das Repository kann in verschiedene, unabhängige Module gegliedert werden.

doc Im Modul *doc* wird die Dokumenation gehalten.

server Das Modul *server* enthält den Source Code für den Server Teil der Arbeit.

client Im Modul *client* befindet sich der gemeinsame Teil (falls es einen solchen geben wird) der Cleint seite.

moz-plug Enthält den Mozilla-spezifischen Teil des Cleint Plugin's.

ie-plug Enthält den Internet Explorer-spezifischen Teil des Cleint Plugin's.

opera-plug Enthält den Opera-spezifischen Teil des Cleint Plugin's.

5.3.2 Dokumentation

Die Dokumentation wird im Modul *doc* des CVS gehalten. für jedes einzelne Dokument existiert ein Unterverzeichniss, sinnvollerweise (aber nicht zwingednd) mit der Abkürzung des entsprechenden Dokument benannt.

Dateien

Datei	Beschreibung
/`Makefile`	Enthält und definiert die Regeln welche zum Übersetzen von /`main.tex` vonnöten sind..
/`article-head.tex`	Diese Dokument deklariert für alle articles in den Unterordnern gemeinsam den Header.
/`main.tex`	Die Header Infomration, welche benötigt wird um alle einzel-Dokumnete zu einem Buch zusammen zu fassen und Dinge wie ein globales Inhaltsverzeichniss und Dokument-übergreifende Referenzen zu ermöglichen.
/`style.tex`	Hier definieren wir Global geltende Styles bzw. Kommandos zur Formatierung der Dokumente (Artikel und Buch).
/*document*/	Verzeichniss für ein beliebiges im PMP erwähntes Dokument. es Enthällt *alle* Informationen und Inhalte, welche zu diesem Dokument gehören.
/*document*/`Makefile`	Enthält und definiert die Regeln welche zum Übersetzen des einzelnen Dokumentes nötig sind.
/*document*/`main.tex`	Für jedes Dokument existiert ein `main.tex`, welches den Inhalt in einen LaTeX *article* fasst.
/*document*/`inc`*n*`.tex`	Der eigentliche Inhalt steht in Dateien welche nach dem Schema `inc`*n*`.tex` benannt sind. Es existiert mindestens ein `inc0.tex`, für weitere Teile wird n jeweils um eins erhöht.
/*document*/`images/`	Beinhaltet die Bild ressourcen zum Dokument. Achtung: Binäre Dateinen müssen unbedingt mit der Option `-kb` ins CVS eingecheckt werden.

Im Verzeichniss `/template/` existiert die jeweils aktuellste Version der Dokumenten Verlage für Einzeldokumente. das Template umfasst `Makefile`, `main.tex` und `inc0.tex`, die alle in den Ordner des neuen Dokumentes kopiert werden.

5.3.3 Source Code

Zum Source Code zählen wir in erster Linie Menschengeschriebener Code (also nicht automatisch generiert).

5.4 Coding Style Guide

5.4.1 Allgemeines

Tabulator

Standard Tabulatorweite ist 3 Spaces. Tabulatoren werden *nicht* durch Spaces ersetzt. Dadurch kann es jeder Programmierer so Einstellen wie er es will.

Blöcke

Informationen und logisch zusammengehöriges wird in Blöcke gegliedert. Solche Blöcke werden gemäss der Logischen Struktur im einen Tabulator eingerückt (zB. Dokument-Hierarchie oder Code Blöcke).

5.4.2 Programmcode

.

Sprache

Der Code ist Englisch. gibResultat() -¿ getResult() Ebenso Dateinamen (Konsistenz zum Code). Kommentare zum Code sowie Texte in Sourcefiles sind ebenfalls in Englisch zu halten.

Kommentare

Sollen die Verständlichkeit für den Programmierer fördern. Dazu gehört Erläuterung der Funktion (was tut das) und Beschreibung des zugrundeliegenden Gedankens (warum so und nicht anders).

TODOs im Code Stellen an denen der Code noch unvollständig ist oder verbessert werden muss etc. werden mit einem entsprechenden TODO-Kommentar gekennzeichnet. Damit muss später der Code nur nach diesen durchsucht werden um keine Stellen zu vergessen.

```
/* find i before the point x */
for(i= 0;(i < upSize) && !(shapeCoords[i+1].x() > x);++i){}
/* we have now i pointing to the index right before x */

/* TODO: do the interpolation */

return(TCoord(-999));
```

Operatoren

Um Zuweisungen und Vergleiche besser unterschieden und schneller erkennen zu können gelten folgende Regelen:

Zuweisungen:
Bei Zuweisungen wird rechts des Operators kein Leerzeichen eingefügt

```
int a= 0;
bla+= 5;
```

Vergleiche:
Beiderseits des Operators ein Leerzeichen

```
if(a == b)
while(n < i)
```

Klammern und Indent

Es wird vom 1TBS[2] abgewichen:

```
if(true)
{
   cout << "hello";
   cout << " world!" << endl;
}
else
{
   cout << "bye";
   cout << " world!" << endl;
}
```

Klassen

Nur eine Klasse pro Datei.

Sofern die Programmiersprache dies erlaubt, folgen nach dem Header (siehe **??**) die includes.

Zuerst kommen die vom System zur verfügung gestellten Inlcudes und anschliessen, mit zwei Zeilen abstand, die selbst selbst geschreibenen (zum Projekt gehörenden) Includes.

Bei Sprachen welche eine Trennung von Implementation und Deklaration kennen[3], dürfen Includes nur in den Headerfiles gemacht werden (ausser natürlich den eigenen header).

```
/*
<HEADER>
*/

#ifndef POCKETBOMB_H
#define POCKETBOMB_H
```

[2] "One True Brace Style", siehe: http://catb.org/esr/jargon/html/0/one-TBS.html, http://catb.org/esr/jargon/html/I/indent-style.html

[3] Die richtigen Sprachen, zB. C und C++

```
#include<math.h>  // "system" includes
#include<iostream.h>

#include"bomb.h"  // own includes
#include"object.h"
```

Mit 3 Leerzeilen Verzug kommt nun die Deklaration oder die Definition der Klasse.

Bei Deklarationsfiles wird pro deklariertem Item eine Zeile Abstand eingefügt. Der Deklaration geht eine Zeile Kommentar voraus wlecher beschreibt was die Mothode tut oder wofür das Attribut gut ist.

```
class PocketBomb : public Bomb, public Object {
   public:
      PocketBomb(void* Parent,
                 const int* Size= 10,
                 Coordinate position= Coordinate(0,0,0),
                 Object* dedicatedTo= 0);
      ~PocketBomb();

      /* get a pointer of the current Operator of the device */
      void* getParent(void);

      /* Assigns a new parent */
      void setParent(void* newParent);

      /* returns the actual position of the device. */
      Coordinate getPosition(void);

      /* assignes (moves) it to a new Position. */
      void setPosition(Coordinate newPosition);

      ...etc....

   private:
      /* Keeps track of the Target */
      Object* theTarget;

      /* Keeps track of itself */
      Coordinate bombPosition;

      /* stores the bombsize */
      int bombSize;

      /* points to the parent */
      void* blaParent;
};

#endif
/*
<FOOTER>
*/
```

Bei der Definition/Implementation werden zwischen allen Methoden drei Zeilen Abstand eingefügt.

```
/*
<HEADER>
*/

#include"ingniter.h"  // self



PocketBomb::PocketBomb(void* Parent,
              const int* Size= 10,
              Coordinate position= Coordinate(0,0,0),
              Object* dedicatedTo= 0)
              : blaParent(Parent),
                bombPosition(position),
                theTarget(dedicatedTo)
{
   ...some code...
}  /* PocketBomb::PocketBomb */



PocketBomb::~PocketBomb()
{
   this->explode();  // we dont let the bomb be defused.
}  /* PocketBomb::~PocketBomb */


/*-------------------------------------------------------
/     THE METHODES/CALLBACKS/WHAT EVER...
/-------------------------------------------------------*/


void* PocketBomb::getParent(void)
{
   return(blaParent);
}  /* PocketBomb::getParent */

...etc...

<FOOTER>
```

Header

Enthält Informationen zu:

- Author, Copyright: Name, Mail
  `(c) by John Random Hacker, j.r.hacker@geek.net`
- Contributers/Coauthors
  `defusing protection added by Henry B. Dynamite, hbd@xplode.net`

- Erstellt: Datum dd.Month.YYYY
  `3. Sep. 2003`
- Modifiziert: Datum dd.Month.YYYY
  `4. Sep. 2003`
- Dateiname
  `source.h`
- Projektname und URL
  `Bookman, http://bookman.sourceforge.net/`
- Beschreibung der Aufgabe/Zweck. 1 bis 10 Zeilen
  `The class BookmanPlugin is intended to get ...`
- Staus der Datei oder Codes: 1 bis 10 Zeilen
  `The class BookmanPlugin is intended to get ...`
- Lizenz (ie. GPL)

Beispiel:

```
/* TShape.cpp

This file is part of the sourcecode of the UNIFOIL project. See
http://unifoil.sourceforge.net for details.

Licensed under the Terms of the GNU General Public License V.2
See the file COPYING or http://www.gnu.org/copyleft for info.

Copyright (C) 2001  by Michael Naef, michael.naef@switzerland.org

interpolation contributetd by John Random Hacker, j.r.hacker@geek.net

--

coding begin:  27. October 2003
last modified: 28. October 2003

(Work done in the early morning hours, eg. in a long coding night,
is considered to be done on the day before, the day the coding session
was started. :-)
--
DESCRIPTION

TShape is the abstract baseclass of classes involved in shape
handling. Its purpose is to ensure a minial common inaerface
for all of these classes.
It is derived from TChainLink wich add the automatic chaining
and updating machanism.

STATUS
```

```
After major changes in the design partly implemented.

*/
```

5.4.3 LaTeX

Soweit hier nicht anders definiert, gellten hier die Regeln aus 5.4.1 und 5.4.2.

Sprache

Da das Projetkt, als Semesterarbeit in der Schweiz entsteht, ist die Dokumentationsprache Deutsch. Einzelne Teile können Zweisprachig in Deutsch und English geführt werden um das Prjekt nach Abschluss der Opensource Comunity zu öffnen.

Zeilenlänge

Als Richtwert für die Zeilenlänge gilt 70 Zeichen, danach erfolgt ein manueller umbruch. Zeilenlängen über 80 Zeichen sollten unbedingt vermiesen werden.

Blöcke und Indent

Um das Dokument lesbarer zu machen, definieren wir hier den Indentstyle noch etwas genauer.

Umgebungen

LaTeX hat durch `\begin{...}` und `\end{...}` umschlossene Umgebungen, die sich fast nach den selben Regeln wie bei Programmcode einrücken lassen:

```
\begin{itemize}
    \item{Sackmesser}
    \item{Küchenmesser}
    \item{Buschmesser}
    \item{Höhenmesser\\
         Altimeter, Variometer}
\end{itemize}
```

Struktur

Das ist jeoch nicht bei allen LaTeX Konstruktionen so. So sind etwa die Kapitel und Sektionen nur an Ihrem Anfang Markiert. Gerade deshalb ist es wichtig den Block drurch *Einrücken* sichtbar zu machen:

```
\section{Einführung}

   \subsection{Sackmesser}
      Sackmesser zeichnen sich durch ihre kompakten Masse und
      vielfältigen Funktionen aus.
```

```
\subsection{Küchenmesser}
   Die Küchenmesser sind in ihrem Design schon viel
   spezifischer auf einen bestimmten Verwendungszweck
   abgestimmt.

\section{Buschmesser}
   Diese Sorte der Messer ist in Europa kaum vorhaden,
   weshalb wir hier auch nicht weiter darauf eingehen.

\section{Höhenmesser}
   Dieser Abkömmling bildet eine spezielle Gattung der Messer.
   Er kann in zwei Arten eingeteilt werden.

      \subsubsection{Altimeter}
         Die erste Art misst die Absoluthöhe anhand des eines
         korrigierten Referenzluftdruckes auf Meereshöhe (QNH).

      \subsubsection{Variometer}
         Die zeitliche Druckänderung kann herangezogen werden
         um die Steig- oder Fallgeschwindigkeit zu ermitteln.
```

Tabellen

Tabellen sind mischformen der Beiden oberen Typen. Auch sie könne sinnvoll eingerückt werden:

```
\begin{tabular}{|p{7cm}|l|l|}
      Messer & Verbreitung \\
   \hline
   \hline
      Sackmesser & Weltweit \\
   \hline
      Küchenmesser & Erste Welt \\
   \hline
      Buschmesser & Busch \\
   \hline
      Höhenmesser & Atmosphäre \\
\end{tabular}
```

Hier wird der `\hline` Befehl verwendet um die Zeilen anzudeuten. Bei Tabellen ohne Trennede Linie kann zB. eine Leerzeile eingefügt werden.

5.5 Dokumentation

Hier definieren wir die Typographischen Regeln im Zusammenhang mit dem Dokumenent und LaTeX.

Merkmal	Beschreibung	LaTeX Befehl
Betonung	Wichtige Stellen, bzw. Stellen die mann *hervorheben* möchte	`\emph{}`
Mozilla	Programmnamen (nicht Kommandos)	`\tprog{}`
ftp://...	URLs	`\turl{}`
`Makefile`	Dateinamen und Pfade	`\tfile{}`
`tar -cf ...`	Shellkommandos	`\tshell{}`
`if [ -z ...`	Source Code	`\tcode{}` `\verbatim`
any number	veräderlicher Inhalt, Variable	`\tvar{}` `$`*formula*`$`
→	kennzeichnet eine weiterführende Ressource	`\df`

5.5.1 Glossar

Nachfolgend einige Glossar spezifische Typos.

Merkmal	Beschreibung	LaTeX Befehl
Schimmel	Begriffe, die im Glossar ebenfalls verzeichnet sind	`\see{}`

5.6 CVS

Verantwortlichkeiten, wer darf was, Kommentare etc...

5.7 Versioning

5.7.1 Nummerierung

Versionsnummern sind durch Punkte getrente Zahlen und keine Dezimalbrüche. Nach 0.99 folgt also 0.100, 0.101 etc.

Die Nummerierung hat eine Maximale Tiefe von drei Zahlen, danach kann noch ein Kleinbuchstbe folgen: 0.22.2.a. Die erste Zahl Identifiziert den Major-Release, die Zweite den Minor-Release, die Dritte den Service-Release (name?) und der Buchstabe allfällige kleine "kosmetische Änderungen".

Nicht aufegführte Versionszahlen gelten als 0. 1.0.0 ist also gleichbedeutend mit 1.0 oder 1. Wenn mann auf einen ganzen Bereich hinweisen will geschient das zB. mit 1.1.x oder 1.1–1.3.3

CVS Versionen

Die File Version im CVS hat grundsätzlich nichts zu tum mit der Dokument oder code Version.

5.7.2 von Versions zu Version

Arbeiten mit Versionsnummern < 1 gelten als Entwicklungs-Releases auf dem Weg zur Major-Release 1.

Mit dem erreichen der Version 1.0 gilt die Arbeit als Stabil (Software) oder Einsatzfähig.

Minor-Releases können sehr gut mit dem Erreichen von Meileinsteinen während der Entwicklung zusammenfallen.

Anhang A

Aufgabestellung

Aufgabenstellung Studienarbeit HSR WS 2003/04

Interoperabler Bookmark-Service

Einführung

Die verschiedenen Web-Browser verwalten Bookmarks in jeweils eigendefinierten Formaten. Arbeitet man an verschiedenen Orten mit verschiedenen Browser (bei Kunden, Tochtergesellschaften, ab PDA, etc.) wird die Verwaltung von persönlichen Bookmarks sehr aufwändig.

Ideal wäre die Verwaltung von Bookmarks in einem browserunabhängigen Format innerhalb eines via Internet erreichbaren Services.

Als Benutzer könnte man ab einem Browser die Bookmarks von diesem Service herunterladen und mit diesen Arbeiten.
Zudem könnten die Bookmarks selber lokal bearbeitet werden und dann wieder auf den Service heraufgeladen werden, damit die Änderungen in den Bookmarks für spätere Sessions mit anderen Browser auch zur Verfügung stehen.

Der Bookmark-Service sollte zudem pro Benutzer mehrere Sets von Bookmarks unterstützen und Merge-Mechanismen zur Verfügung stellen, mit denen aus verschiedenen Sets ein neuer Set erzeugt werden kann.

Aufgabenstellung

Erstellung eines interoperablen Bookmark-Services gemäss obiger Beschreibung für die Web-Browser:

- MS Internet Explorer
- Netscape/Mozilla
- Opera

Abbildung A.1:

Aufgabenstellung Studienarbeit HSR WS 2003/04

Infrastruktur

WinNT 2000 (HSR Abt. I Standard-Konfiguration)

Termine

27.10.03 Start
06.02.04 1700 Ende und Abgabe der Arbeit

Organisation

Von den Besprechungen ist jeweils ein Protokoll zu erstellen.
Format: Text
Filename: `YYYYMMDD`.txt
Verteilung: per Mail als Attachment
Termin: 3 Tage nach Besprechung

Es muss pro Resultat (*Dokument*, *Source-File*, etc.) die Verantwortlichkeit ersichtlich sein.

Es ist pro Projekt-Mitarbeiter eine Zeiterfassung über die für die Arbeit aufgewendeten Stunden zu führen.

Betreuung

Thomas Letsch
tletsch@hsr.ch
055 - 22 24 567 (HSR 5.204); 055 - 214 43 50 (Geschäft)

Rapperswil, 27.Oktober 2003

Thomas Letsch

Abbildung A.2:

Anhang B

Glossar

B.1 Administratives

B.1.1 Versionsliste

Version	Beschreibung	Datum	Autor
0.1	Dokument erstellt	29. 10. 2025	cal
0.1.1	Header angepasst (gemass SG)	19. 11. 2003	cal
0.1.2	Zusätzliche Einträge	19. 11. 2003	cal
0.1.3	Viele Zusätzlich Einträge	3 . 12. 2003	cal

Anmerkung: Die Version wird nicht nach jedem Eintrag erhöht, sondern nach gutdünken von Zeit zu Zeit wenn sich genügend relevante Änderungen angesammelt haben.

B.1.2 TODO

was	bis wann	wer
Inhalt hinzufügen	immer	cal, adi

B.2 Glossar

ASAP Jargon: *As Soon As Possible*

CVS–Module, Module Modules beinhalten von einanander unabhängige Daten auf dem Repository.

CVS–Repository, Repository Die dem Projekt zur Verfügung stehende Datenablage des CVS Systems.

CVS *Concurrent Versions System*. Dienst zum Verwalten und verteilten Bearbeiten von Source Code[1]. Eine gute Erklärung der Terminologie und des Services allgemein ist auf Sourceforge erhältlich[2].

Definition (–sfile) siehe →*Implementation*.

Deklaration (–sfile) Bei Programmiersprachen welche eine Trennung des Codes *Deklaration* und *Implementation* zulassen, bezeichnet dies die formale Angabe der Schnittstellen ohne weitere Angaben zu Implementation (*Headerfile* in C/C++).

Dienst Bezeichnet die Gesammtheit des Services, der dem Anwender erbracht wird.

Freie Software Freie Software ist eine Untergruppe der *OSS*[3].Der Term wurde und wird vorallem von der FSF[4] und Richard Stallman geprägt. Sie erweitert den Katalog der Anfoderungen um einen wichtigen Punkt: Freie Software *muss* frei bleiben. Der Wichtigste Vertreter ist die *GPL*.

FS siehe →*Freie Software*

GPL *GNU General Public License*. *OSS* Lizenz für *freie Software*. Ursprünglich für das GNU–Projekt[5] formuliert, heute eine der bedeutendsten *OSS*/*FS* Lizenzen.

Implementation (–sfile) Bei Programmiersprachen welche eine Trennung des Codes *Deklaration* und *Implementation* zulassen, bezeichnet dies die die konkrete Implementation der Funktionalität.

Katergorie (Bookmark–) Mit einer *Kategorie* wird die Zuordung eines Bookmarks zu *einem* bestimmten Thema oder Gebiet beschrieben.

Major–Release Bezeichnung für die Hauptversion des Programmes. Innerhalb verschiedener Versionen des selben *Major–Release* können zwar unterschiedlich viele Features oder anderweitige Unterschiede auftrerten, die Programme selbst entstammen aber enkennbar demselben Major–Release. Das Globale Konzept ist allen Versionen eines *Major–Releases* gemein.

[1] *http://www.cvshome.org/*
[2] *http://sourceforge.net/docman/display_doc.php?docid=14033&group_id=1#basicterminology*
[3] Es gibt einige Leute, die würden mich für diese Aussage wohl steinigen.
[4] Free Software Foundation, *http://www.fsf.org*
[5] GNU's Not Unix, *http://www.gnu.org*

Master–Set Das *Master–Set* ist die Bookmarksammlung auf dem Server. Ein Client kann davon eine *Working–Copy* herunterladen oder sein *Working–Set* damit synchronisieren.

Minor–Release Bezeichnung für die Sub–Version eines *Major–Releases*. Minor Releases unterscheiden sich durch markannte neue Funktionen, oder wenn sich das Verhalten eines Programmes markannt verändert.

NIL *No Items Listed*. Zu Deutsch: Kein Eintrag vorhanden. Steht als Kommentar in (temporär oder absichtlich) leer gelassenen Tabellen u.ä.

Open Source Software Software die einer Speziellen von der OSI[6] zugelassen Lizenz unterliegt. Eine OSS Lizenz muss im wesentlichen

- die Weitergabe des Programms und des Quellcodes erlauben,
- das Recht Änderungen anbringen zu dürfen
- und das Recht die Änderungen weiter zu verteilen garantieren,

um der OSD[7] zu genügen. Bekannte Vertreter sind die *GPL*[8], oder die BSD–License[9]

OSS siehe →*Open Source Software*

Service–Release Als *Service–Release* kann eine Sub–Version eines *Minor–Release* bezeichnet werden. Services–Releases desselben Minor–Release unterscheiden sich nicht durch *markannte* neue Funktionalität (→*Minor–Release*) sondern durch *kleine* Verbesserungen oder Fehlerkorrekturen.

Set (Bookmark–) So wird die Gesammtheit aller Bookmarks eines Users bezeichnet

SF siehe →*Sourceforge*

Sourceforge Portal und Dienstleister für *Open Source Software*. *SF* Stellt Infrastruktur zur Etnwicklung von *OSS* Projekten zur Verfügung. Unter anderem ist unsere Prjektwebseite[10] wie auch das CVS–Repository bei Sourceforge. →*http://sourceforge.net/*

Subset (Bookmark–) So bezeichnen wir eine Auswahl aus den *Set*.

SW *Software*. Hardware runs the world, software controls the hardware, code generates the software, have you coded today? — Neil Franklin

synchronisieren Vorgang bei dem die Daten auf dem Bookman Server und einem Client einander nach bestimmten Regeln angeglichen werden.

UC *Use Case*. Zu Deutsch *Anwendungsfall*.

UML *Unified Modelling Language*[11]. Sprache zur beschreibung von Programmen, abläufen etc bei Software–Entwurf.

Working–Copy Arbeitskopie. Im zusammenhang mit dem Bookman Dienst meist ein *Working–Set*.

[6] Open Source Initiative, *http://www.opensource.org*
[7] Open Source Definition, *http://www.opensource.org/docs/definition.html*
[8] *http://www.gnu.org/licenses/gpl.txt*
[9] *http://www.opensource.org/licenses/bsd-license.php*
[10] *http://kookman.sf.net/*
[11] http://www.uml.org/

Working–Set Das *Working–Set* ist die Arbeitskopie der Bookmarks auf dem Client. Sie kann mit dem *Master–Set* auf dem Server synchronisiert werden

Anhang C

Protokolle

C.1 Kickoff Meeting, 27.10.2003

```
Studienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)


Kickoff Meeting, 27.10.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 12:00 bis 12:45 Uhr

Ablauf
- Offizielle Aufgabenstellung der Studienarbeit besprochen
- Allgemeiner Projektablauf besprochen
  -> Inhalt des zu erstellenden Projektplanes besprochen

Entscheidungen
- Projekt auf SourceForge.net publizieren
  -> Projekthomepage auf SourceForge
  -> CVS von SourceForge verwenden
- Sitzungen jeweils wöchentlich
  -> Sitzungsprotokoll innert drei Tage als Attachement an T.Letsch mailen

ToDo
- erste Version des Projekt-Plan bis Freitag Mittag an T. Letsch mailen
- Recherche: -> Einarbeiten PlugIn in Browser einbetten

Nächstes Meeting
- Montag, 03. November 2003, 12:00 Uhr,
```

```
Author:
- Adrian RölliStudienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)
```

C.2 Meeting, 03.11.2003

```
Meeting, 03.11.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 12:00 bis 12:40 Uhr

Ablauf
- allgemeine Fragen zum Projekt besprochen
- erste Version des Projekt Plan überflogen
  (ThL hat das Mail vom Freitag mit dem PP nicht erhalten)
- Ergänzungen des Projekt Plan bestimmt

Entscheidungen
- definitiever Projektname ist "Bookman"
- zukünftig alle E-Mail Attachements zippen (ausser .pdf und
  Sitzungsprotokolle)
- E-Mails für cal und adi an die Mailingliste:
  -> bookman-devel@lists.sourceforge.net
- Inhalt Projekt Plan ergänzt um:
  -> Summary dessen Inhaltes (Document Summary)
  -> auflisten welches Dokument was beinhaltet (Doku Liste)
  -> Resultate der Phasen auflisten
  -> Management Summary über die ganze Arbeit
- in jedem Dokument steht dessen Zweck (von Projekt Plan Doku
  Liste übernehmen)
- 1 Phase Output:
  -> Projekt Plan v1.0
  -> Anforderungsspezifikation v1.0
- 2 Phase Output:
  -> Projekt Plan v1.1
  -> System Architektur (die wichtigsten technologischen Festlegungen)

ToDo
- überarbeiteter Projekt-Plan bis Freitag Mittag an T. Letsch mailen
- Anforderungs Specifikation anfangen
- Zeitplan (soweit es sich bereits planen lässt; dh. im Wesentlichen
  die ersten zwei Phasen)

Nächstes Meeting
- Montag, 10. November 2003, 12:00 Uhr,


Author:
- Adrian Rölli
Studienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)
```

C.3 Meeting, 10.11.2003

```
Meeting, 10.11.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 12:00 bis 12:30 Uhr

Ablauf
- auf Mängel im PMP hingewiesen
- Vorhgehensweise bei der Start-Phase eines Projektes besprochen
- Soll-Inhalt eines Projektplans besprochen (-> Review des alten PMP)

ToDo
- neu erstellter Projekt-Plan bis Dienstang Morgen früh an ThL mailen.

Nächstes Meeting
- Dienstag, 18. November 2003, 17:00 Uhr,


Author:
- Adrian Rölli
Studienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)
```

C.4 Meeting, 18.11.2003

```
Meeting, 18.11.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 17:00 bis 17:30 Uhr

Ablauf
- aktueller PMP genügt als Basis für die folgenden Arbeiten
  (wird vorerst nicht wesentlich geändert)
- weiteres Vorgehen besprochen
  -> AnfSpez erstellen

ToDo
- erste Version der AnfSpez bis Freitag Mittag an ThL mailen.

Nächstes Meeting
- Montag, 24. November 2003, 12:00 Uhr,


Author:
- Adrian Rölli
```

C.5 Meeting, 27.11.2003

```
Studienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)


Meeting, 27.11.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 18:00 bis 19:10 Uhr

Ablauf
- AnfSpez besprochen
  -> bereits zu viele Entscheide in der AnfSpez gefällt
- Mängel beim taktischen Vorgehen aufgedeckt
- das Wort Set's (siehe Aufgabenstellung) erläutert, weil
  bisher nicht vom gleichen gesprochen wurde

ToDo
- Bookmark-Mechanismen der drei Ziel-Browser analysieren
- zukünftig hat die Version eines Dokumentes auf dem Titelblatt
  zu erscheinen
- AnfSpez überarbeiten.

Nächstes Meeting
- Montag, 1. Dezember 2003, 12:00 Uhr,


Author:
- Adrian Rölli
Studienarbeit Bookman (Start 27.10.2003, Ende 06.02.2004)
```

C.6 Meeting, 1.12.2003

```
Meeting, 01.12.2003

Teilnehmer:
- Thomas Letsch, Betreuer HSR
- Michael Naef, Stud HSR
- Adrian Rölli, Stud HSR

Ort/Zeit
- HSR, 11:55 bis 12:50 Uhr

Ablauf
- Erleutert, was gemacht wurde
- nochmals Ziele der Aufgabe definiert.
- Missverständniss bei der Aufgabe
  entdeckt.
-> Wir halten uns an die formulierte
Aufgabestellung.
-> hohlen und ablegen von Bookmark-Sets ohne
     Informationsverlust bei Bookmarks
- UC als Blackbox anschauen
- Input/Output der UC definieren

ToDo
- Protokoll vom 27.11.2003 ergänzen
-> Standard-HSR-Linux-Installation als Testumgebung
     für das Produkt (in AnfSpez aufnehmen)
- Bookmarks der Zielbrowser genauer analysieren
- AnfSpez den neuen Erkentnissen anpassen
-> minimale Version (BlackBox-Variante)

Nächstes Meeting
- Montag, 8. Dezember 2003, 12:00 Uhr,


Author:
- Adrian Rölli
```