Dokumentation
Praktikum Software Entwicklung

Dozent: Prof. Dr. Richard Göbel
Beteiligte Studenten:
Christoph Piechula
Christoph Cwelich
Christopher Pahl
Eduard Schneider
Florian Bauer
Sabrina Biersack

7. Mai 2012

Inhaltsverzeichnis

Teil I.

Spezifikation

1. Übersicht

Über einen konfigurierbaren Crawler können HTML-Inhalte von Webseiten bis zu einer bestimmten Tiefe aus dem Netz in ein Archiv geladen werden. Da dies parallelisiert erfolgen soll, müssen die Daten nach dem Herunterladen von temporären Verzeichnissen in das gemeinsame Archivverzeichnis synchronisiert werden. Der aus der URL extrahierte Pfad der HTML-Dateien wird dabei auf das Archiv abgebildet, wobei jede HTML-Datei in einen eigenen Archivordner verschoben wird.

Beim Crawlvorgang werden zusätzlich Metadaten der HTML-Seiten erstellt. Diese werden in einer Datenbank und als XML-Datei im jeweiligen HTML-Ordner gespeichert. Die Datenbank soll dabei wieder aus den XML-Daten rekonstruierbar sein.

Außerdem sollen Filter eingehängt werden können, die bereits in den TMP-Ordnern ungültige Dateien löschen. Über eine Java-Schnittstelle kann anschließend wieder auf die Daten zugegriffen werden. Hierzu müssen sich Clients über eine vorgegebene Schnittstelle beim Textarchiv anmelden. Die Clients werden anschließend bei Änderungen oder neuen Einträgen benachrichtigt und können sich selbstständig über die Schnittstelle einen beliebigen Stand der Daten herunterladen und an Analysetools weitergeben. Dabei sollen auch neue Dateien den Archivordnern hinzugefügt werden können. Ebenso sollen die oben genannten XML-Daten um neue Nodes erweiterbar sein. Zu Vorführzwecken wird ein Testanalysetool erstellt.

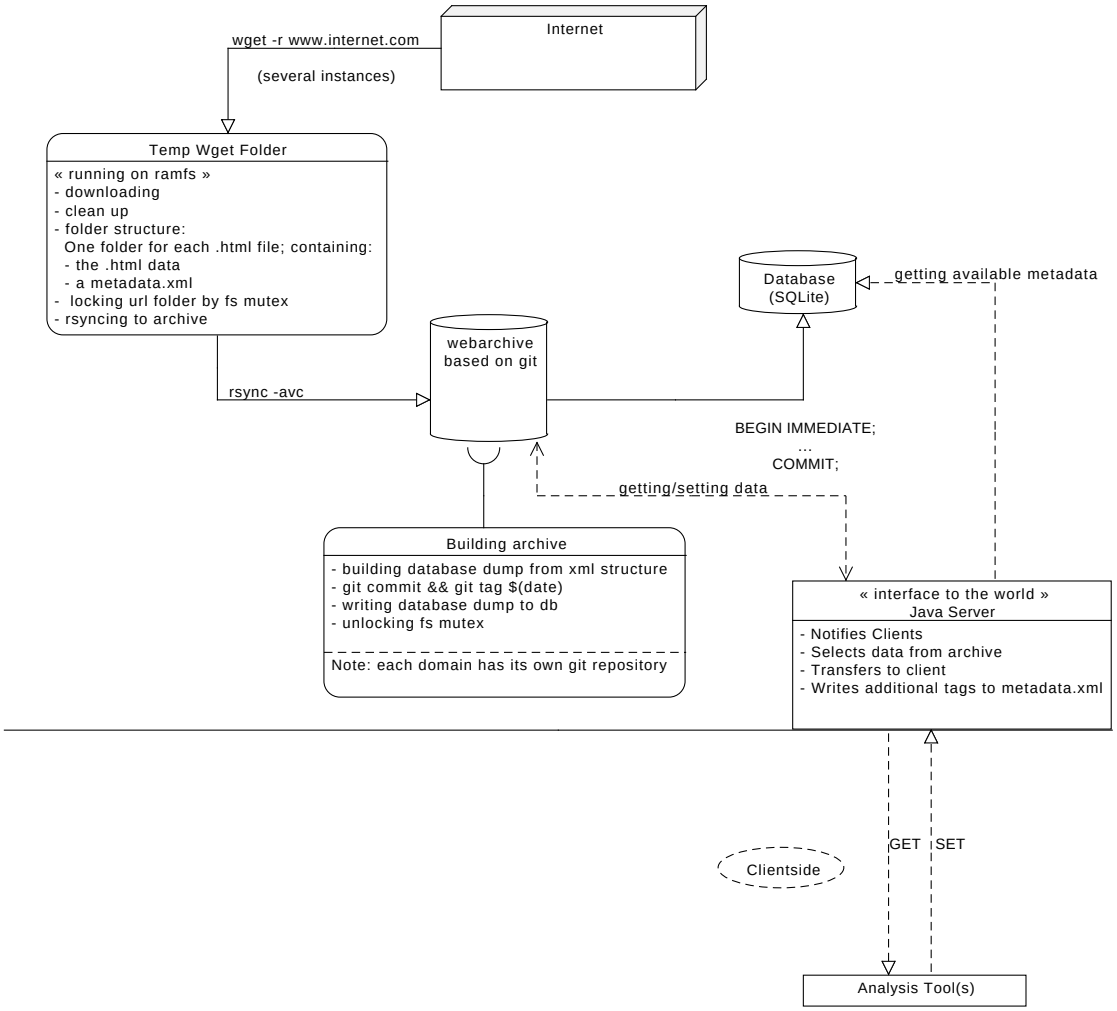

Abbildung 1.1.: Diagramm: Grundlegendes Design

2. Datenmodell der Metadaten

An dieser Stelle wird bereits ein vereinfachtes Datenmodell festgelegt, welches später auf Datenbank und XML-Daten umgesetzt wird und zentrale Metadaten über HTML-Seiten erfassen soll.

Für jede HTML-Datei wird dabei ein solcher Satz von Metadaten angelegt. Die Metadaten sollen vor allen Dingen Such- und Sortieraufgaben erleichtern.

Name	Datentyp	Beschreibung
URL	String	Original URL der HTML-Datei
Titel	String	Soweit vorhanden wird der Titel der Seite gespeichert
Archivpfad	String	Dateipfad des HTML-Archiv-Ordners im Webarchiv
Crawlzeit	Timestamp oder Integer (UTC in ms)	Zeitpunkt des Crawls
Commit Tag	String	Der Committag dient zum Wiederauffinden in der Versionsverwaltung. Der Tag setzt sich aus Domainnamme und dem Zeitpunkt des Crawlvorgangs zusammen.

Tabelle 2.1.: Metadaten

3. Anforderungen

Im folgenden sind die Anforderungen an die Software spezifiziert. Soweit möglich wurden die Anforderungen schon in einzelne Komponenten und Module gegliedert. Eine grobe Übersicht gibt auch Diagramm 1.

Alle Anforderungen werden mit einer Kennnummer in der Form

$$/ < Gruppenprefix > . < Anforderungsnummer > [. < Unternummer >]/$$

gekennzeichnet, damit diese später wieder referenziert werden können.

3.1. Crawlermodul

Dieses Modul soll als eigenständiger Prozess laufen und in regelmäßigen Abständen Crawlvorgänge starten und die Daten in das Archiv schreiben.

3.1.1. Steuerung

/Cr-1/ Config-file Die Steuerung des bzw. der Crawler erfolgt über eine Config-Datei. Ist die Config-Datei nicht vorhanden oder kann sie nicht gelesen werden, so wird auf eine interne Config-Datei zurückgefallen. Es können folgende Parameter eingestellt werden:

- **/Cr-1.1/ Suchtiefe** Suchtiefe bis zu der Links gefolgt werden soll.
- **/Cr-1.2/ Suchintervall** Zeitintervalle der Crawlvorgänge.
- **/Cr-1.3/ Maximale Instanzen** maximale Anzahl der gleichzeitig gestarteten Crawlerinstanzen.
- **/Cr-1.4/ Filtereinstellungen** Liste von Verwendeten Ausschlussfiltern und deren Modulpfad.

/Cr-2/ Kommandozeileninterface Optional können beim Start mittels Kommandozeile zusätzliche Parameter übergeben werden.

- **/Cr-2.1/ Überschreiben** Werte aus dem Config-file werden damit überschrieben.
- **/Cr-2.2/ Domainliste** Es kann eine Liste von Domains übergeben werden, die als Startpunkte für die Crawler verwendet werden sollen. Die Übergabe von mindestens einem Element ist aber immer notwendig.

/Cr-2.3/ DB-Recovery Es kann ein Datenbank-Recovery erzwungen werden. Siehe auch /Db-6/

3.1.2. Ausführung des Crawlvorgangs

/Cr-3/ Parallelität Die Ausführung der Crawlvorgänge soll parallel durchgeführt werden.

/Cr-4/ Instanziierung Pro angegebener URL wird eine Crawlerinstanz gestartet bis die Obergrenze an Instanzen (siehe /Cr-1.3/) erreicht wird. Um Überschneidungen zu vermeiden, werden Domainnamen anderer Instanzen ausgeschlossen. Für die Crawlerinstanzen wird ein externes Tool verwendet (wget).

/Cr-5/ Crawlen Jede gestartete Instanz beginnt nun den Crawlvorgang.

/Cr-5.1/ Herunterladen Jede Instanz kopiert die heruntergeladene HTML-Dateien der Seite in ein temporäres Verzeichnis je Crawlerinstanz.

/Cr-5.2/ Ordnerstruktur Dabei wird die online vorhandene URL-Pfadstruktur der HTML-Dateien auf das Dateisystem abgebildet. Je Domain wird dadurch ein Hauptverzeichnis im TMP-Ordner erzeugt.

/Cr-6/ Filterung Für alle HTML-Dateien wird eine Liste von Filtern durchlaufen. Jeder Filter prüft, ob die HTML-Datei behalten oder verworfen werden soll. Verworfene Dateien werden sogleich gelöscht.

/Cr-7/ Bereinigung In diesem Teil werden die heruntergeladenen Ordner bereinigt, also leere Ordner oder Nicht-HTML-Dateien gelöscht.

/Cr-8/ Normalisierung Es wird ein Archivordner (im Folgenden HTML-Archiv genannt) mit dem Namen des HTML-files (inklusive Dateiendung) erzeugt, das HTML-file in "data.html" umbenannt und in das soeben erzeugte HTML-Archiv verschoben.

/Cr-9/ Extraktion der Metadaten In diesem Vorgang werden die im Datenmodell (2) definierten Metadaten aus dem HTML extrahiert und zwischengespeichert. Es wird auch der CommitTag erzeugt, wobei der Zeitpunkt für alle Dateien pro Domain und Crawler-Instanz gleich ist.

/Cr-10/ Erzeugung von XML-Dateien Die zwischengespeicherten Metadaten werden in XML-Metadateien (siehe auch 3.6) geschrieben und jeweils im zugehörigen HTML-Archiv gespeichert.

/Cr-11/ Synchronisation Die vorbereiteten Daten in den TMP-Ordnern werden nun in das vorhandene Archiv synchronisiert (mit rsync). Veraltete Domain-Ordner werden dabei

komplett überschrieben, können aber gegebenfalls mithilfe der Versionierung wiederhergestellt werden. Dabei sollen auch veraltete Analysedaten gelöscht werden. (Siehe auch 3.3)

/Cr-12/ Datenbankaktualisierung Die Datenbank wird nun mithilfe der zwischengespeicherten Metadaten aktualisiert.

3.2. Filtermodule

/Fi-1/ Schnittstelle Alle Filtermodule sollen eine vorgegebene Schnittstelle erfüllen.

/Fi-2/ Konfiguration Alle Module müssen wie unter /Cr-1.4/ beschrieben dem Crawlmodul bekanntgemacht werden. Eine Zentrale Speicherung der einzelnen Module in einem Filterverzeichnis ist anzustreben.

/Fi-3/ HTML Überprüfung Ein Filter soll ein gegebenes HTML-File überprüfen und einen Wahrheitswert zurückgeben, ob dieses behalten oder verworfen werden soll. Die Prüfmethode ist dabei vom einzelnen Zweck des Filters abhängig.

/Fi-4/ Implementierung Testfilter: Werbefilter Als Testfilter wird ein Filter implementiert, der mittels einer Blacklist HTML-Dateien bestimmter Werbedomains aussortiert.

3.3. Dateiarchiv

/Ar-1/ Struktur Die Dateistruktur wird bereits vom Crawlvorgang vorgegeben, wird hier aber nochmal erläutert:

- Auf der obersten Ebene stehen die Domain-Ordner.
- Darunter wird die von der URL abgebildeten Struktur innerhalb der Domain nachgebildet.
- Die einzelnen HTML-Dateien werden durch HTML-Archiv-Ordner ersetzt, bzw. in diese verschoben.
- Jedes HTML-Archiventhält die Quell-HTML-Datei, die aber in data.html unbenannt wurde sowie die dazugehörige XML-Datei (data.xml). Ein HTML-Archiv kann aber auch weitere Dateien enthalten, welche nach dem Crawlen hinzugefügt werden können.

/Ar-2/ Berechtigungen Generell dürfen keine Änderungen an bestehenden Dateien durchgeführt werden (Ausname: siehe /Xm-3/).

/Ar-2.1/ Externe Benutzer Von exterenen Nutzern (Java-Clients) dürfen nur Dateien in *aktuelle* HTML-Archiv-Ordner hinzugefügt werden.

/Ar-2.2/ Crawler Crawler dürfen neue HTML-Archiv-Ordner hinzufügen oder alte löschen bzw. ersetzen, wobei immer das gesamte HTML-Archiv ersetzt wird.

/Ar-3/ Synchronisation Um gleichzeitige Zugriffe auf Dateien zu verhindern, muss mit Locks oder Mutexen gearbeitet werden. Dabei wird immer der gesamte Domain-Ordner gesperrt. Es muss vor jedem Lese- oder Schreibvorgang ein Lock gesetzt und beim Beenden des Vorgangs wieder entfernt werden.

/Ar-4/ Versionierung Jeder Domain-Ordner wird über eine dezentrale Versionsverwaltung (git) versioniert. Damit ist auch das Wiederherstellen älterer Versionen möglich.

/Ar-4.1/ Hinzufügen Beim Hinzufügen von neuen Dateien müssen diese der Versionsverwaltung bekanntgemacht werden (git add).

/Ar-4.2/ Commit Änderungen müssen stets mit einem Commit bestätigt werden. Dabei wird zur Identifikation immer ein bestimmter CommitTag verwendet. Dabei ist zu beachten:

- wurden Dateien während des Crawlvorgangs hinzugefügt, wird ein neuer CommitTag erzeugt.
- beim nachträglichen Hinzufügen von Dateien wird der alte CommitTag wiederverwendet.

/Ar-4.3/ Ändern und Löschen Beim Überschreiben und Löschen von Dateien müssen keine besonderen Vorkehrungen getroffen werden. Diese werden von der Versionierung in tiefere Versionsebenen verschoben.

/Ar-4.4/ Schreiben in veraltete Verzeichnisse Sollte in Ausnahmefällen in veraltete Verzeichnisse geschrieben werden, zum Beispiel weil die Daten von den Crawlern während einer Analyse geändert wurden, dann werden die Daten verworfen und der Schreiber muss benachrichtigt werden. (z.B. mittels einer Exception)

/Ar-5/ Dateisystem Beim darunterliegenden Dateisystem wird von einem vorhandenen Unix-Filesystem ausgegangen.

/Ar-6/ Komprimierung Eine explizite Dateikomprimierung wird nicht vorgesehen, ist aber zum Teil schon durch die Versionierung gegeben, da alte Revisionen gepackt abgelegt werden.

3.4. Programmierschnittstelle - Java-Client

Diese Schnittstelle soll die Anbindung der Analysemethoden ermöglichen und macht gleichzeitig einen Zugriff über das Netzwerk möglich.

/Cl-1/ Konfiguration Es sind die IP und der Port des Servers in einer Config-Datei zu hinterlegen.

/CI-2/ Client-API Für Benutzer des Clients wird eine Programmierschnittstelle in Java zur Verfügung gestellt. Die API umfasst dabei folgende Schnittstellen:

/CI-2.1/ MetaData Grundlegende Methoden einer Metadatenklasse. Neben den Metadateninformationen soll diese Klasse auch als Schlüsselelement zum Zugriff auf die Archivordner und XML-Dateien dienen.

/CI-2.2/ WebarchiveClient Zentrale Schnittstelle zum Zugriff auf das Webarchiv, Details siehe unten.

/CI-2.3/ Observer Implementierungen dieses Observers können sich beim Client anmelden, um über Änderungen informiert zu werden. Die Schnittstelle enthält eine Methode um Update-Informationen zu erhalten.

/CI-3/ Registrierung am Server Alle aktiven Java-Clients werden beim Server gespeichert, um diese über Änderungen informieren zu können. Beim Abmelden oder Beenden muss ein Client aus dieser Registrierung gelöscht werden.

/CI-4/ Observerregistrierung Benutzer des Clients können sich mittels o.g. Schnittstelle beim Client als Observer an- und abmelden.

/CI-5/ Benachrichtigungen Vom Server kommen in regelmäßigen Abständen Nachrichten über Änderungen. Diese werden an angemeldete Observer weitergegeben. Die Informationen bestehen dabei aus einer Liste von neuen CommitTags. Benutzer des Client können dann entscheiden, welche Daten Sie abrufen wollen.

/CI-6/ Datenbankabfragen Es sollen auch vorbereitete SQL-Statements an den Server geschickt werden können. Die SQL-Abfrage wird soweit vorbereitet, dass nur noch ein SQL-Bedingungsausdruck für die WHERE-Klausel angegeben werden muss. Optional soll auch eine ORDER-BY-Klausel im selben Stil angegeben werden können. Als return-Wert wird eine Liste von Metadatenobjekten zurückgegeben.

/CI-7/ Datei Listing Mittels der Metadatenobjekte kann man sich über eine gesonderte Anfrage eine Auflistung über den Inhalt eines HTML-Archiv-Ordners zurückgeben lassen.

/CI-8/ Datei Lesen Mittels Metadatenobjekt und Dateipfadangabe wird ein Dateistream zum Lesen zurückgegeben.

/CI-9/ Datei Schreiben Mittels Metadatenobjekt und Dateipfadangabe wird ein Dateistream zum Schreiben zurückgegeben. Wie in /Ar-2/ beschrieben, dürfen dabei keine Dateien überschrieben werden.

/CI-10/ Auslesen von zusätzlichen Tags Durch Übergabe eines Metadata-objekts und eines Tagnamens an eine get-Methode wird ein passender XML-Node herausgesucht.

/Cl-11/ Erweiterung der XML-Daten Mittels einer set-Methode, die Namen und Inhalt des Tags als Parameter erhält, können neue Tags zur XML-Datei hinzugefügt werden.

/Cl-12/ Test Analysetool Zur Demonstrations- und Testzwecken der Java-Clientschnittstelle wird ein Analysetool erstellt, welches die Wörter im HTML zählt und das Ergebnis im Archiv als Datei sowie im XML als zusätzliches Tag speichert.

3.5. Server

/Sv-1/ Konfiguration Es ist der Port des Servers in einer Config-Datei zu hinterlegen. Desweiteren werden darin auch alle Pfade zu externe Libraries, Treibern und Ressourcen hinterlegt.

/Sv-2/ Client-Server Kommunikation Es muss ein Nachrichtensystem zwischen dem Client und dem Server implementiert werden. Über bestimmte Kennungen (z.B. über enums) ist der Inhalt einer Nachricht zu kennzeichnen. Dabei müssen folgende Informationen Ausgetauscht werden können:

/Sv-2.1/ Austausch von Stream-Daten Zwischen C. und S. müssen Daten in Form von Streams verschickt werden können. Diese müssen in geeigneter Form bei der Übertragung gepuffert werden.

/Sv-2.2/ Exceptions Vom Clientanwender verursachte Exceptions werden an diesen weitergeleitet und im Client erneut geworfen.

/Sv-2.3/ Datenbankabfragen Datenbankanfragen werden in Form von SQL vom Client geschickt und diesem in Form von Metadaten-Objekten beantwortet.

/Sv-2.4/ Änderungen im Archiv Diese Nachrichtenform enthält Informationen für Client, welche Änderungen im Archiv betreffen.

/Sv-3/ Clienten registrieren Der Server hält eine Liste von angemeldeten Java-Clienten und verwaltet die Verbindungen der Clienten und verwirft sie bei Verbindungsverlust oder beim Beenden der Clients.

/Sv-4/ Update-Notifier Der Update-Notifier ist ein eigens laufender Thread des Servers, der in einem vorgegebenen Intervall (z.B. stündlich) in der Datenbank prüft ob neue Commit-Tags vorhanden sind. Hierfür speichert er sich den Zeitpunkt der letzten Update-Suche und vergleicht ihn mit dem Datum der Datensätze in der Datenbank. Werden Änderungen gefunden, werden das Datum und der Commit-Tag zwischengepuffert.

/Sv-4.1/ Konfiguration Der Benachrichtigungsintervall muss in einem Configfile gespeichert werden.

/Sv-4.2/ Clienten informieren Sobald die Update-Suche fertig ist, werden die gepufferten Informationen an alle registrierten Clienten geschickt. Siehe auch /Cl-5/

3.6. XML-Dateien

/Xm-1/ Inhalt Die XML-Dateien enthalten in ihrem Wurzelknoten eine Meta- und einen Datenknoten. Der Metaknoten ist für die in 2 beschriebenen Metadaten reserviert. Der Datenknoten ist anfangs leer und kann von Benutzern um weitere Knoten erweitert werden.

/Xm-2/ Validierung Für die Validierung der XML-Daten muss eine XML-Schema ausgearbeitet werden. Eine Validierung ist dann durchzuführen, nachdem eine XML-Datei erweitert worden ist. Dabei auftretende Fehler werden in eine Log-Datei geschrieben.

/Xm-3/ Erweiterbarkeit Wie oben beschrieben ist beim Design auf Erweiterbarkeit zu achten. Der Inhalt des Knotens ist frei wählbar, der Name des Knotens muss aber eindeutig sein und darf mit vorhandenen Knoten nicht kollidieren (ggf. mit namespaces arbeiten). Sollten die XML-Dateien erweitert werden, so muss vom Benutzer auch das Schema erweitert werden.

/Xm-4/ Schreibschutz Bei schreibenden Zugriffen auf die XML-Datei ist darauf zu achten, dass nur neue Knoten hinzugefügt werden dürfen und keine vorhandenen überschrieben oder geändert werden können. Desweiteren muss der Metaknoten geschützt werden. Es dürfen auch keine gleichnamigen Knoten in der obersten Ebene des Datenknotens vorhanden sein.

3.7. Datenbank

/Db-1/ Konfiguration Die Struktur der Datenbank wird als CREATE-TABLE-statement in einer SQL-Datei gespeichert. Falls nötig werden auch alle weiteren vorbereiteten SQL-Dateien im selben Ordner abgelegt. Da SQLITE verwendet wird, kann die Datenbank über einfache Dateioperationen erzeugt werden.

/Db-2/ Inhalt Es werden die Speicherstände aller HTML-Dateien und Versionen im Archiv festgehalten. Die gespeicherten Daten entsprechen den Knoten aus 2.

/Db-3/ Erweiterbarkeit Ein Möglichkeit zum dynamischen Erweitern der Datenbank ist nicht vorgesehen und gewünscht.

/Db-4/ Berechtigungen Schreibrechte werden nur dem Crawlmodul erteilt. Gelesen kann falls notwendig von allen Komponenten werden. Wobei externe Benutzer über die Schnittstellen der Clients SELECT-statements abzusetzen können.

/Db-5/ Aktualisierung Die Datenbank muss immer beim Fertigstellen eines Crawlvorgangs auf den neuesten Stand gebracht werden.

/Db-6/ Wiederherstellung Sollte die Datenbank beschädigt oder geändert werden, dann soll diese wieder aus den im Archiv vorhanden XML-Metadaten rekonstruiert werden können. Dabei müssen auch alte Versionstände wieder mit eingefügt werden.

4. Entwicklungsumgebung

4.1. Programmiersprachen

Es werden die Sprachen Python und Java benutzt.

4.1.1. Python

Python in der Version 2.7 oder 3 wird für die systemnahen Teile verwendet:

- Das gesamte Crawlermodul
- Die Filtermodule
- Der Zugriff auf das Dateisystem (Archiv) und die notwendige Ordnersynchronisation.

Python entält bereits leistungsfähige Libraries für den Zugriff auf das Dateisystem, SQLite und die Versionsverwaltung git.

4.1.2. Java

Die Client-Server Architektur der oben genannten Programmierschnittstelle werden mit Java 1.7 umgesetzt. Hierzu ist Java besonders geeignet und es ist gewährleistet, eine an der Hochschule Hof allgemein verständliche Schnittstelle zu schaffen.

4.2. Dokumentation

Die Dokumentation wird in Latex als ein fortlaufendes Gesamtdokument erstellt, welches je nach Phase um weitere Teile erweitert wird.

4.3. Teamsynchronisation

Dokumente und Quellcode werden über ein gemeinsames Repository auf github.com synchronisiert.

4.4. Sprache

Die Sprache der Dokumentation ist Deutsch wobei natürlich geläufige Fremdwörter enthalten sind. Quellcode, Kommentare und daraus abgeleitete APIs (Javadoc und Sphinx) sind in Englisch zu verfassen.