Gottfried Wilhelm Leibniz Universität Hannover

Fakultät für Elektrotechnik und Informatik

Institut für Praktische Informatik

Fachgebiet Software Engineering

# Entwicklung einer fachspezifischen Literaturdatenbank zum Requirements Engineering

Bachelorarbeit

im Studiengang Informatik

von

**Stefanie Grube**

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr. Nicola Henze

Betreuer: M.Sc. Eric Knauss

11. Januar 2008

# Zusammenfassung

Im Software Engineering werden Forschungsergebnisse oft als mehr oder weniger kurzer Aufsatz oder Artikel veröffentlicht. Entsprechend gibt es große Mengen dieser Texte aus unterschiedlichsten Quellen und es ist für Einzelpersonen relativ schwer, den Überblick zu behalten oder statistische Aussagen zu treffen.

Diese Arbeit beschreibt die Entwicklung einer fachspezifischen Literaturdatenbank zum Requirements Engineering.

Die Datenbank wurde als Webseite konzipiert, die zum Beispiel von Gruppen im Intranet genutzt werden kann, um gemeinsam eine auf den eigenen Bedarf angepasste Literaturdatenbank aufzubauen. Auf diese Weise können anhand eines fachspezifischen Schlagwortkataloges interessante Publikationen schnell gefunden werden.

Darüber hinaus ermöglicht es die Datenbank, die Relevanz der Dokumente zu bewerten und so in Verbindung mit spezifischen Mengenanfragen Rückschlüsse auf den Stand der Forschung zu ziehen.

# Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Stefanie Grube, Hannover, den 11.01.2008

# Danksagung

Hiermit möchte ich mich bei Herrn M.Sc. Eric Knauss für die vorbildliche Betreuung meiner Bachelorarbeit bedanken.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Die Erfassung von Anforderungen ist von zentraler Bedeutung für den Entwicklungsprozess von Software. Eine systematische und strukturierte Vorgehensweise trägt erheblich zur Qualität des entstehenden Systems und damit zu seinem Erfolg bei.

Die Forschung auf dem Gebiet des Requirements Engineering ist deshalb bestrebt, die Aktivitäten, die zu brauchbaren Anforderungen führen, zu formalisieren, um optimale Prozesse zu ermitteln.

Die Ergebnisse dieser Forschungen werden zum Beispiel auf Tagungen und Seminaren ausgetauscht und anschließend in Tagungsbänden oder anderen Fachpublikationen veröffentlicht. Eine dieser Konferenzen, die „IEEE International Requirements Engineering Conference“, fand im Jahr 2007 zum fünfzehnten Mal statt [1].

Durch die wachsende Bedeutung nimmt die Zahl der Publikationen zusätzlich stetig zu und es ist zunehmend schwieriger, den Überblick über die für den eigenen Bereich relevanten Publikationen zu behalten.

Es gibt bereits eine Reihe von Werkzeugen, um Literatur zu katalogisieren, aber die meisten dieser Anwendungen sind nicht dafür geeignet, den Inhalt der Dokumente über einen Schlagwortkatalog hinaus zu erfassen und ermöglichen auch keine statistischen Auswertungen des Datenbestands.

## 1.2 Aufgabenstellung

Ziel dieser Arbeit ist es, Software für eine fachspezifische Literaturdatenbank zu entwickeln. Damit soll es nicht nur möglich sein, Forschungspapiere zu katalogisieren und Literatur zu einem Thema zu finden, sondern auch einen Überblick über aktuelle Themen zu erhalten und Trends zu erkennen. Die Software soll darüber hinaus an andere Fragestellungen anpassbar sein.

Um diese Aufgabe zu erfüllen, ist zunächst zu prüfen, welche Werkzeuge es bereits gibt, was ihre Stärken und Schwächen sind und inwieweit sie zur Problemlösung beitragen können. Außerdem soll erarbeitet werden, wie der Prozess der Erfassung abläuft, welche Rollen wie daran beteiligt sind und welche Anforderungen, an das System, sich daraus ergeben, unter anderem in Bezug auf von Usability Aspekte.

Dann ist zu analysieren, wie das Modell, in das die Daten eingeordnet werden sollen, aussehen soll, welche Datenstruktur dafür geeignet ist und welche Methoden für die Handhabung notwendig sind. Es ist dabei wichtig, dass das Schema flexibel ist, um an zukünftige Entwicklungen angepasst werden zu können.

## 1.3 Gliederung

Nach dem ersten Kapitel, der Einleitung, werden im zweiten Kapitel die Grundlagen der Modellbildung erläutert, soweit sie für Literaturverwaltung und Requirements Engineering relevant sind. Hier wird auf das Referenzmodell zum Requirements Engineering als ein bestehendes Modell hingewiesen und seine Anwendungsmöglichkeiten erläutert. Anschließend werden angrenzende und benachbarte Systeme betrachtet und auf ihre jeweiligen Möglichkeiten, die Aufgabenstellung zu erfüllen, untersucht.

Im dritten Kapitel werden der Vorgang der Katalogisierung erläutert und die teilnehmenden Rollen identifiziert.

Das vierte Kapitel beschäftigt sich mit der Konzeption der Datenbank. Es beginnt mit einer Analyse der besonderen Anforderungen an das Benutzerinterface, dann folgt die Entwicklung der zu implementierenden Datenstrukturen und ihrer Methoden. Anschließend werden Anfragetypen für die Ergebnisdarstellung erarbeitet.

Im fünften Kapitel geht es um den Entwurf und Entwurfsentscheidungen. Es beginnt mit der Wahl der Architektur, dann folgt eine Erläuterung des modularen Aufbaus, eine Beschreibung der Benutzerschnittstelle und schließlich werden die Möglichkeiten zum Im- und Export von Daten angesprochen.

Das sechste Kapitel ist das Fazit. Es fasst die Inhalte der Arbeit noch einmal zusammen und stellt sie übersichtlich dar.

Im siebten Kapitel findet sich ein Ausblick, wie die Anwendung weiter ausgebaut werden könnte.

Die Anhänge schließlich beinhalten ein Glossar, das einzelne wichtige Begriffe noch einmal erläutert, eine Übersicht über die Baumoperationen, sowie Abbildungs-, Tabellen und Literaturverzeichnis.

# 2 Grundlagen

## *2.1 Modelle und Modellbildung*

Um von einem Text auf einem Stück Papier zu einem Datensatz zu gelangen, bedarf es eines Abstraktionsprozesses, bei dem die physische Realität in ein logisches Modell überführt wird. Dabei wird nur ein Teil der Realität erfasst, nämlich diejenigen Aspekte, die für die weitere Verarbeitung interessant sind.

Physisch gesehen ist das Rohmaterial einer Literaturdatenbank eine mehr oder weniger umfangreiche Anzahl Texte, die auf Papier gedruckt oder in einer Datei gespeichert sind sowie eine Liste mit Schlagwörtern.

Die für die Literaturverwaltung interessanten Eigenschaften eines Dokumentes sind beispielsweise sein Titel, der oder die Autoren und das Erscheinungsjahr. Eigenschaften wie Farbe und Qualität des Papiers oder Art der Bindung sind in diesem Zusammenhang nicht interessant und müssen deshalb auch nicht erfasst werden.

Der Inhalt des Dokumentes wird durch Schlagwörter ausgedrückt, wobei es festgelegte Schlagwörter gibt, die für bestimmte Themen stehen.

*Beispiel:* *Einem Dokument, das sich mit Interview-Techniken beschäftigt, werden die Schlagwörter „Elicitation“ und „Techniken“ zugeordnet.*

Obwohl es natürlich möglich wäre, die Liste der Schlagwörter zum Beispiel alphabetisch oder nach Wortlänge zu ordnen, ist eine solche Sortierung wahrscheinlich nicht optimal und zur Problemlösung ungeeignet.

Deshalb muss ein anderer Ansatz verfolgt werden: Die Worte werden in einen semantischen Zusammenhang gebracht, das heißt, sie werden nach ihrer Bedeutung sortiert. Um das zu erreichen, muss jedoch dieser Sinn bekannt sein und auch der Kontext, in dem sich alle Schlagwörter bewegen.

Für den Bereich des Requirements Engineering existiert ein Referenzmodell, das Begriffe und Aktivitäten in einen thematischen Zusammenhang stellt. Es wird im folgenden Abschnitt vorgestellt.

### 2.1.1 Das Referenzmodell

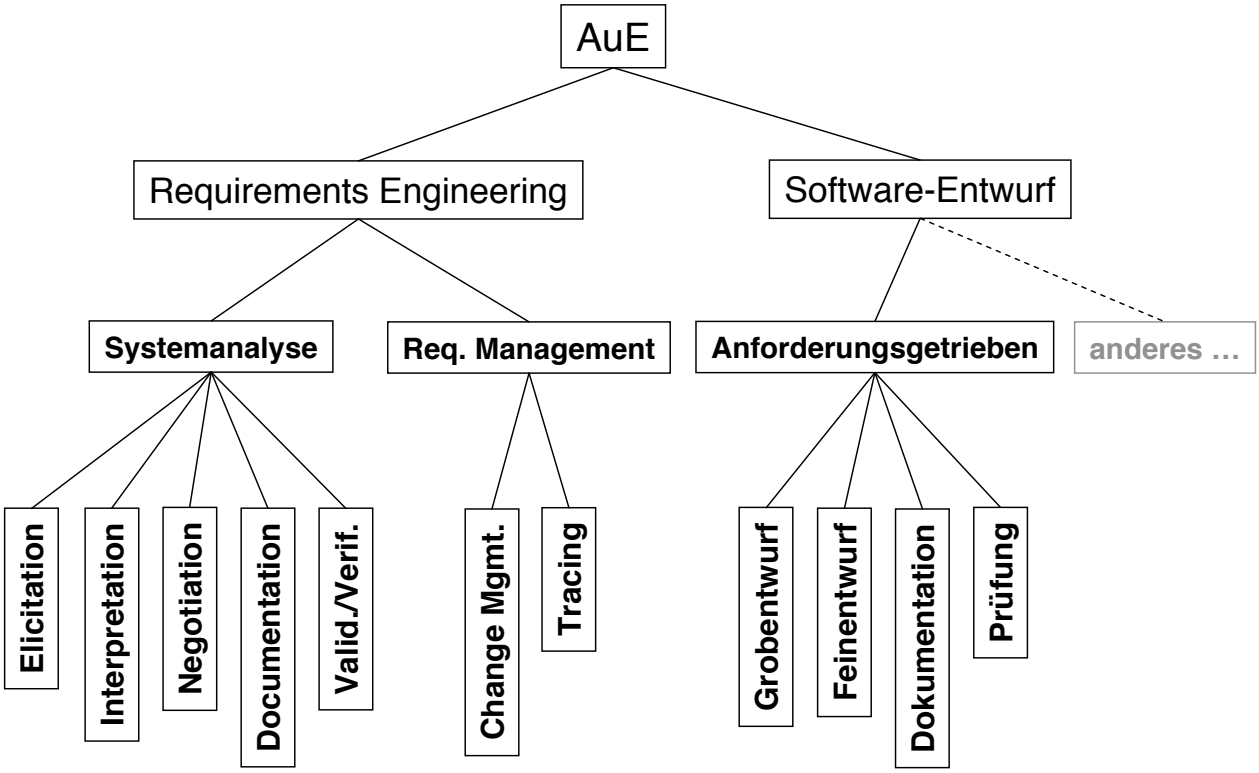


Abbildung 1: AuE-Referenzmodell

Das Referenzmodell [2], wie es am SE gelehrt und genutzt wird (siehe Abbildung 1), umfasst neben dem Bereich des reinen Requirements Engineering auch Teile des Software-Entwurfes, soweit sie unmittelbar mit Anforderungen zusammenhängen. Es bietet sich an, dieses vorhandene Modell als Ausgangsmaterial zu übernehmen.

Das Referenzmodell ist für den Softwareingenieur hilfreich, um seine Aktivitäten im Bereich der Anforderungen und des Entwurfes zu koordinieren und obwohl es keinen strengen Ablaufplan darstellt, ist es als Checkliste gut geeignet, da alle wichtigen Aktivitäten im Laufe eines normalen Softwareprojektes im groben Zusammenhang skizziert werden.

Es ordnet Begriffe des Software Anforderung und Entwurf (AuE) in einer baumartigen Struktur an, um das Themengebiet zu kategorisieren. Dabei wird AuE zunächst in die Teilbereiche „Requirements Engineering“ (RE) und Software-Entwurf unterteilt. Zum RE gehört die Systemanalyse (oder Requirement Analysis, RA) und das Requirements Management. Beide Teilbereiche lassen sich immer weiter unterteilen und spezialisieren.

Will man also Dokumente anhand dieses Modells klassifizieren, muss man eine baumartige Kategorienstruktur modellieren. Abschnitt 4.2.3 geht näher darauf ein.

## 2.2 Ähnliche und angrenzende Systeme

In diesem Abschnitt werden ähnliche und angrenzende Systeme behandelt und erläutert, was ihre Hauptaufgabe ist sowie ob und wie gut sie in der Lage sind, die Anforderungen der Aufgabenstellung zu erfüllen.

Die Auswahl erfolgte mehr oder weniger willkürlich anhand dessen, welche Software am SE bereits genutzt wird oder bekannt ist.

### 2.2.1 Referenzverwaltungssysteme

Die Hauptaufgabe von Programmen zur Verwaltung von Literaturreferenzen liegt darin, Dokumente anhand von Metadaten in einem Katalog zu erfassen, damit sie in Textverarbeitungssysteme als wissenschaftliche Zitate ausgegeben werden können, etwa für die Erstellung von Literaturlisten.

Die Metadaten können zum Beispiel über das Internet vom Verleger bezogen werden. So können die Referenzdaten (Citations) der im Rahmen von IEEE-Konferenzen veröffentlichten Papers (Conference Proceedings) in verschiedenen gängigen Formaten direkt von der Webseite des IEEE [3] in Referenzverwaltungssysteme importiert werden.

Die Ausgabe erfolgt entweder in formatierter Form über ein Plugin in einen Text oder als Export, beispielsweise im XML- oder RTF-Format.

Es gibt einige Standardprogramme für die üblichen Textbearbeitungssysteme, so zum Beispiel EndNote für eine Microsoft-Office-Umgebung oder BibTeX für eine LateX-Umgebung. Der folgende Abschnitt geht exemplarisch auf EndNote ein, da Vorzüge und Einschränkungen entsprechend sind.

#### Beispiel: EndNote

EndNote [4] stellt ein Plugin für Microsoft Word bereit. Das SE verfügt bereits über eine Referenzendatenbank, die für eine Vielzahl an Standardwerken Metadaten wie Titel, Erscheinungsdatum und Autor bereithält.

Die Webseite des IEEE ermöglicht die relativ einfache Übernahme vieler Metadaten, kurzer Zusammenfassungen (Abstracts) und sogar Schlüsselwörter für viele Dokumente. Darüber hinaus handelt es sich um ein bestehendes System, was die Einarbeitung erleichtern würde.

Nachteilig ist jedoch, dass vor allem literaturspezifische Daten erfasst werden und die Handhabung zusätzlicher domänenspezifischer Kategorien relativ unflexibel ist. So stehen, zumindest in der dem SE gegenwärtig zur Verfügung stehenden Version, insgesamt nur sechs frei definierbare Felder zur Verfügung.

Darüber hinaus ist das Programm nicht in der Lage, Mengenanfragen zu bewältigen.

Um die Anforderungen zu erfüllen ist das zu wenig, deshalb kommt EndNote als alleinige Lösung nicht in Frage.

Die Stärken im Im- und Export können aber ausgenutzt werden, um Titel und Autoren von in Frage kommenden Dokumenten schnell zu erfassen und ins zu erstellende System zu importieren. Näheres dazu findet sich in Kapitel 5.4.

### 2.2.2 Ontologie-Editor: Protégé

Protégé [5] ist ein Ontologie-Editor, der als Open Source verfügbar ist und mit dem man Wissensdatenbanken erstellen und Anfragen an diese stellen kann. Die Stärke dieser Software liegt darin, ein Modell der Realität in einer Ontologie abzubilden, so dass automatisiert logische Schlussfolgerungen gezogen werden können. Protégé ist ein mächtiges Werkzeug, speziell wenn es darum geht, eine Wissensdatenbank mit sehr heterogenen Informationen zu erstellen.

Das Programm ist jedoch allein nicht geeignet, komplexe Mengenanfragen zu beantworten. Es ist zum Beispiel nicht möglich, Anfragen mit Unteranfragen zu stellen.

Eine Anpassung des Programms wäre prinzipiell möglich, da der Java-Sourcecode unter einer Open-Source-Lizenz verfügbar ist. Jedoch ist der dafür nötige Aufwand schwer einzuschätzen, der Ansatz wurde daher nicht weiter verfolgt.

### 2.2.3 Bibliothekskatalog: OPAC

Ein Bibliothekskatalog wie OPAC (Online Public Access Catalogue) wird benutzt, um die in einer Bibliothek verfügbaren Medien zu identifizieren und zu verwalten. Dabei wird üblicherweise als kleinste Einheit das physische Objekt erfasst, zum Beispiel ein Buch oder eine Ausgabe eines Magazins. Die einzelnen Kapitel des Buches beziehungsweise Artikel des Magazins werden nicht einzeln erfasst. Der Katalog enthält neben Informationen über den Inhalt auch solche über Standort, Beschaffung und Ausleihe.

Ein OPAC ist elektronisch nach Meta-Daten durchsuchbar, boolesche Operationen sind möglich. Die Anfragemöglichkeiten sind darüber hinaus relativ begrenzt.

Die Erstellung von Mengengerüsten wird nicht unterstützt. Da das SE auch nicht über vorhandene OPAC-Software verfügt, kommt eine Erweiterung eines bestehenden Systems nicht in Frage.

### 2.2.4 Suchmaschine: Google Scholar

Die Hauptaufgabe von Suchmaschinen wie Google Scholar [6] ist es, Dokumente zu finden, deren Speicherort zuvor nicht bekannt ist. Zu den dazu eingesetzten Techniken gehört die Erstellung eines Volltextregisters (Data Mining). Das Anlegen eines eigenen domänenspezifischen Datenbestandes ist jedoch ebenso wie die Erstellung von Trends oder Mengengerüsten nicht möglich.

Eine Suchmaschine kann benutzt werden, um interessante Dokumente im Internet aufzufinden, die weitere Verarbeitung muss jedoch in einem anderen System erfolgen.

### 2.2.5 Spezialdatenbanken: SE-Map

Ein bestehendes angrenzendes System ist die SE-Map [7] des Fachgebiets Software Engineering. In dieser Datenbank werden Zusammenhänge zwischen Personen, Instituten und Orten erfasst und geben einen Überblick, welche Personen an welchen Themen forschen.

Die SE-Map umfasst teilnehmende Personen und Institute in ganz Deutschland, geht jedoch nicht bis hinunter auf die Ebene einzelner Publikationen.

Diese Datenbank ist für die Lösung der Aufgabe nicht relevant, könnte jedoch in der Zukunft eine mögliche Erweiterung darstellen, falls zum Beispiel Aussagen über das Wirken einzelner Personen getroffen werden sollen.

# 3 Workflow und Rollen

## 3.1 Allgemeines

In diesem Kapitel wird erläutert, wie die Bewertung eines Dokumentes zustande kommt, welche Abläufe dabei eine Rolle spielen und welche Personen daran in welchen Rollen beteiligt sind.

**Workflow**

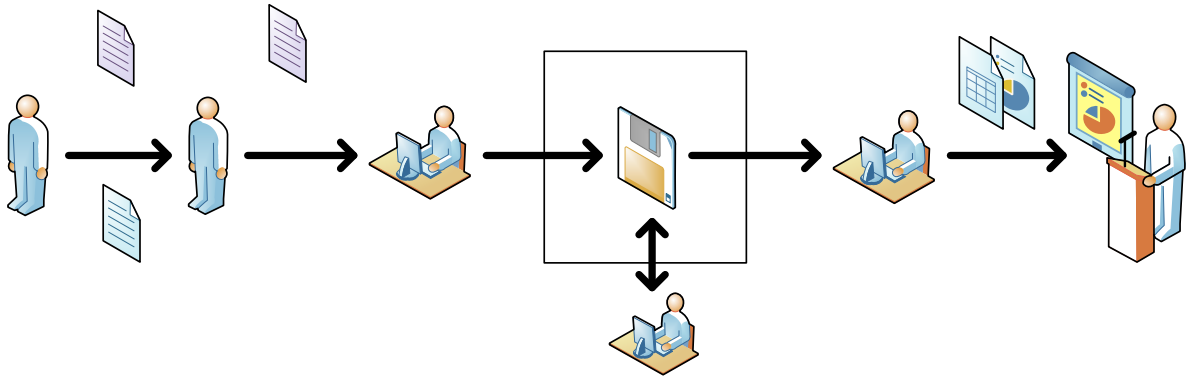


Abbildung 2: Workflow und beteiligte Personen

Der grobe Ablauf (siehe Abbildung 1) ist, dass ein Experte oder Aufraggeber (Rolle: „Experte“) zu der vorgegebenen Fachliteratur ein Bewertungsschema mit den zu bewertenden Kategorien erstellt und dieses mit den Dokumenten an einen oder mehrere Fachpersonen (Rolle: „Leser“) weitergibt, die anhand der Regeln eine Einordnung jedes einzelnen Dokumentes vornehmen. Diese Einordnung halten sie auf einem Erfassungsbogen fest. Alle Erfassungsbögen werden von einer Schreibkraft (Rolle: „Schreibkraft“) ins System eingegeben.

Die Daten werden von einer weiteren Fachperson (Rolle: „Analytiker“) zur Analyse abgerufen und aufbereitet um schließlich vom Experten wissenschaftlich ausgewertet oder präsentiert zu werden.

Die einzelnen Rollen können durchaus zusammenfallen. So können alle Schritte von der einer Person allein von Anfang bis Ende ausgeführt werden oder die Schreibkraft übersprungen werden, wenn alle Leser ihre Ergebnisse direkt ins System einpflegen.

Weitestgehend außerhalb des Workflows steht der Administrator (Rolle: „Admin“), dessen Aufgabe überwiegend technischer Natur sind.

## 3.2 Rollen

Im Folgenden werden nochmals die einzelnen Rollen und ihre Motivationen im Detail beschrieben. Außerdem werden typische Anwendungsfälle beschrieben.

### 3.2.1 Experte

Der Experte benötigt einen Überblick über aktuelle Trends in seinem Forschungsgebiet oder einer Anwendungsdomäne. Es ist eine größere Menge Literatur zu sichten und einzuordnen und die allgemeine Relevanz zu bewerten.

Der Experte bestimmt, welche Kriterien bei der Einordnung zu berücksichtigen sind und erstellt entsprechende Regeln, die von allen nachfolgenden Bearbeitern/Rollen zu befolgen sind, um ein einheitliches Ergebnis zu erhalten.

Am Ende des Prozesses erhält er so aus den Dokumenten die ihn interessierenden Daten und kann sie für weitere Forschungen oder zur Präsentation nutzen.

Der Experte steht außerhalb der Systemgrenzen, da er keine direkte Interaktion mit dem System hat, beeinflusst es aber indirekt, indem er festlegt, was genau im System gespeichert wird. Da das System sehr flexibel ist, legen diese Regeln den Rahmen fest in dem die Realität in Form von Research Papers in natürlicher Sprache in ein den Anforderungen angemessenes maschinenlesbares Modell abgebildet wird.

*Beispiel:* *Um eine These zu untermauern, möchte Professor X wissen, was in den vergangenen Jahren die Top Themen auf den internationalen Konferenzen zum Requirements Engineering waren. Eine Abschätzung aus dem Gedächtnis reicht nicht aus, da es besonders bei den älteren Papers ist es schon eine Weile her ist, dass er sie zuletzt gelesen hat.*
*Er legt unter anderem fest, dass Schlüsselworte auf Englisch zu erfassen sind und in das Referenzmodell zum Requirements Engineering eingeordnet werden sollen.*

### 3.2.2 Leser

Der oder die Leser klassifizieren die ihnen zugewiesenen Dokumente anhand der Regeln, die der Experte aufgestellt hat. Auch der Leser gehört nicht direkt zum System, beeinflusst es aber, da er die Rohdaten für die maschinelle Erfassung vorbereitet.

Der Leser füllt entweder einen Erfassungsbogen aus, auf dem er seine Klassifikation notiert oder gibt seine Ergebnisse direkt ins System ein (siehe Abschnitt 3.2.3).

*Beispiel:* *Professor X bittet seinen wissenschaftlichen Mitarbeiter A und B, einen Teil der Papers zu übernehmen, um ihn zu entlasten. Er trifft sich mit ihnen, um die Erfassungsregeln kurz zu besprechen und teilt jedem die von ihm zu klassifizierenden Dokumente zu.*
*Mitarbeiter A liest seinen Anteil der Papers in seinem Büro und gibt seine Ergebnisse direkt ins System ein.*
*Mitarbeiter B nimmt seine Dokumente mit ins Wochenende. Dort hat er keinen Zugriff aufs System und füllt deshalb für jedes Paper einen Erfassungsbogen aus.*

### 3.2.3 Schreibkraft (Benutzer)

Die Schreibkraft erhält die Klassifikationsdaten der Leser und gibt sie ins System ein. Diese Rolle kann entweder eine eigenständige Person sein, die die Erfassungsbögen der Leser ins System überträgt oder es sind die Leser selbst, die anschließend an ihre lesende Tätigkeit die Rolle wechseln und ihre Ergebnisse selbst ins System eingeben.

In jedem Fall ist die Arbeit mit der Literaturdatenbank nur ein untergeordneter Teil ihres Tagesablaufes, es kommt also darauf an, dass möglichst viele Datensätze pro Zeit eingegeben werden können.

*Beispiel:* *Da es eilt und er keine Zeit hat, übergibt Professor X seine Notizen an die studentische Hilfskraft S, die die Daten ins System überträgt. Sie hat zwei Stunden Zeit, bevor im Nachbargebäude ihre nächste Vorlesung beginnt. Bis dahin müssen die Daten eingegeben sein.*

### 3.2.4 Analytiker (Benutzer)

Der Analytiker arbeitet mit dem Datenbestand, wenn genug Daten eingegeben wurden. Seine Aufgabe ist es, aus den dem Datenmodell folgenden maschinenlesbaren Daten eine für Menschen verständliche Form der Realität zu erstellen, sei es die Antwort auf eine Frage, eine Diagramm oder eine Zahlenreihe.

*Beispiel:* *Doktorand D lässt sich für jedes betrachtete Jahr die zehn meistgenannten Schlüsselwörter ausgeben. Diese stellt er in einem Tabellenkalkulationsprogramm übersichtlich zusammen. Anschließend trifft er sich mit Professor X, um mit ihm zu besprechen, ob die ermittelten Daten den Voraussagen entsprechen.*

### 3.2.5 Admin (Benutzer)

Die Aufgaben des Administrators sind überwiegend technischer Natur, so ist er für regelmäßige Backups zuständig sowie für eventuell notwendige Erweiterungen oder Änderungen.

# 4 Konzept

In diesem Kapitel werden die grundlegenden Konzepte erläutert. Im ersten Abschnitt werden die Anforderungen an das Benutzerinterface beschrieben. Im zweiten Abschnitt erfolgen die Beschreibung der Datenstruktur, ihrer zentralen Objekte und ihrer Relationen sowie die wichtigsten Operationen, um den Begriffsbaum zu verändern oder um Informationen aus ihm zu erhalten. Der dritte Abschnitt behandelt Anfragen an die Literaturdatenbank sowie ihre Präsentation.

## *4.1 Benutzerinterface*

### 4.1.1 Anforderungen an die GUI

Programme, deren Bedienung kompliziert oder schwer erlernbar ist, werden aller Voraussicht nach weniger gern benutzt und unzureichend gepflegt. Daher ist es wichtig, dass die Applikation für den Benutzer möglichst schnell und gut bedienbar ist. Offensichtlich muss beim Entwurf verstärkt auf Usability, vor allem auf Ergonomie geachtet werden.

Der normale Nutzer ist, unabhängig davon welche Rolle (siehe Abschnitt 3.2) er ausfüllt, ein erfahrener PC-User, der nach anfänglicher Einarbeitung sämtliche Eingaben routinemäßig vornimmt. Die Bedienoberfläche kann somit schlicht und ohne viele Erläuterungen und Hilfestellungen programmiert werden.

Der kritischste Punkt ist die kompakte Darstellung des Kategorienbaumes. Für jedes einzuordnende Dokument muss er durchgesehen werden, um alle Verknüpfungen an den richtigen Stellen zu setzen. Doch schon die anfängliche Entwurfsversion des Schemas umfasste ohne Aufteilung mehrere Bildschirmseiten, durch die ständig navigiert werden müsste. Das ist jedoch unergonomisch und unübersichtlich und würde den Eingabeprozess verzögern. Es ist also unbedingt nötig, den Kategorienbaum in Teilbäume aufzuteilen, die eine Bildschirmseite nach Möglichkeit nicht überschreiten sollten.

## 4.2 Datenstruktur

Um zu einer verwendbaren Datenstruktur zu gelangen, müssen die relevanten Objekte, identifiziert werden sowie ihre Beziehungen untereinander.

Im vorliegenden Fall werden Dokumente in ein thematisch angeordnetes Modell eingeordnet. Zu modellieren sind also Klassen für Dokument und Kategorienbaum.

Der Kategorienbaum setzt sich zusammen aus vielen einzelnen Kategorien, die nach bestimmten Regeln angeordnet sind und den Operationen, die notwendig sind, um den Baum zu verändern oder etwas darin zu finden.

### 4.2.1 Dokumente

Zu einem Dokument werden hauptsächlich seine Attribute gespeichert. Ein großer Teil der Eigenschaften beschreibt das Dokument an sich. Angelehnt an die Elemente des Dublin Core [8] werden der Titel, die Autoren und das Erscheinungsjahr gespeichert. Je nachdem, welche Daten zur Verfügung stehen, lässt sich die Liste der Attribute erweitern.

Es ist vorgesehen, dass jedes Dokument nur einmal und von einer Person klassifiziert wird. Daher können Daten, die den Status der Erfassung betreffen, als Attribute des Dokumentes betrachtet werden. Dazu gehört auch ob und von wem es erfasst wurde, ob ein Bogen angelegt wurde oder ob die Klassifikation direkt ins System eingegeben wurde sowie die Domänenrelevanz sowie allgemeine Notizen.

Die Datenstruktur der Dokumente hat keinen Bezug auf sich selbst und nur zu einer anderen Objektklasse (Kategorie). Sie benötigt daher nur einfache Operationen für Ein- und Ausgabe. Auf Datenbankebene sind das die Operationen Ausgabe (Select), Einfügen (Insert), Bearbeiten (Update) und Löschen (Delete).

Die einzige etwas kritischere Operation darunter ist das Löschen, da auf verknüpfte Kategorien zu achten ist. Allgemein kann man sagen, dass ein Dokument erst dann gelöscht werden darf, wenn es mit keiner Kategorie mehr verknüpft ist. Eine Operation, die Dokumente löscht, muss daher erst alle Verknüpfungen löschen, bevor das Dokument selbst gelöscht werden kann (siehe Codebeispiel 1).

```
function deleteDocument(docID) {
        deleteCategoryLinks(docID);
        deleteDocumentFromDB(docID);
}
```

Codebeispiel 1: Dokument löschen

Um einen Verlust der Konsistenz zu vermeiden oder eine Wiederherstellung zu ermöglichen, könnte eine Löschoperation auch so aussehen, dass ein Dokument und die zugehörigen Verknüpfungen nicht gelöscht, sondern nur ausgeblendet werden. Dazu kann man Felder für die Darstellung vorzusehen, die zum Beispiel anzeigen, ob ein Dokument in der Auflistung angezeigt werden soll oder ob es beispielsweise zum Löschen markiert wurde.

### 4.2.2 Kategorien

Eine Kategorie kann mehrere Unterkategorien haben, das bedeutet, dass die Struktur hierarchisch ist. Die Beschränkung auf eine einzige Oberkategorie führt zu einer Baumstruktur, d.h. es gibt eine Wurzel, an der Teilbäume hängen, deren Kinder wiederum Bäume sind. Prinzipiell ist es möglich, eine Kategorie mehreren Elternkategorien zuzuordnen. Das macht die Baumoperationen jedoch komplizierter und bietet für die konkrete Aufgabenstellung keinen Mehrwert.

Jede Kategorie bekommt zur Identifikation eine laufende Nummer und als weitere Attribute einen Titel sowie eine Beschreibung, denn der Titel kann durchaus eine Abkürzung sein. Die Beschreibung ist optional.

Zusätzlich benötigt jede Kategorie eine Referenz auf die jeweilige Elternkategorie. Das ist in einer Datenbanktabelle durch eine Fremdschlüsselbeziehung auf die eigene Tabelle möglich.

Als weitere Attribute sind Darstellungsoptionen denkbar, um beispielsweise auswählen zu können, welche Attribute der Verknüpfungsrelation zwischen Dokument und Kategorie eingegeben werden sollen oder um eine Kategorie zum Löschen zu markieren.

Um auf einzelne Kategorien-Datensätze zuzugreifen, werden Methoden zum Lesen und Schreiben der Attribute benötigt. Auf Datenbankebene sind das Methoden zum Einfügen (insert), Bearbeiten (update) und zum Löschen (delete). Wie bei den Dokumenten ist darauf zu achten, dass eine Kategorie nur dann gelöscht werden darf, wenn sie leer ist. Das heißt, es dürfen keine Dokumente mit ihr verknüpft sein und zusätzlich darf sie keine Unterkategorien haben. Verknüpfungen und Unterkategorien müssen daher zuerst entfernt oder verschoben werden.

Für die Navigation im Baum werden Methoden benötigt, um auf Vorgänger- oder Nachfolgerknoten zugreifen zu können. Dabei kann man sich an den Achsen aus XPATH [9], einer Anfragesprache für XML-Dokumente, orientieren. XPATH kennt eine Reihe von Bezeichnungen für Knoten, die sich relativ zum aktuellen Knoten befinden. So heißen direkte Kindknoten „child“ und die Operation, die die Kindkategorien einer gegebenen Kategorie zurückgibt, heißt „getChildren“. Entsprechend sind Vorgängerkategorien „ancestors“ (mit der Operation getAncestors), Nachfolger sind „descendants“ und Kategorien mit dem gleichen Elternknoten heißen „siblings“ (Geschwister). Für jeden dieser Aspekte kann man auch die Existenz prüfen, so prüft zum Beispiel „hasChildren“, ob eine bestimmte Kategorie Nachfolger hat.

Auf diese und andere Methoden den Kategorienbaum zu durchlaufen und zu verändern, wird in Abschnitt 4.3 näher eingegangen. Eine Aufzählung dieser Operationen findet sich in Anhang B.

### 4.2.3 Verknüpfung von Dokument und Kategorie

In einer einfachen Literaturverwaltung gibt es eine Liste von Dokumenten und eine Liste von Kategorien. Jedes Dokument wird einer oder mehrerer Kategorien zugeordnet.

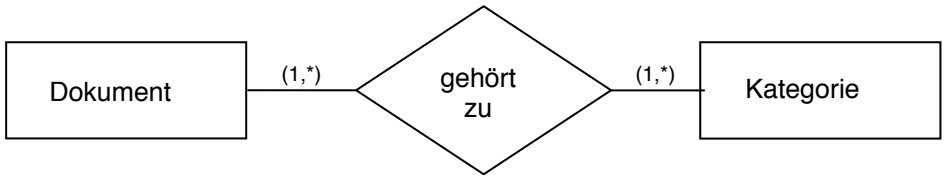


Abbildung 3: einfache Literaturverwaltung

Wie bereits dargestellt, sind die Kategorien hierarchisch angeordnet, wobei jede Kategorie genau eine Oberkategorie sowie beliebig viele oder keine Unterkategorien hat.

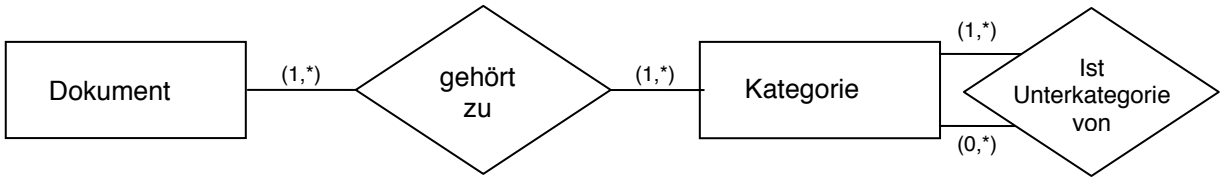


Abbildung 4: Literaturverwaltung mit hierarchischen Kategorien

#### Resultierende Klasse: Klassifikation

Die Klassifikation stellt die Verknüpfung zwischen Dokumenten und Kategorien dar. Neben einer laufenden Nummer zur Identifikation enthält die Tabelle für die Klassifikation daher Fremdschlüssel für Dokument und Kategorie.

Wie bei Dokumenten und Kategorien benötigt man Methoden zum Lesen und Schreiben. Schreiboperationen sind Einfügen (insert), Bearbeiten (update) und Löschen (delete). Im Gegensatz zu den vorher genannten ist für die Klassifikation alleine die Löschoperation nicht kritisch, da nur eine Fremdschlüsselbeziehung gelöscht wird. Nützlich sind auch Operationen, die ein Dokument an eine andere Kategorie oder eine Kategorie an ein anderes Dokument verschieben können, zum Beispiel wenn der ursprüngliche Datensatz gelöscht werden soll.

Weitere notwendige Leseoperationen sind Methoden, die prüfen, welche Verknüpfungen mit einem gegebenen Dokument oder einer gegebenen Kategorie bestehen.

## 4.3 Baumoperationen

Dieser Abschnitt beschreibt die wichtigsten Operationen im Kategorienbaum. Zum einen Operationen, um den Baum zu verändern zum anderen, um Operationen aus dem Baum zu erhalten.

### 4.3.1 Baumreorganisation

Der Kategorienbaum benötigt nicht nur Methoden zur Ausgabe sondern er wird auch ständig verändert. Wenn er sich nachträglich stark ändert, ist unter Umständen ist eine Reorganisation nötig. Das bedeutet, dass nicht oder nur wenig benutzte Kategorien zusammengefasst werden oder an eine andere Stelle im Baum verschoben werden.

Wie bereits in Abschnitt 4.2.2 angesprochen, sind die wichtigsten Schreiboperationen das Einfügen, Bearbeiten und das Löschen. Aus diesen Grundoperationen werden alle weiteren Operationen, wie zum Beispiel das Verschieben von Kategorien, zusammengesetzt. Da die Datenstruktur hierarchisch ist, sind diese Operationen nicht immer trivial.

#### Zyklenfreiheit

Der Kategorienbaum darf keine Zyklen enthalten, da komplexe Baumoperationen wie die Aggregation rekursive Aufrufe enthalten. Enthält nun der Kategoriengraph einen oder mehrere Zyklen, ist es möglich, dass eine Endlosschleife entsteht, die die Anwendung zum Absturz bringt. Einzelne Kategorien könnten überhaupt nicht mehr von der Wurzel aus zu erreichen sein.

Dies lässt sich umgehen, wenn bei jedem Updatevorgang geprüft wird, ob die zu bearbeitende Kategorie Vorgänger der neuen Elternkategorie ist. Eine Kategorie darf außerdem nicht sich selbst zur Oberkategorie haben.

```
function isAncestorOrSelf(cat1, cat2) {
        if (cat1 == cat2) return true;
        parent = cat2;
        while (parent != getRoot()) {
                if (parent == cat1) return true;
                parent = getParent(parent);
        }
        return false;
}
```

Codebeispiel 2: Vermeidung von Zyklen

Codebeispiel 2 zeigt die Vorgehensweise: Es wird geprüft, ob eine Kategorie cat1 Vorgänger der Kategorie cat2 oder die Kategorie cat2 selbst ist. Sind beide Kategorien gleich, gibt die Funktion den Wert true zurück. Ist das nicht der Fall, wird von Kategorie cat2 ausgehend der Baum von unten nach oben vertikal durchlaufen und geprüft, ob die jeweilige Elternkategorie cat1 ist. Ist das der Fall, wir wiederum true zurückgegeben. Ist man an der Wurzel angekommen ohne eine Übereinstimmung gefunden zu haben, ist cat1 nicht Vorgänger von cat2 und es wird false zurückgegeben.

### Einfügen und Bearbeiten

Zum Einfügen einer Kategorie in den Kategorienbaum benötigt man mindestens ihren Titel und die gewünschte Elternkategorie. Dazu muss der Titel eindeutig sein und die Elternkategorie muss auf Vorhandensein geprüft werden.

Das Bearbeiten ist auch nicht kritisch, es sei denn, die Elternkategorie soll verändert werden. In diesem Fall handelt es sich um eine Verschiebung.

### Kategorie verschieben

Abgesehen davon, dass keine Zyklen entstehen dürfen, ist das Verschieben einer Kategorie sehr einfach. Man wählt eine neue Elternkategorie aus und die Kategorie wird mit allen Unterkategorien und angehängter Dokumente an eine andere Stelle im Baum verschoben.

```
function moveCategory(catID, newParent) {
        if (!isAncestorOrSelf(catID, newParent))
                setParent(catID, newParent);
}
```

Codebeispiel 3: Kategorie verschieben

### Kategorie löschen

Eine Kategorie darf nur dann gelöscht werden, wenn sie leer ist. Das heißt, es dürfen keine Dokumente mit ihr verknüpft sein und sie darf keine Kindkategorien haben.

Die einfachste Vorgehensweise ist, alle verknüpften Dokumente und alle Kindkategorien zu finden und sie an die Elternkategorie anzuhängen. Das ist auch die einzige, die ohne zusätzliche Informationen automatisch durchgeführt werden kann. Wird eine differenziertere Zuordnung gewünscht, muss sie manuell vorgenommen werden.

```
function deleteCategory(catID) {
        new_parent = getParent(catID);
        if (hasLinkedDocuments(catID)) {
                linked_docs = getLinkedDocuments(catID);
                foreach (linked_docs as doclink) {
                        linkDocToNewParent(doclink, new_parent);
                }
        if (hasChildren (catID)) {
                children = getChildren(catID);
                foreach (children as child) {
                        setParent(child, new_parent);
                }
        deleteCategoryFromDB(catID);
}
```

Codebeispiel 4: Kategorie löschen

### 4.3.2 Aggregation

Um Kategorien mitsamt ihrer Unterkategorien durchsuchen zu können, müssen diese zusammengefasst werden. Diese Zusammenfassung nennt man Aggregation.

Ziel der Aggregation ist es, alle Dokumente zu finden, die einer Kategorie inklusive ihrer aller Unterkategorien angehört.

Die Vorgehensweise ist hier, zuerst die IDs aller Kategorien zu ermitteln, die auf einer gegebenen Ebene unter der Startkategorie liegen. Dann wird für jede dieser Kategorien getrennt eine Liste aller Nachfolgekategorien erstellt und in einem weiteren Schritt alle Dokumente gezählt, die mit mindestens einer der Kategorien aus der Liste verknüpft sind. Codebeispiel 5 stellt dies im Pseudocode dar.

```
Ermittle alle Nachfolgekategorien auf einer bestimmten Ebene unterhalb der Startkategorie.

DescendantsOfLevelX = getDescendantsOfLevel(x)

Ermittle alle Nachfolgekategorien dieser Kategorien und schreibe sie in eine Liste.

foreach (descendantOfLevel as child) {
        descendantsOfChild = getDescendants(child)
}

Ermittle alle angehängten Dokumente in diesen Kategorien, bzw. deren Anzahl

foreach descendantOfLevel {
        count = getLinkedDocuments(descendantsOfChild)
}
```

Codebeispiel 5: Aggregation

## 4.4 Anfragen und Präsentation

In einem Gespräch mit Experten (im Sinne der Rolle des Experten, siehe Kapitel 3) ergab sich eine Liste möglicher Anfragen. Wichtig war vor allem, ob die potentielle Ausgabe eine sinnvolle Aussage ermöglicht und weiterverarbeitet werden kann.

Diese Anfragen teilen sich in drei größere Gruppen auf. So gibt es Anfragen, welche Dokumente bestimmte Merkmale erfüllen und Anfragen, wie viele Dokumente das jeweils sind und als dritte Gruppe komplexe Anfragen. Jede dieser Standardanfragen wird im folgenden Abschnitt erläutert.

### 4.4.1 Alphanumerische Liste

Eine wichtige Fragestellung ist, welche Dokumente bestimmte Bedingungen erfüllen. Das kann zum Beispiel sein, um ein bestimmtes Dokument wieder zu finden oder um eine Liste von Vorschlägen zu bekommen.

Die Eingabe besteht hierfür aus einer Liste von Eigenschaften des Dokuments sowie einer Liste von Kategorien, denen das Dokument zugeordnet ist. Das lässt sich erweitern durch boolsche Operatoren, eine Auswahl wie genau die Kriterien getroffen werden müssen (exakt, Teilstring, regulärer Ausdruck, ähnlich) oder ob Unterkategorien in die Suche einbezogen werden sollen. Darüber hinaus kann man angeben, welche Metadaten der Dokumente ausgegeben werden sollen und wonach und in welcher Richtung sie sortiert werden sollen.

Diese Anfragenbedingungen müssen nun von der Anwendung in eine oder mehrere SQL-Anfragen übersetzt werden.

In einem ersten Schritt wird die Liste der Kategorien zusammengestellt, in denen gesucht werden muss. Dafür wird eine Liste aller in den Bedingungen erwähnten Kategorien und ihrer Unterkategorien erstellt. Dazu wird eine Aggregationsfunktion benutzt (siehe Kapitel 4.3.2).

In einem zweiten Schritt werden nun diejenigen Dokumente gesucht, die mit Kategorien aus in der in Schritt Eins erhaltenen Liste verknüpft sind und die weiteren dokumentspezifischen Kriterien erfüllen.

Die Ausgabe ist eine Liste von Dokumenten, die diesen Kriterien entsprechen.

| Beispiel: | |
|---|---|
| Grundfrage: | Welche Dokumente behandeln neue Elicitation-Techniken? |
| Prinzip: | • *„Suche alle Nachfolgekategorien von ‚Elicitation' und ‚Techniken'"*<br>• *„Suche die Titel der Dokumente, die in einer Nachfolgekategorie von Elicitation und in einer Nachfolgekategorie von Techniken enthalten sind und höchstens ein Jahr alt sind"* |
| Ausgabe: | Liste von Titeln |
| Darstellung: | Tabelle / Liste |

Tabelle 1: Alphanumerische Liste

### 4.4.2 Mengengerüst

Ein Mengengerüst beschreibt, wie der Name schon sagt, die Anzahl der Dokumente, die bestimmte Kriterien erfüllen. Mengengerüste ermittelt man, wenn man nicht an konkreten Dokumenten sondern an ihrer Anzahl interessiert ist.

Die Anfrage funktioniert genau wie im vorherigen Abschnitt, nur dass keine Titel ausgegeben werden sondern die Anzahl zutreffender Datensätze.

Die Eingabe ist entsprechend, wenn auch keine Sortierkriterien angegeben werden.

| Beispiel: | |
|---|---|
| Grundfrage: | Welche(r) Bereich(e) des RE wurde(n) am häufigsten behandelt? |
| Prinzip: | • *Suche die Kindkategorien von „RE“ und alle deren Nachfolgekategorien*<br>• *Suche die Kategorie(n) mit der maximalen Anzahl Dokumente* |
| Ausgabe: | Liste von Kategorien und zugehörigen Anzahlen |
| Darstellung: | Säulendiagramm, Tortendiagramm |

Tabelle 2: Mengengerüst

Die Ausgabe ist eine Liste von Kategorien und dazugehörige Anzahlen. Diese kann man beispielsweise als Säulen- oder Tortendiagramm darstellen, je nachdem, ob man eine Aussage über absolute Zahlen oder die prozentuale Verteilung treffen will.

Abbildung 5 zeigt, wie das Ergebnis einer solchen Anfrage aussehen könnte.

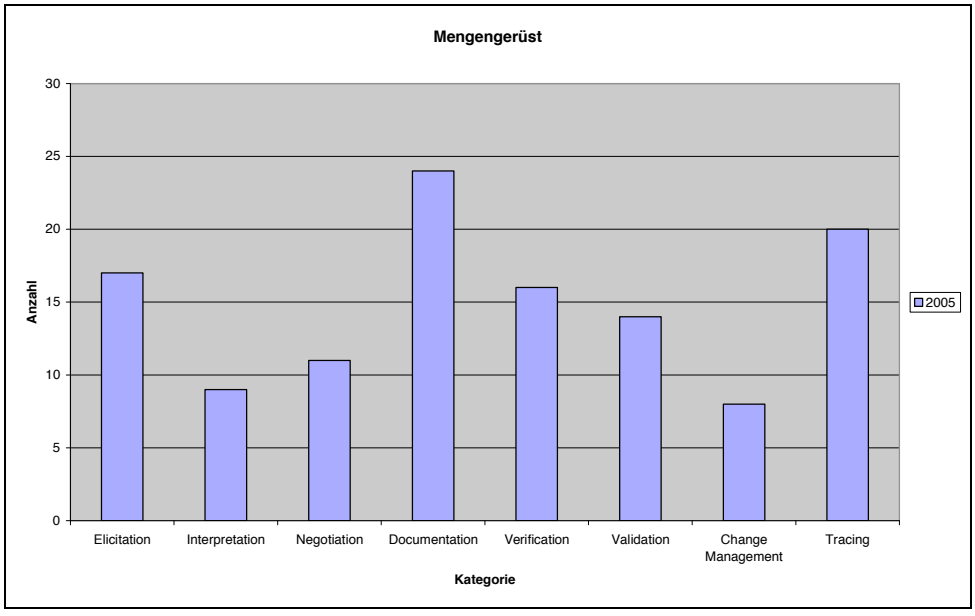


Abbildung 5: Mengengerüst als Säulendiagramm

### 4.4.3 Trend

Ein Trend entspricht im Wesentlichen einem Mengengerüst, nur dass die gleichen Kriterien zu verschiedenen Zeitpunkten betrachtet werden. Trends werden ermittelt, um zu erkennen, wie sich die Publikationszahlen zu einem bestimmten Thema im Lauf der Jahre ändern, zum Beispiel um Themen zu finden, die momentan besondere Aufmerksamkeit genießen.

Trends sind im Grunde nichts anderes als Mengengerüste pro Zeit. Und so werden sie auch ermittelt.

| Beispiel: | |
|---|---|
| Grundfrage: | Wie viele Dokumente erschienen in den letzten beiden Jahren in den einzelnen Bereichen des RE? |
| Prinzip: | • *Suche die Nachfolgekategorien von „RE“ einer bestimmten Ebene und ermittle alle deren Nachfolgekategorien*<br>• *Ermittle die Anzahl Dokumente in den letzten beiden Jahren in diesen Kategorien* |
| Ausgabe: | Liste von Kategorien und Jahren und zugehörigen Anzahlen |
| Darstellung: | Säulendiagramm |

Tabelle 3: Trend

Die Ausgabe ist wie beim Mengengerüst eine Liste von Kategorien und zugehörige Anzahlen, jedoch entsteht für jedes betrachtete Jahr eine weitere Spalte.

Für die Darstellung bietet sich ebenfalls ein Säulendiagramm an, wie Abbildung 6 zeigt.

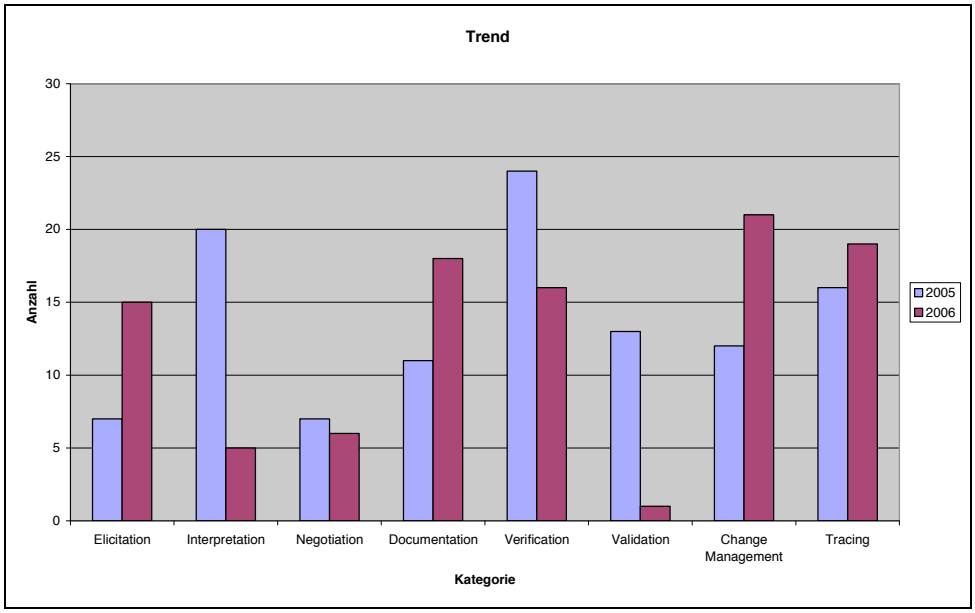


Abbildung 6: Trend als Säulendiagramm

### 4.4.4 Portfolio

Portfolios sind eine spezielle Art der Anfrage, die es erlaubt, domänenspezifische Publikationen im allgemeinwissenschaftlichen Zusammenhang zu bewerten. So erlauben sie, für verschiedene Mitbewerber in einer Domäne darzustellen, wie sich die eigenen Publikationen in den Gesamtkontext einfügen.

Für Portfolios ist es notwendig, dass die Relevanz der Dokumente erfasst wurde, zum einen die Relevanz für das Gebiet des Requirements Engineering zum anderen die Relevanz in Bezug auf die betrachtete Domäne.

Die eigentliche Anfrage besteht darin, zwei Mengengerüste zu erstellen: Beide ermitteln die Anzahl der Dokumente, auf die die Kriterien zutreffen. Eins zählt nur Dokumente, die eine Mindestrelevanz im Gebiet des RE haben und das andere nur diejenigen, die eine Mindestrelevanz in Bezug auf die Anwendungsdomäne haben.

Da die Relevanz eine rein subjektive Bewertung ist, ist auch das Ergebnis einer daraus resultierenden Anfrage eine Zusammenfassung der Expertenmeinung. Eine Literaturdatenbank kann in diesem Fall nur Unterstützung bieten.

| Beispiel: | |
|---|---|
| Grundfrage: | Wie verhalten sich domänenspezifische Publikationen im Vergleich zum allgemeinwissenschaftlichen Trend in Bezug auf bestimmte Kategorien? |
| Prinzip: | • *Wie viele wissenschaftliche Dokumente gibt es zu einem Thema und wie viele domänenspezifische?* |
| Ausgabe: | Schlagwörter mit Domänen- und RE-Relevanz |
| Darstellung: | Als XY-Diagramm |

Tabelle 4: Portfolio-Darstellung

Die Ausgabe ist eine Liste von Kategorien beziehungsweise Schlagwörtern und jeweils zwei Werten für die Relevanz. Daraus wird ein zweidimensionales Diagramm, bei dem die Relevanzen als X- und Y-Wert aufgetragen werden.

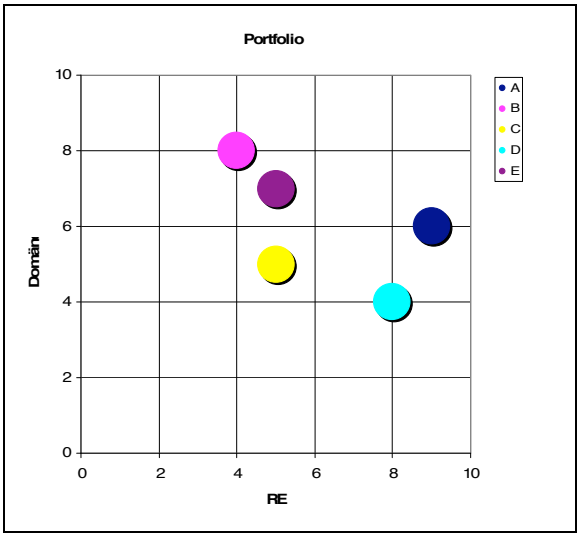


Abbildung 7: Portfolio Diagramm

# 5 Entwurf

In diesem Kapitel geht es um den Entwurf der Anwendung. Im ersten Abschnitt wird die Wahl der Architektur erläutert und begründet. Im zweiten Abschnitt geht es um den modularen Aufbau nach dem MVC-Entwurfsmuster. Der dritte Abschnitt behandelt den Entwurf der graphischen Oberfläche.

## *5.1 Wahl der Architektur*

Eine der ersten Entwurfsentscheidungen ist die Wahl der Architektur. Dabei standen mehrere unterschiedliche Ansätze zur Verfügung.

Eine Möglichkeit ist, die Applikation als Webseite zu implementieren, die auf einem Server installiert wird und von allen Rechnern im Netzwerk über einen herkömmlichen Webbrowser bedient wird, wobei mehrere Programmiersprachen in Frage kommen. Alternativ kann man eine Implementierung als Java-Applikation wählen oder man setzt auf Microsoft-Produkte und realisiert eine Anwendung auf der Basis von .NET, Access und weiteren Office-Produkten.

Nach Abschätzung aller Vor- und Nachteile, die in den folgenden Abschnitten erläutert werden, fiel die Entscheidung zugunsten einer Implementierung als Webseite mit PHP/MySQL entschieden. Ausschlaggebend waren neben der vollen Unterstützung von SQL inklusive Unterabfragen die Mehrbenutzerfähigkeit sowie die Sicherheit einer bewährten Anwendung.

### 5.1.1 Datenbank-Architektur

Grundsätzlich sind zwei Sorten von Datenbanken zu unterscheiden: Serverbasierte Datenbanken wie MySQL oder Oracle auf der einen Seite und dateibasierte Datenbanken wie MS-Access oder XML-Dateien auf der anderen Seite.

#### serverbasierte Datenbank

Der Vorteil serverbasierter Datenbanken ist, dass Features der Serversoftware genutzt werden können. Die Mehrbenutzerfähigkeit ist ebenso implementiert wie Sicherheitsaspekte. Dazu unterstützen sie in hohem Maße den Sprachstandard von SQL.

Die Datenbanksoftware MySQL ist darüber hinaus kostenlos für viele Betriebssysteme verfügbar, ebenso Treiber für eine ganze Reihe Programmiersprachen. Eine einmal erstellte Datenbank kann also ohne größere Probleme von verschiedenen Anwendungen oder im Netzwerk benutzt werden.

Nachteilig ist hingegen, dass überhaupt ein Server installiert werden muss.

#### dateibasierte Datenbank

Die Datenbank mit der größten Verbreitung ist wahrscheinlich Microsoft Access, das Teil der Office Suite von Microsoft ist. Sein Vorteil liegt vor allem in der relativ hohen Verfügbarkeit von Microsoft Office. In der Wirtschaft werden Lösungen auf Basis von Office-Produkten begrüßt, da es zu den Standardanwendungen gehört. Im akademischen Bereich greift man aus Kostengründen oft lieber auf Open Source Software zurück.

Eine andere Möglichkeit, eine dateibasierte Datenbank zu realisieren, ist in Form einer Textdatei in einem XML-Format. Das würde es ermöglichen, ein dichteres semantisches Netz zu erstellen als eine einfache Baumstruktur, da die für Ontologien verwendeten XML-Dialekte RDF und OWL sehr mächtig sind.

Leider sind die Möglichkeiten, Anfragen an eine derartige Datenbasis zu stellen, schwieriger, da die zur Verfügung stehenden Anfragesprachen nicht an die Möglichkeiten von SQL heranreichen.

Eine Mehrbenutzerfähigkeit ist nur mit Mehraufwand möglich, außerdem muss eine eigene Implementierung sicherstellen, dass keine Daten durch fehlerhafte Schreiboperationen verloren gehen.

### 5.1.2 Anwenderschnittstelle

#### Implementierung als Webseite

Die Implementierung als Webseite hat den Vorteil, dass auf vorhandene Software und Netzwerke zurückgegriffen werden kann. So verfügt vermutlich jeder Bürocomputer über einen installierten Webbrowser, eine weitergehende Installation ist nicht erforderlich. In einem vorhandenen Netzwerk wird nur ein Server benötigt, auf den alle Arbeitsstationen zugreifen können. Das ist ein Vorteil, wenn eine größere Personengruppe an der Datenerfassung beteiligt ist.

Die Darstellung im Webbrowser erfolgt durch die Auszeichnungssprache HTML, die ausgezeichnet dokumentiert und auch für Anfänger gut zu handhaben ist. Mit HTML ist es zum Beispiel im Vergleich zu Java, einfacher, graphische Oberflächen zu erstellen oder vorhandenen Code an veränderte Bedürfnisse anzupassen.

Als Programmiersprache kommt PHP in Frage, das ebenfalls gut dokumentiert ist und dessen Syntax C++ oder Java ähnlich ist. Einer der Hauptvorteile von PHP ist, dass es interpretiert wird. Es ist nicht nötig, den Quellcode vor Benutzung zu kompilieren. Alle Dateien sind reine Textdateien. Das macht Änderungen und Anpassungen verhältnismäßig einfach. Die minimale Programmierumgebung ist ein Texteditor. Für PHP gibt es eine große Reihe fertiger Bibliotheken, die zum Beispiel den Im- und Export von XML- oder Microsoft Excel Dateien ermöglichen.

Eine Alternative ist JSP (JavaServer Pages). JSP basiert auf der Programmiersprache Java, was die objektorientierte Programmierung erleichtert, jedoch auch die Installation eines speziellen Servers (Tomcat-Server) erfordert.

Einen Server zu verwenden hat den Vorteil, dass Sicherheitsaspekte bereits in der Serversoftware berücksichtigt sind und dass sie von Haus aus multiuserfähig ist. Außerdem kann Standard-Software auch von Personen administriert und gewartet werden, die keinen Bezug zur darin gespeicherten Literaturdatenbank haben.

Der Nachteil ist, dass dieser Server immer benötigt wird, auch wenn die Datenbank nur auf einem einzigen Rechner laufen soll. In diesem Fall muss der Server im Hintergrund auf diesem PC laufen.

Außerdem könnte besonders bei sehr großen Datenmengen und komplexen Anfragen die Skriptlaufzeit zu kurz sein. Dies lässt sich in der Konfiguration des Servers regeln. Allerdings sind die Schnittstellen zu den gängigen Datenbanken standardisiert, sodass man immer noch mit einer externen Anwendung auf den Datenbestand zugreifen könnte, sollte dieser Fall eintreten.

### Implementierung als Java-Applikation

Der Vorteil einer Java-Applikation liegt in der Objektorientierung der Programmiersprache Java. So sind zahlreiche Bibliotheken verfügbar, um die verschiedensten Datenstrukturen zu handhaben und auch Frameworks für die Modellierung von Ontologien, die ein dichtes semantisches Netz ermöglichen. Java ermöglicht aber auch den Zugriff auf relationale Datenbanken.

Die Handhabung graphischer Oberflächen ist in Java dagegen vergleichsweise kompliziert, da Java als prinzipiell plattformunabhängige Sprache nicht auf die graphischen Elemente des Betriebssystems zugreift und eine eigene Implementierung notwendig ist.

Eine Java-Applikation benötigt die Java-Laufzeitumgebung, genauer die Java Virtual Machine. Will man die Anwendung verändern, muss der veränderte Quellcode zu Bytecode kompiliert werden, dazu benötigt man den Java Compiler, der wiederum ein installiertes Java Development Kit voraussetzt. Alle diese Produkte sind kostenlos von Sun Microsystems erhältlich, sie müssen jedoch korrekt installiert werden.

### Implementierung als .NET-Applikation

.NET ist eine Programmierumgebung von Microsoft, um Programme für Microsoft Windows zu erstellen. Das bedeutet, dass eine damit programmierte Anwendung eine echte Windows-Anwendung ist, die im einfachsten Fall mit einem Klick auf ein Icon gestartet werden kann ohne dass umfangreiche Softwarepakete nachinstalliert werden müssen (vielleicht abgesehen vom .NET-Framework selbst, das jedoch üblicherweise Bestandteil von Windows Updates ist)

Die Verwendung von .NET ist vor allem in Verbindung mit Access interessant, weil beides vom gleichen Hersteller stammt.

### 5.1.3 Zusammenfassung

Es bieten sich bestimmte Technologien zur Kombination an. Webseiten funktionieren am besten mit relationalen Datenbanken wie MySQL, .NET und Access gehören natürlich zusammen und Java kann man mit allem verwenden, wenn man die richtigen Bibliotheken hat.

Die Vor- und Nachteile lassen sich tabellarisch darstellen:

| Architektur | Vorteile | Nachteile |
|---|---|---|
| Webseite<br>(PHP bzw. JSP und relationale Datenbank MySQL bzw. Oracle) | Code leicht änderbar<br>Läuft im Webbrowser<br>Multiuserfähig<br>Nutzt Features der Serversoftware<br>SQL | Laufzeit von Skripten<br>Erfordert Server |
| Java-Applikation<br>(& XML/Ontologie oder Datenbank) | objektorientiert<br>viele Bibliotheken verfügbar<br>Ontologie ermöglicht dichtes semantisches Netz | JRE (keine exe-Datei)<br>dateibasierte Datenbank und Oberfläche müssen komplett implementiert werden |
| .NET-Applikation<br>(& MSAccess) | hohe Verfügbarkeit von MS-Access<br>„Ein-Klick-Applikation“ | Windows-Applikation<br>Proprietäres Format von Microsoft |

Tabelle 5: Vor- und Nachteile verschiedener Architekturen

Die Entscheidung fiel zugunsten einer PHP/MySQL-Applikation, da MySQL den SQL-Standard unterstützt und man mit PHP sehr einfach darauf zugreifen kann. Ein weiteres starkes Argument ist, dass MySQL eine seit Jahren bewährte Software ist, die mir eine eigene Implementierung sicherer Dateioperationen abnimmt. Außerdem stellt es eine komfortable und flexible Umgebung dar, deren Serversoftware kostenlos verfügbar und einfach zu installieren ist.

## 5.2 Modularer Aufbau

In PHP geschriebener Code wird leicht unordentlich und schwer wartbar, wenn Programmcode mit Code zur Darstellung vermischt wird. Daher erfolgte die Implementierung unter Ausnutzung des MVC-Patterns. Das bedeutet, dass der Code zur Darstellung der Oberfläche, zur Datenhaltung und zur Kontrolle voneinander getrennt ist. Das erfordert etwas mehr Programmieraufwand, da an mehreren Stellen Daten für die Verwendung in einer anderen Prozedur vorbereitet werden müssen, man gewinnt jedoch viel Flexibilität, da alle Komponenten unabhängig voneinander gewartet und verändert werden können.

### Model

Alle Anfragen an die Datenbank finden in eigenen Klassen bzw. Skripten statt, getrennt von der Darstellung der Oberfläche. Dadurch müssen zum Beispiel bei Wechsel der Datenbank oder der Datenbankstruktur nur diese Handler-Klassen umgeschrieben werden.

### View

Um die Darstellung konsequent von der Logik zu trennen, wurde eine Template-Engine verwendet. Die Template-Engine vlib hat sich dabei als brauchbar erwiesen.

Alle Bildschirmseiten existieren als HTML-Templates und werden von der Template-Engine mit Inhalt gefüllt. Dadurch kann die Oberfläche unabhängig vom Rest des Programms verändert werden und man muss nicht unbedingt PHP beherrschen, um die Oberfläche zu verändern.

Bei den HTML-Templates wurde darauf geachtet, Formatierungen vor allem über Stylesheets vorzunehmen, das macht die Oberfläche zusätzlich flexibler.

### Controller

Die Kontrollskripte übernehmen Werte aus den Formularfeldern der HTML-Seite und übergeben sie an Datenbank-Funktionen oder bekommen Daten von Datenbank-Funktionen und übergeben sie an die Template-Engine zur Darstellung.

## 5.3 Anwenderschnittstelle

Wie in den Anforderungen für die graphische Oberfläche (siehe Abschnitt 4.1.1) beschrieben, ist es wichtig, die Oberfläche übersichtlich zu gestalten.

Ein Menü in der linken Spalte ermöglicht die Navigation durch die verschiedenen Unterseiten. Der Rest des Bildschirms stellt die momentane Aufgabe dar. In der folgenden Abbildung (Abbildung 8) sieht man beispielsweise die Maske für die Eingabe eines neuen Dokumentes. Die gleiche Maske wird mit vorbelegten Daten zum Bearbeiten von Dokumenten verwendet. Dabei gibt die Seitenüberschrift stets an, auf welcher Seite man sich befindet und was man auf der Seite tun kann.

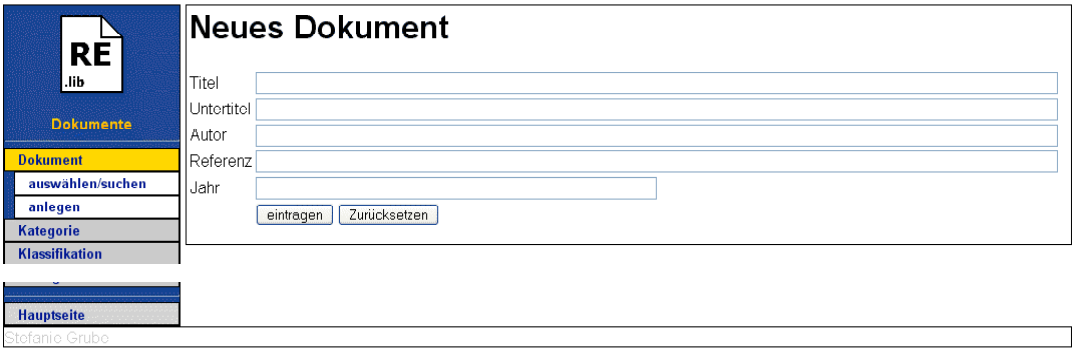


Abbildung 8: Aufteilung der Oberfläche

Der Klassifikationsbaum wurde wie in Abbildung 9 ersichtlich in mehrere Teile geteilt, um auf diese Weise übersichtlicher dargestellt werden zu können. Die ersten beiden Ebenen des Baumes bilden ein zweistufiges Menü im oberen Seitenbereich. Das bedeutet, wenn man auf eins der Schlagworte in der oberen Reihe klickt, ändert sich die entsprechende untere Reihe.

Auf diese Weise wird darunter nur noch ein Teilbaum dargestellt, die Seite wirkt so insgesamt übersichtlicher als wenn der gesamte Baum auf einer Seite dargestellt würde.

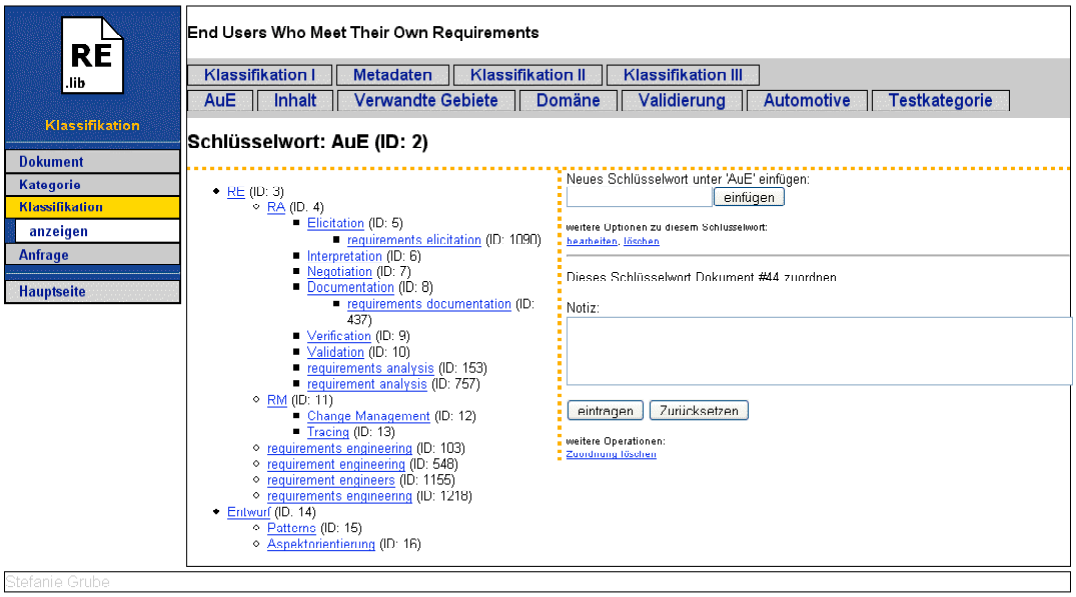


Abbildung 9: Darstellung des Kategorienbaumes

## 5.4 Im- und Export

### Import

Um sich Tipparbeit zu erleichtern, kann man Metadaten zum Teil auch vom Herausgeber einer Publikation erhalten. Das IEEE gibt zum Beispiel für alle seine Konferenzen Tagungsbände heraus, deren Metadaten für registrierte Benutzer online verfügbar sind.

Diese Citations können in verschiedenen Formaten direkt in Referenzverwaltungssysteme importiert werden und so als Literaturreferenz verwendet werden.

Das in Abschnitt 2.2.1 beschriebene Programm EndNote verfügt über die Möglichkeit, solche Referenzen als XML zu exportieren (siehe Codebeispiel 6). Auf diese Weise ist es möglich, viele Dokumente auf einmal zu importieren.

```
<XML>
<RECORDS>
<RECORD>
<REFERENCE_TYPE>3</REFERENCE_TYPE>
<REFNUM>124</REFNUM>
<YEAR>2006</YEAR>
<NOTES>TY  - CONF</NOTES>
<SECONDARY_TITLE>
              Requirements Engineering Conference, 2006. RE 2006. 14th IEEE
              International</SECONDARY_TITLE>
<TITLE>An Empirical Quality Assessment of Automotive Use Cases</TITLE>
<ISBN>1090-705X</ISBN>
<PAGES>86-95</PAGES>
<KEYWORDS>
<KEYWORD>automobile industry</KEYWORD>
<KEYWORD>production engineering computing</KEYWORD>
<KEYWORD>quality control</KEYWORD>
<KEYWORD>Volvo Car Corporation</KEYWORD>
<KEYWORD>automotive use cases</KEYWORD>
<KEYWORD>empirical quality assessment</KEYWORD>
<KEYWORD>evaluation criteria</KEYWORD>
</KEYWORDS>
<ALTERNATE_TITLE>
              Requirements Engineering Conference, 2006. RE 2006. 14th IEEE
              International</ALTERNATE_TITLE>
<ABSTRACT>
              As functionality in vehicles grows more complex and development becomes
              distributed over several geographical sites, elicitation and visualization of
              requirements become more critical. This paper […]</ABSTRACT>
</RECORD>
</RECORDS>
</XML>
```

Codebeispiel 6: XML-Export von EndNote

Eine automatische Verarbeitung setzt jedoch voraus, dass das Schema der XML-Datei genau eingehalten wird. Daher muss für jede Importquelle ein angepasstes Skript geschrieben werden.

### Export

Es ist vorgesehen, dass die Literaturdatenbank Anfrageergebnisse vorbereitet ausgibt, und diese dann über die Zwischenablage in ein Tabellenkalkulationsprogramm kopiert werden.

Der Hauptgrund ist, dass Tabellenkalkulationsprogramme, wie zum Beispiel in Microsoft Excel, von Haus aus eine Vielzahl von Möglichkeiten zur Formatierung und Diagrammerstellung bieten. Eine eigene Implementierung wäre im Funktionsumfang nicht vergleichbar und somit redundant.

Es ist grundsätzlich möglich, einen direkten Export in eine Excel-Datei zu implementieren, jedoch wurde davon abgesehen, weil die Benutzung der Zwischenablage im Rahmen dieser Arbeit ausreichend erscheint.

Daten können grundsätzlich auch über die Exportfunktion von phpMyAdmin, einer PHP-Applikation zur Verwaltung von MySQL-Datenbanken, vorgenommen werden. Hier ist es möglich, Daten tabellenweise in SQL, LateX, Word oder Excel-Format zu exportieren.

# 6 Fazit

In dieser Arbeit wurden die Grundlagen einer Literaturdatenbank mit dem Schwerpunkt des Requirements Engineerings entwickelt.

Eine Betrachtung vorhandener Systeme ergab, dass keines davon allein uneingeschränkt die Anforderungen der Aufgabe erfüllt. Die meisten dieser Systeme sind nicht in der Lage, ein Mengengerüst zu erstellen, also die Frage zu beantworten, wie viele Dokumente es zu einem gegebenen Thema einschließlich Unterthemen überhaupt gibt.

Die einzelnen Dokumente lassen sich, repräsentiert durch Schlagwörter, in Kategorien einordnen, die ihrerseits eine baumartige Struktur bilden. Das heißt, ausgehend vom allgemeinen Thema Requirements Engineering lässt sich das Themengebiet breit gefächert in Unterkategorien anordnen.

Eine Analyse des Worksflows ergab zwei zentrale Bereiche: Zum einen die Erfassung von Rohdaten, zum anderen die Weiterverarbeitung und Präsentation der Ergebnisse.

Die Erfassung von Dokumenten und Klassifikationen wird dadurch bestimmt, dass viele Dokumente in kurzer Zeit erfasst werden sollen. Die kognitive Last muss deshalb gering gehalten werden, das erfordert eine intuitiv bedienbare Oberfläche, deren Darstellung die zu erfassende Struktur widerspiegelt. Dies erfolgt durch eine kompakte Darstellung des Begriffsbaumes.

Für die Ausgabe von Ergebnisdaten wurden Standardanfragen entwickelt, die es ermöglichen, aussagekräftige Ergebnisse aus den Rohdaten zu erstellen. Zum einen sind das Anfragen, die es ermöglichen, einen bestimmten Datensatz wieder zu finden, um ihn zum Beispiel als Referenz zu benutzen, zum anderen sind es Anfragen, wie viele Dokumente den vorgegebenen Kriterien entsprechen. Dazu wurden Methoden beschrieben, um Daten aus der Gesamtdatenbank zu erhalten, wie zum Beispiel die Aggregation, die Unterkategorien einer gegebenen Kategorie zusammenfasst. Damit ist es möglich, den Datenbestand auf einer gröberen Ebene zu betrachten als er ursprünglich erfasst worden ist.

Um den Kategorienbaum flexibel an die jeweiligen Gegebenheiten anpassen zu können, wurden eine Reihe von Methoden beschrieben, um ihn bearbeiten zu können, ohne die Konsistenz der Datenbank zu verlieren.

Das ist zunächst eine Einfügeoperation, um neue Kategorien hinzuzufügen sowie Methoden, um bestehende Kategorien zu verändern. Dazu gehören einfache Operationen, die das Umbenennen einer Kategorie ermöglichen, eine Verschiebeoperation, um eine Kategorie oder einen Teilbaum an eine andere Stelle im Baum verschieben zu können und eine Löschoperation, die vor dem Löschen alle bestehenden Verknüpfungen und Unterkategorien auf eine andere Kategorie im Baum überträgt.

Insgesamt ermöglicht es diese Literaturdatenbank, Dokumente aus dem Bereich des Requirements Engineering zu erfassen, ihre Relevanz zu bewerten und die in den Kontext des Themengebietes, repräsentiert durch ein baumartiges Modell, einzuordnen.

Es ist so möglich, Rückschlüsse auf den Stand der Forschung zu ziehen oder auch nach längerer Zeit ein Dokument zu einem Thema wieder zu finden ohne alle Dokumente noch einmal lesen zu müssen. Die Literaturdatenbank spart dem Softwareingenieur also Zeit.

# 7 Ausblick

Die Literaturdatenbank ermöglicht bisher nur grundlegende Funktionen der Ein- und Ausgabe. Was noch fehlt, sind Methoden zum Im- und Export, um eine Kommunikation mit anderen Anwendungen zu ermöglichen. So wäre zum Beispiel ein Export in Endnote eine gute Möglichkeit, die in einer Anfrage erhaltenen Datensätze direkt in eine Referenzdatenbank umzusetzen, aus der dann in anderen Dokumenten Bezug genommen werden kann.

Das gleiche gilt für einen Export in ein Tabellenkalkulationsprogramm wie Microsoft Excel. Momentan müssen Anfrageergebnisse über die Zwischenablage kopiert werden, um sie weiterverarbeiten zu können. Ein direkter Export würde das erleichtern und zum Teil auch beschleunigen.

Ebenso erweiterungsfähig sind die Anfragen. Derzeit beschränkt sich das Spektrum auf einige wenige Standardanfragen, es sind jedoch noch andere Anfragetypen denkbar, die sich aus der jeweiligen Fragestellung ergeben. Eine Grundlage ist schon durch die Baumoperationen gegeben, aus denen sich neue Anfragen zusammensetzen lassen.

Vor dem professionellen Einsatz beispielsweise im Internet muss die allgemeine Sicherheit der Anwendung dafür optimiert werden, es wurde beispielsweise kein Zugriffsschutz implementiert, was bedeutet, dass alle Daten für jeden einsichtig und änderbar sind.

Zusammen mit einem Zugriffsschutz könnten Benutzerprofile realisiert werden, damit bei mehreren Benutzern nachvollzogen werden kann, wer wann was geändert hat.

# Anhang:

## *Anhang A: Glossar*

| | |
|---|---|
| AuE | Abkürzung für „Anforderung und Entwurf“ als Teilgebiet des Software Engineering |
| Conference Proceedings | Tagungsband, Sammlung von Forschungspapieren, die auf einer Konferenz vorgestellt wurden |
| Ontologie | |
| RE | Requirements Engineering |
| Research Paper | Forschungspapier. Kurzer Aufsatz, der Forschungsergebnisse dokumentiert |
| SE | Abkürzung für „Software Engineering“ als Teilgebiet der Informatik oder für das Fachgebiet für Software Engineering an der Leibniz Universität Hannover |
| | |

## *Anhang B: Baumoperationen*

| Funktion | Rückgabe | Macht was? |
|---|---|---|
| ***Existenzprüfung*** | | |
| catExists(cat) | boolean | Prüft, ob eine Kategorie existiert |
| catTitleExists(name) | boolean | Prüft, ob eine Kategorie dieses Namens existiert |
| isParent(p, c) | boolean | Prüft, ob Kat. p Elternkategorie von Kat. c ist |
| isChild(c, p) | boolean | Prüft, ob Kat. c Kindkategorie von Kat. p ist. |
| hasChildren(cat) | boolean | Prüft, ob die Kategorie cat Kindkategorien hat |
| isSibling(cat1, cat2) | boolean | Prüft, ob Kat. cat1 Geschwisterkategorie von Kat. cat2 ist (ob sie die gleiche Oberkategorie haben) |
| hasSibling(cat) | boolean | Prüft, ob Kat. cat Geschwisterkategorien hat |
| isAncestor(p, c) | boolean | Prüft, ob Kat. p Vorgängerkategorie von Kat. c ist. |
| isAncestorOrSelf(p,c) | boolean | Prüft, ob Kat. p Vorgängerkategorie von Kat. c oder gleich Kat. c ist (vgl. Codebeispiel 2) |
| isDescendant(c, p) | boolean | Prüft, ob Kat. c Nachfolgekategorie von Kat. p ist. |
| isDescendantOrSelf(c,p) | boolean | Prüft, ob Kat. c Nachfolgekategorie von Kat. p oder gleich Kat. p ist |
| ***Leseoperationen*** | | |
| getParent(cat) | int | Gibt die ID der Elternkategorie aus |
| getChildren(cat) | int[] | Gibt die IDs der Kindkategorien von Kat. cat aus. |
| getFirstChild(cat) | int | Gibt die ID der ersten Kindkategorie von Kat. cat aus |
| getSiblings(cat) | int[] | Gibt die IDs der Geschwisterkategorien von Kat.cat aus |
| getAncestors(cat) | int[] | Gibt die IDs der Vorgängerkategorien von Kat. cat aus |
| getDescendantsList(cat) | int[] | Gibt die IDs der Nachfolgekategorien von Kat. cat als eindimensionales Array aus |
| getDescendantsTree(cat) | array[] | Gibt die Nachfolgekategorien von Kat. cat als Baum aus |
| ***Schreiboperationen*** | | |
| insertCategory(title, p) | int | Fügt eine Kategorie mit dem Titel „titel“ und der Elternkategorie p ein, gibt die ID der neu eingefügten Kategorie zurück |
| updateCategory(cat, title, p) | void | Ändert Titel und Elternkategorie der Kat. p |
| setParent(cat, p) | void | Ändert die Elternkategorie der Kat. cat in Kat. p |
| setTitle(cat, title) | void | Ändert den Namen der Kat. cat zu „title“ |
| moveCategory(cat, new_p) | void | Verschiebt die Kat. cat unter die neue Elternkategorie new_p (vgl. Codebeispiel 3) |
| deleteCategory(cat) | void | Löscht die Kat. cat. (vgl. Codebeispiel 4) |
| deleteCatFromDB(cat) | void | Löscht die Kat. cat aus der Datenbank |

# Abbildungsverzeichnis



# Tabellenverzeichnis

# Literaturverzeichnis


[1] 15th IEEE International Requirements Engineering Conference,
http://www.re07.org/home/
zuletzt gesehen: 10.01.2008

[2] Referenzmodell AuE: K. Schneider, Folien zur Vorlesung Anforderung und Entwurf an der Leibniz Universität Hannover, Sommersemester 2007

[3] IEEE: Institute of Electrical and Electronics Engineers. http://www.ieee.org.
zuletzt gesehen: 10.01.2008

[4] EndNote: http://www.endnote.com/
zuletzt gesehen: 10.01.2008

[5] Protégé: http://protege.stanford.edu/
zuletzt gesehen: 10.01.2008

[6] Google Scholar Hilfe: http://scholar.google.de/intl/de/scholar/help.html
zuletzt gesehen: 10.01.2008

[7] SE-Map: http://www.se.uni-hannover.de/SEMap/
zuletzt gesehen: 10.01.2008

[8] Dublin Core: http://dublincore.org/documents/usageguide/
zuletzt gesehen: 10.01.2008

[9] XPath: http://www.w3.org/TR/1999/REC-xpath-19991116
zuletzt gesehen: 10.01.2008