Implementierung eines Ajax basierten DHTML File-System-Trees für die Online-Enzyklopädie Austria-Forum

Masterarbeit

an der

Technischen Universität Graz

vorgelegt von

Siegfried Oberhauser

Institut für Informationssysteme und Computer Medien IICM
Technische Universität Graz
A-8010 Graz
Österreich

Betreuer: Assoc.Prof. Dipl.-Ing. Dr.techn. Denis Helic

Kurzfassung

Wiki Systeme sind ein wesentlicher und beliebter Bestandteil im Web 2.0. Hauptgrund dafür ist das einfache Konzept von Wikis, welches jeden Benutzer erlauben soll ohne Einschränkung Beiträge zu verfassen, zu ändern und natürlich auch zu lesen. Der Erfolg von Wikipedia unterstreicht das Funktionieren dieses Prinzips eindrucksvoll. Durch die Unkompliziertheit kann sich jedoch bald eine beträchtliche Anzahl an Artikeln ansammeln, die bei herkömmlichen Wikis auf einer Ebene existieren. Dies führt zu einer *spaghetti-ähnlichen* Struktur der Artikel, in der die Benutzer die Orientierung und den Überblick leicht verlieren können. Diesem Phänomen wird durch sogenannte *structured* Wikis entgegengewirkt. Das System des Austria-Forums implementiert eine solche Methode, die als *subpaging* bekannt ist. Dieses System soll hier vorgestellt werden und auch aus einem anderen Blickwinkel betrachtet werden. Weiters wurde ein Tool implementiert, welches die daraus resultierende Artikelstruktur visualisiert. Dieses Werkzeug soll eine Erleichterung für die Navigation durch die Beiträge und eine Unterstützung der Verwaltung von Dateien des Systems darstellen.

Abstract

Wiki systems are an essential and popular part of Web 2.0. The main reason for this is the simple concept of wikis, which allows each user to write, modify, or read articles without any restrictions. The success of Wikipedia underscores the functioning of this principle in an impressive way. Due to this simplicity huge numbers of articles can be created quickly. However, state-of-the-art wikis possess a flat structure and all pages are stored at one and the same level. This typically leads in a *spaghetti-like* structure of articles, where users loose their orientation and overview very easily. Therefore, so-called *structured* wikis have been developed to counteract this phenomenon. Austria-Forum wiki implements such a method, which is known as *subpaging*. In this work we will introduce that system. Furthermore, we implemented a tool to visualize the resulting article structure. That tool supports an easier navigation through the articles and provides for an easier maintenance and administration.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Danksagung

An dieser Stelle möchte ich mich bei Assoc.Prof. Dipl.-Ing. Dr.techn. Denis Helic für die Betreuung und Unterstützung während dieser Arbeit bedanken.

Weiters bedanke ich mich herzlich bei meiner Frau, meiner Tochter und meinem Sohn, die mir in schwierigen Situationen während meines Studiums immer wieder neue Kraft gaben.

Inhaltsverzeichnis

1. Einleitung

In Zeiten des Web 2.0, welches Benutzern erlauben soll, Inhalte selbst zu gestalten bzw. zu bearbeiten, gehören Wiki Systeme zu einem wesentlichen Bestandteil der Web-basierten Informationstechnologie. Sie bieten die Grundlage, ein riesengroßes Potential an Wissen auf einen Punkt zu konzentrieren. Dies ist ein wesentlicher Grund dafür, dass seit 1996 bestehende, nicht ganz so erfolgreiche Online-Lexikon AEIOU, zu großen Teilen vom Austria-Forum zu übernehmen und so diesem Informationsportal über Österreich wieder neues Leben einzuhauchen. Dem Benutzer wird einerseits die Anwendung erleichtert und zugleich die Möglichkeit gegeben, aktiv bei der Gestaltung des Lexikons mitzuarbeiten. Das neue System basiert auf JSPWiki, ein auf Java und JavaServer Pages aufgebautes Wiki System.

1.1 Motivation – Structured Wiki

Im Jahre 1995 entwickelte Ward Cunningham ein System, welches jedem Benutzer ohne Einschränkungen und auf einfachste Weise gewährte, Beiträge zu lesen, selbst zu verfassen oder vorhandene Beiträge zu ändern. Diesem Prinzip wurden sofort anarchistische Zustände voraus gesagt. Cunningham meinte allerdings, dass wenn genug User beteiligt sind, sich alles selbst reguliert. Spätestens seit der Einführung von Wikipedia bestätigte sich seine Meinung.

Aufgrund der Einfachheit dieses Konzepts ist es möglich, schnell viele Beiträge zu kreieren, jedoch geht die Übersicht genauso schnell verloren. Vor allem Wikis der ersten Stunde legten alle Seiten auf einer Ebene ab. Die Wikis verwilderten sprichwörtlich. Wegen der fehlenden strukturellen Beziehungen zwischen Beiträgen untereinander und dadurch auch der Links, wurde es immer schwerer sich im System zurechtzufinden bzw. durch das System zu navigieren. Eine hierarchische Suche konnte natürlich auch

nicht geboten werden. Sogar die Namensvergabe wird in flachen Dateiensystemen bei steigender Dateimenge immer komplexer.

Somit war der Weg frei für die sogenannten *structured* Wikis. Solche Systeme bieten Features an, um den Inhalt eines Wikis zu strukturieren. Eine dieser Möglichkeiten ist *subpaging*. Hier können in Kategorien Beiträge oder neue Unterkategorien mit Beiträgen und so weiter erstellt werden. Dadurch ergibt sich ein hierarchisches System indem die Beziehungen von Beiträgen untereinander sofort ersichtlich sind. Damit wird auch die Navigation durch das System intuitiver. Die gewünschten Informationen sind leichter zu finden, da man sich Ebene für Ebene bis zur gewünschten Information durch die Hierarchie hangeln kann (Hierarchische Suche siehe Kapitel 1.4). Der Einsatz solcher Wikis bewirkt eine effizientere Nutzung und war auch Motivation, das System des Austria-Forums auf diese Weise zu konzipieren.

1.1.1 Ziele

In dieser Arbeit soll die Hierarchisierung der Dateienstruktur und deren Entwicklung zusätzlich aus Sicht der Betriebssysteme betrachtet werden, um Vergleiche und Erkenntnisse auch daraus zu erzielen. Weiters soll die Umstellung auf dieses System und daraus resultierende strukturgebende Features im Austria-Forum erläutert werden.

Um die Möglichkeiten eines hierarchischen Systems noch weiter zu unterstützen, wird ein *File-System-Tree* implementiert. Dieser soll die Dateienstruktur des Austria-Forums über ein Webinterface visualisieren und nicht nur ein Werkzeug zur Navigation sein, sondern auch Möglichkeiten zur Dateienverwaltung bieten. Vorher werden noch andere Möglichkeiten zur Darstellung von hierarchischen Strukturen vorgestellt um eine geeignete Visualisierungsmethode für dieses Tool herauszufiltern.

1.2 Hierarchien

Das Wort Hierarchie kommt aus dem Griechischen (*hierarchia*) und ist eine Komposition der Ausdrücke *hierós (heilig) und arché (*Anfang, Führung, Herrschaft) und bezeichnet ein System von Elementen die einander über- oder untergeordnet sind (Hierarchie, 2011).

Hierarchien begegnen einem überall im täglichen Leben. Gesetzestexte, Produktverzeichnisse in Katalogen oder Inhaltsverzeichnisse in Büchern sind hierarchisch geordnet. Auch bei Menschen und Tieren gibt es Rangordnungen, welche als Hierarchie gedeutet werden können. Befehlsstrukturen im Militär oder in großen Unternehmungen sind Beispiele für größere Hierarchien. In allen Fällen sind sie durch eine klare Strukturierung gekennzeichnet, welche nicht nur zum Zweck der Übersicht dient sondern auch eine Bedeutung hat (Klempt, 2008).

Hierarchien sind ein sehr guter Mechanismus um große Informationsräume, in welchen man in der Informatik oft operiert, zu organisieren (Egger, 2004).

Grundsätzlich unterscheidet man zwischen:

- *Natürlicher Hierarchie*
  (z.B. Stammbaum, Wolfsrudel usw.) und
- *Künstlicher Hierarchie*
  (z.B. Befehlsstruktur beim Militär oder in einer Firma)

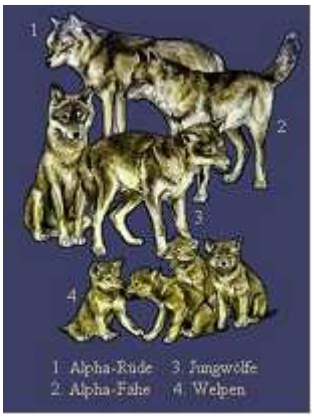

Abbildung 1: Natürliche Hierarchie – Aufbau eines Wolfrudels (Klempt, 2008)

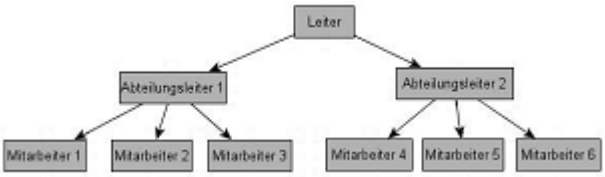

Abbildung 2: Künstliche Hierarchie – Einfache Firmenhierarchie (Klempt, 2008)

Das menschliche Gehirn benützt auch hierarchische Strukturen zum Speichern von Informationen und wird häufig mit dem Aufbau eines Baumes veranschaulicht. Grundsätzlich lassen sich alle hierarchischen Daten als Baumstrukturen abbilden (Winkler, 2006).

In der Graphentheorie wird ein hierarchischer Baum folgend definiert (Klempt, 2008):

Graph (gerichteter Graph): G = (V,E) heißt gerichteter Graph genau dann, wenn gilt:

- *V ist eine nichtleere Menge – die Menge der Knoten von G.*
- *E ist eine Menge, deren Elemente entweder Paare von Elementen aus V oder Einzelelemente aus V sind. E bezeichnet die Menge der Kanten und Schlingen von G.*
  - *Ein Paar (u,v) ϵ E (mit u ϵ V und v ϵ V) heißt Kante vom Knoten u zum Knoten v.*
  - *Ein Element u ϵ E (wobei gleichzeitig u ϵ V gilt) heißt Schlinge am Knoten u.*

Baum: Ein gerichteter Graph G = (V,E) heißt Baum genau dann, wenn gilt:

- *Es gibt genau einen Knoten $w \in V$ ohne hineinlaufende Kante. w heißt Wurzelknoten (Wurzel, engl. root[node]) des Baumes.*
- *Es gibt für jeden Knoten $v \in V$ mit $v \neq w$ genau einen Weg von w nach v.*

Eine Hierarchie bzw. Baum ist demnach eine Sammlung von Objekten, in welcher jedes Objekt in Bezug zu Unterobjekten stehen kann, also eine Ordnung in welcher alle Objekte Nachfahren und/oder Vaterobjekt anderer Objekte sind. An der Spitze steht das Wurzelelement, welches selbst kein Nachfahre ist. Jedes Objekt darf außerdem nur ein Vaterobjekt haben, somit schließt man implizit auch alle Zirkelbezüge aus. Die Beziehung zwischen Objekten wird als Kante bezeichnet (Egger, 2004).

1.3 Datenstruktur

Datenstrukturen sind ein elementarer Bestandteil der Informatik, die zur Speicherung und effektiven Verwaltung der für eine Anwendung benötigten Daten dienen. Deshalb möchte ich auch hier einleitend ein paar Worte darüber verlieren, obwohl es nicht direkt mit der Dateienverwaltung vergleichbar ist. Je nach Art der Anwendung werden flache bzw. lineare (Listen) oder hierarchische (Bäume) Strukturen eingesetzt.

Betrachtet man das häufig in Wikis verwendete flache Verzeichnissystem genauer, kann es im Prinzip mit einer Baumstruktur (*tree*), mit der maximalen Tiefe 1 (*depth*) gegenüber gestellt werden. Diese Struktur besitzt also eine Wurzel (*root*) von welcher aus alle vorhandenen Knoten (*nodes*) über Kanten (*edges*) mit der Wurzel verbunden sind.

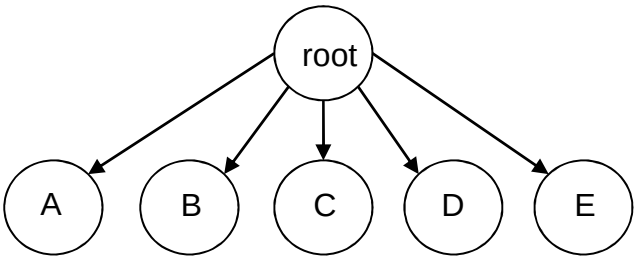

Abbildung 3: Baumstruktur mit Tiefe 1

In Wikis ist die Wurzel das Verzeichnis bzw. der Ordner, welcher alle Dateien ohne weitere Unterverzeichnisse beinhaltet.
Diese Struktur gleicht aufgrund des Umstandes, dass alle Knoten dieselbe Wurzel haben bzw. dass alle Dateien ohne weitere Unterverzeichnisse bzw. Unterordner in einem Verzeichnis enthalten sind, eher einer *linearen Datenstruktur* z.B. einer Liste.

Das daraus adaptierte hierarchische Verzeichnis- und Kategorisierungssystem für das Austria-Forum hingegen ist mit einer *hierarchischen Datenstruktur*, also einer Baumstruktur beliebiger Tiefe vergleichbar. Das heißt, dass nicht nur die Wurzel Elternteil (*parent*) aller vorhandenen Knoten ist, sondern jeder Knoten auch sein eigenes Kind (*child*) bzw. Kinder haben kann.

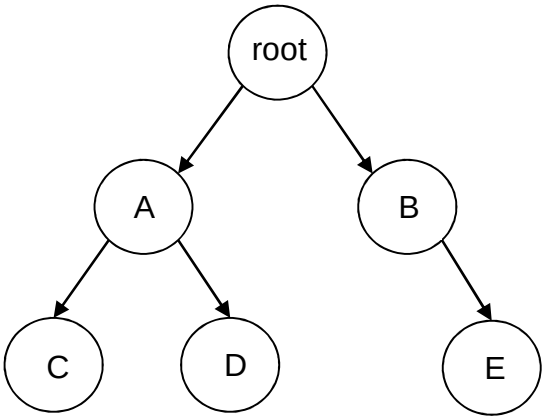

Abbildung 4: Baumstruktur mit beliebiger Tiefe

Im Austria-Forum ist die Wurzel also das Verzeichnis, das alle Dateien beinhaltet, welche zusätzlich auch Inhalt weiterer Unterverzeichnisse sein können.

Betrachten wir diese zwei grundsätzlichen Modelle und grundlegende Typen daraus kurz aus Sicht der in der Informatik ständig benötigten Datenstrukturen (abstrakte Datentypen).

1.3.1 Lineare Datenstruktur

In einer linearen Datenstruktur stehen die benachbarten Elemente im Speicher entweder direkt nebeneinander oder die Elemente sind miteinander verkettet.

1.3.1.1 Arrays

Arrays sind einfache Datenstrukturen. Der Zugriff auf einzelne Elemente erfolgt über den Index (A[i]) und ist daher recht einfach. Im Grund ist der Index nichts anderes als die Differenz zwischen Startadresse des Arrays und der Startadresse des jeweiligen Elementes. Ein Array hat eine feste Größe.

1.3.1.2 Listen

Listen werden zur dynamischen Speicherung von beliebig vielen Elementen verwendet. Jedes Element ist mit dem nächsten Element verkettet (z.B. Pointer auf nächstes Element).

- *Stapel (Stack)*
  In einem Stapel können die gespeicherten Elemente in umgekehrter Reihenfolge ihres Ablegens ausgelesen werden, d. h. dass nur am Ende des Stapels Elemente eingefügt bzw. entfernt werden können. Dieses Prinzip ist unter der Bezeichnung *LIFO* (last in-first out) bekannt.

- *Warteschlange (Queue)*
  In einer Warteschlange werden Elemente in der Reihenfolge ihres Ablegens auch wieder ausgelesen. Das bedeutet das Elemente am Ende der Schlange eingefügt und am Anfang entfernt werden können. Dieses Prinzip wird *FIFO* (first in-first out) genannt.

1.3.2 Hierarchische Datenstruktur

Bei hierarchischen Datenstrukturen werden die Elemente in mehreren Ebenen abgelegt. Der wichtigste Vertreter dieser Kategorie ist der Baum in all seinen Variationen. Bäume werden oft benützt, um ein strukturiertes Verwalten von Daten zu ermöglichen und somit gewisse Operationen wie z.B. das Suchen zu erleichtern.

1.3.2.1 Baum (Tree)

Unter Baum versteht man in der Informatik eine bestimmte Art dynamischer Datenstrukturen, die dazu verwendet werden, Daten hierarchisch zu speichern. Dynamisch deshalb, weil die Anzahl der Elemente nicht von vornherein begrenzt ist, sondern jedes Mal, wenn ein neues Element gespeichert werden soll, Speicherplatz dafür reserviert wird und deshalb die Anzahl der Elemente, die gespeichert werden können nicht von der Datenstruktur, sondern der Größe des zur Verfügung stehenden Speichers abhängig ist (Koch, 2001).

Zur effektiven Verarbeitung der in der Struktur befindlichen Daten gibt es verschiedenste Ausführungen von Bäumen, wie z.B. balancierte Bäume (B-Bäume), welche bei der Implementation von vielen Dateisystemen zur Speicherverwaltung im Hintergrund oft verwendet werden.

1.3.3 Lineare Datenstruktur vs. Hierarchische Datenstruktur

Generell muss gesagt werden, dass für verschiedene Probleme verschiedene Datenstrukturen von Vorteil sind. So wäre es sinnlos für eine einfa-

che Druckerwarteschlange, wo Aufträge die älter sind zuerst abgearbeitet werden (FIFO), etwas anderes als eine Queue zu verwenden.
Rein pauschal kann also nicht gesagt werden, welche Struktur die beste ist. Generell betrachtet, könnte man folgende Punkte aufzählen.

Vorteile von Linearen Datenstrukturen:

- sehr schnelles Einfügen und
- Entfernen von Objekten möglich (konstante Anzahl von Schritten)
- einfach zu implementieren

Nachteile:

- Suchen von Elementen kann zeitaufwändig sein,
  es sei denn, man sortiert die Liste, aber dann geht Vorteil des schnellen Einfügens verloren (lineare Anzahl von Schritten)

Vorteile von Bäumen:

- schnelles Suchen und Finden von Objekten möglich
  (sehr wichtig, da dies in unserem Fall zu einer besseren Navigation durch die Daten des Systems führt)
- Einfügen und Entfernen von Objekten recht schnell möglich (logarithmische Anzahl von Schritten)

Nachteile:

- etwas komplizierter zu implementieren
  (Vor-&Nachteile: Lineare Listen, Bäume, 2005)

Ein weiterer wesentlicher und für unseren Fall wichtiger Vorteil von Bäumen ist die gute visuelle Darstellung von hierarchischen Strukturen wie zum Beispiel (Pareigis, Kahlbrandt, & Stephan, 2009):

- Organigramm eines Unternehmens (Aufteilung in Abteilungen, Gruppen und deren Mitarbeiter)
- Gliederung eines Buches (Kapitel, Unterkapitel und Abschnitte)
- Aufteilung von Österreich in Bundesländer, Bezirke und Gemeinden
- Stammbaum eines Menschen (Eltern, Großeltern usw.)

- Darstellung der Dateienstruktur eines Systems (Ordner, Unterordner und Dateien wie z. B.: Windows Explorer)
- Auch mathematische Ausdrücke können als Baum visuell dargestellt werden wie z. B.: (3 * 12 + 4) * (5 / 7)

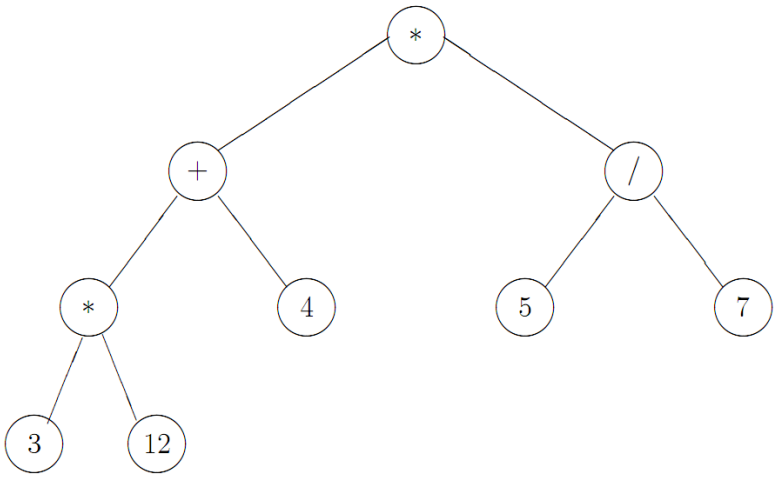

Abbildung 5: Vollständig geklammerter Ausdruck als Baum (Pareigis, Kahlbrandt, & Stephan, 2009)

1.4 Hierarchische Suche

Hierarchische Suche bedeutet, dass man ausgehend von einem globalen Problem eine Suchanfrage stufenweise verfeinert, um so schlussendlich die elementare Information zu erhalten (ENZYKLO online Enzyklopädie, 2011).

Hierarchische Suchsysteme ermöglichen es einem, über die Auswahl von Haupt- und Unterkategorien, welche hierarchisch geordnet sind, zu einem gewünschten Themengebiet die gewünschte Antwort zu finden. Eine solche Hierarchie von Kategorien hat folgenden Vorteil: Will man sich z.B. in ein Themengebiet neu einarbeiten, fallen einem zuerst nicht gleich die richtigen Stichwörter ein. Über ein Kategoriensystem kann man jedoch schnell erste grobe Informationen finden und so schrittweise an die gewünschte Information gelangen. Es gibt bislang noch nicht die Möglichkeit, Webseiten automatisch korrekt zu kategorisieren. Solche Systeme werden bisher nur manuell erstellt - z.B. durch eine Redaktion - weshalb solche

Systeme bei weitem nicht so umfangreich sind wie die Datenbanken einer Suchmaschine (Werle, 2011).

1.5 Gliederung der Arbeit

Im 2. Kapitel soll als eine grundlegende Form des Datenmanagements das Dateisystem und daraus vor allem das Verzeichnissystem betrachtet werden. Es werden gebräuchliche Formen erläutert um die Vorteile von hierarchischen gegenüber flachen Systemen zu erfahren und damit die Einführung eines hierarchischen Dateien-Managements auch aus Sicht der Informationstechnologie darzustellen.

Außerdem wird der Begriff der Visualisierung erläutert. Von der Entwicklung bis zu den verschiedenen Arten der Visualisierung. Wesentlicher Bestandteil ist die Beschreibung der Informations- Visualisierung. Wobei hauptsächlich auf die Visualisierung von Strukturen eingegangen wird. Über eine Gegenüberstellung gebräuchlicher Arten zur Darstellung von Hierarchien soll schlussendlich auch eine geeignete Darstellungsart herausgefiltert werden, um das Verzeichnissystem des Austria-Forums webbasiert darzustellen.

Im 3. Kapitel wird grundlegend auf Wiki-Systeme eingegangen. Was sind Wikis, welche Features stellen sie grundsätzlich zur Verfügung. Außerdem werden einige wichtige *Wiki-Engines* vorgestellt.

Im 4. Kapitel wird JSPWiki und die wichtigsten Änderungen des daraus resultierenden Wiki-Systems des Austria-Forums vorgestellt. Ausführlicher wird die hierarchische Dateienverwaltung des Austria-Forums vorgestellt, welche einen wesentlichen Punkt in Richtung *structured Wiki* darstellt.

Im 5. Kapitel wird die Umsetzung der webbasierten Visualisierung der Dateienstruktur einschließlich der Interaktionsmöglichkeiten und deren Um-

setzung vorgestellt. Welche Technologien oder Frameworks wurden verwendet? Wie wurde das Tool getestet?

Im 6. Kapitel werden abschließend die wichtigsten Erkenntnisse zusammengefasst.

Siegi Oberhauser, März 2012

2. Aspekte aus der Informationstechnologie

2.1 Dateisystem – Verzeichnissystem

Computer sind heutzutage nicht nur Rechenmaschinen, sondern besitzen in den meisten Fällen auch Speichermedien wie Festplatten, Speicherkarten oder DVD/CD-ROM, auf denen Daten dauerhaft abgelegt, organisiert und auch manipuliert werden können. Für diese Ablageorganisation gibt es das Dateisystem, welches ein wesentlicher Bestandteil des Betriebssystems ist (Geuter, 2009).

Ein Dateisystem muss grob gesagt zwei Aufgaben bewältigen:

- Repräsentation der Daten für den Benutzer
- Organisation der Daten am Speicher

Aus Sicht des Benutzers ist die Repräsentation des Dateisystems gegenüber ihm die wichtigste Eigenschaft. Das heißt, woraus eine Datei besteht, wie sie benannt und geschützt wird, welche Operationen auf ihr durchführbar sind und vor allem wie die Strukturierung der Dateien in Verzeichnisse (Verzeichnisstruktur) aussieht (Tanenbaum, 2002).

Die Organisation der Daten am Speicher, also Fragen ob verkettete Listen oder Bitmaps zur Verwaltung des freien Speicherplatzes benutzt werden oder wie viele Sektoren in einem logischen Block untergebracht sind, bleibt dem Benutzer verborgen und muss ihn auch nicht interessieren (Geuter, 2009).

Viele heute gebräuchliche Dateisysteme verwenden zur internen Speicherverwaltung am Datenträger als Datenstruktur (siehe Kapitel 1.3 Datenstruktur) den B-Baum (NTFS, HPFS, HFS), die Tabelle (FAT, ext2, ext3) oder die verkettete Liste (ext4).

In diesem Kapitel werde ich jedoch fast ausschließlich auf den Aspekt der Repräsentation der Daten bzw. Dateien eingehen.

2.1.1 Datei

Um Daten überhaupt speichern, lesen, löschen und ändern zu können oder überhaupt wieder zu finden, wird eine endliche Menge zusammengehöriger Daten bzw. Informationen (Texte usw.) zu einer logischen Einheit, der Datei, zusammengeschlossen und mit einem Namen versehen. Sie sind der atomare Bestandteil eines Dateisystems, weshalb Daten nur in solchen Einheiten gespeichert werden können. Dateien haben immer einen eindeutigen Namen und diverse Attribute um gewisse Aussagen machen zu können (Tuschl, 2008).

2.1.1.1 Dateinamen

Das wahrscheinlich wichtigste Merkmal der Abstraktion Datei ist die Benennung dieser. Dies variiert von System zu System. Bei älteren Dateisystemen waren Zeichenfolgen von bis zu acht Buchstaben erlaubt. Oft dürfen Zahlen oder Sonderzeichen verwendet werden. Heutzutage sind Längen von 255 Zeichen Standard. Unter UNIX wird auch zwischen Groß- und Kleinschreibung unterschieden, während dies z.B. bei MS-DOS bzw. MS-Windows egal ist. So sind unter UNIX *datei.txt* und *DATEI.TXT* zwei verschiedene Dateien, während unter Windows-Systemen beides auf die gleiche Datei zurückführen würde (Zeiner, 2010).

Von vielen Betriebssystemen werden zweigeteilte Dateinamen unterstützt, die durch einen Punkt getrennt werden. Der Teil nach dem Punkt ist die Dateierweiterung (file extension) und enthält in der Regel Informationen über die Datei wie z.B. um welchen Typ von Datei es sich handelt. Die Datei *NAME.TXT* würde in diesem Fall eine Textdatei bezeichnen. In UNIX werden Erweiterungen nicht erzwungen, hier sind sie nur eine Konvention. Windows hingegen weißt den Erweiterungen eine Bedeutung zu.

So kann jeder Erweiterung ein Programm zugewiesen werden, sodass es durch Doppelklick einfach geöffnet werden kann (Tanenbaum, 2002).

2.1.1.2 Dateityp

Damit das Betriebssystem weiß, wie es eine Datei interpretieren soll, muss es Informationen haben, um welchen Dateityp es sich bei einer Datei handelt. Es gibt eine große Anzahl von Typen (siehe: http://en.wikipedia.org /wiki/List_of_file_formats) die von System zu System anders erkannt werden. Wie schon vorher beschrieben, gibt die Dateierweiterung bei Windows-Systemen Auskunft über das Dateiformat. Dies kann zu Problemen führen wenn die Extension evtl. ungewollt geändert wird. In diesem Fall würde das System diese Datei als falsches Format ansehen und daher falsch interpretieren. UNIX basierte Systeme erkennen das Format anhand einer sogenannten *magischen Zahl* im Header der Datei. Eine weitere Möglichkeit ist es den Dateityp in den Dateiattributen, auch Metadaten einer Datei genannt, mitzuteilen (Dateiformat, 2011).

2.1.1.3 Dateiattribute

Neben Namen und Inhalt verbinden alle Betriebssysteme weitere Informationen mit Dateien, die Dateiattribute. Diese können neben den vom System erzeugten Informationen auch Angaben des Benutzers enthalten. Die Art und Anzahl der Attribute ist vom Dateisystem abhängig. Typische Attribute sind (Tanenbaum, 2002):

- Größe
- Datum und Zeit
- Eigentümer
- Berechtigungen
- Typ

2.1.1.4 Dateioperationen

Um nun mit den gespeicherten Dateien arbeiten zu können stellen die Systeme unterschiedliche Dienste zur Verfügung. Einige dieser Dienste sind (Baumgartner & Siegert, 2007):

- *erzeuge_datei(name,attribut):* Eine Datei mit angegebenem Namen wird erzeugt.
- *lösche_datei(name):* Die angegebene Datei wird gelöscht. Sollte die Datei von anderen Nutzern geöffnet sein, gibt es von System zu System verschiedene Optionen wie weiter verfahren werden soll.
- *lese_attribute(name,attribute):* Die Attribute der angegebenen Datei werden ausgelesen.
- *änder_attribute(name,attribute):* Die angegebenen Dateiattribute werden geändert.
- *öffne_datei(name,modus):* Die Datei wird je nach der im Modus angegebenen Bearbeitungsart (z.B. nur lesen) geöffnet.
- *schließe_datei(name):* Die Datei wird von der Bearbeitung abgemeldet. Das Schließen erfolgt, sobald alle Informationen aus dem Hintergrundspeicher aktualisiert worden sind.
- *umbenennen_datei(name,neuer_name):* Die angegebene Datei wird umbenannt. Sollte die Datei von anderen Nutzern geöffnet sein, gibt es von System zu System verschiedene Optionen wie weiter verfahren werden soll.

2.1.2 Verzeichnis - Verzeichnissystem

Während eines Computerlebens sammeln sich riesige Mengen an Dateien an, die sinnvoll verwaltet werden sollten, um den Überblick zu bewahren. In modernen Betriebssystemen erfolgt dies in Verzeichnissen, welche hierarchisch strukturiert sind. Verzeichnisse sind *spezielle* Dateien, weshalb keine bestimmten Dienstprogramme zur Ausführung und Wartung benötigt werden. Je nach Aufbau der Verzeichnistopologie kann man folgende Strukturen unterscheiden (Red Hat, 2011).

2.1.2.1 Lineare Struktur ohne Verzeichnis

Die ersten Systeme hatten eine lineare Struktur ohne Verzeichnis. Hier wurden die Daten einfach linear am Medium abgelegt. Solche Strukturen wurden für Lochband- und Lochkartensystemen eingesetzt, aber auch in den Anfangszeiten des Diskettenzeitalters verwendet (z.B. unter MS-DOS Version 1). Heute finden sie noch bei der Datensicherung durch Magnetbänder Verwendung (Dateisystem, 2011).

2.1.2.2 Verzeichnisstruktur mit einer Ebene

Bei dieser Struktur gibt es nur ein globales Verzeichnis, das Wurzelverzeichnis (Root), in welches alle vorhandenen Dateien gespeichert werden. Weitere Verzeichnisse sind nicht möglich.

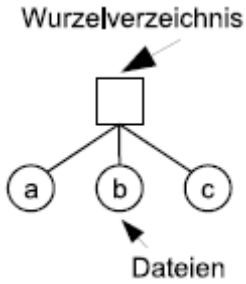

Abbildung 6: Verzeichnisstruktur mit einer Ebene (Tanenbaum, 2002)

Dieses System ist zwar einfach zu verstehen und zu warten, gleich wie die Struktur ohne Verzeichnis, hat jedoch signifikante Nachteile, wenn die Anzahl der Dateien zunimmt oder mehrere Nutzer einen Rechner verwenden. Jede Datei muss einen anderen Namen haben. Sollten an einem System z.B. 100 User arbeiten, wird die Namensvergabe bei steigender Anzahl der Dateien immer komplexer oder Dateien werden einfach von anderen Usern überschrieben. Bei alten Systemen kam noch die maximale Zeichenlänge für Dateinamen dazu, welche das Problem der Namensvergabe auch nicht erleichterte (Silberschatz, Peterso, & Galvin, 1991).

Solche Methoden finden bei heutigen Mehrbenutzersystemen keine Verwendung mehr. Einzig auf kleinen eingebetteten Rechnern könnte diese Struktur noch Verwendung finden.

2.1.2.3 Verzeichnisstruktur mit zwei Ebenen

Um dem Nachteil der Namensgebung unter mehreren Benutzern auszuweichen, wurde diese Handhabung eingeführt. Hier wird einfach jedem User ein eigenes Verzeichnis zugewiesen. So stört die Namensgebung eines Users nicht die eines anderen. Dieser Ansatz ist neben Mehrbenutzer Rechnern auch in einfachen Netzwerken aus PCs mit einem Dateiserver möglich. Solche Systeme erfordern eine Login-Prozedur, um dem System mitzuteilen, in welchem Verzeichnis gearbeitet werden darf (Tanenbaum, 2002).

Diese Struktur reicht jedoch auch nur Usern mit wenigen Dateien. Heute können Benutzer mitunter mehrere tausend Dateien unterschiedlichen Typs haben. Hier ist auch dieses System bald an seine Grenzen getrieben und die Namensgebung sowie die Übersicht wird mit jeder zusätzlichen Datei schwieriger.

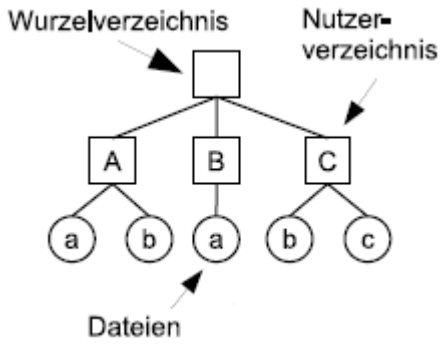

Abbildung 7: Verzeichnisstruktur mit zwei Ebenen (Tanenbaum, 2002)

2.1.2.4 Hierarchische Verzeichnisstruktur

Um die Schwierigkeiten aus den vorher beschriebenen Systemen zu verbessern ging die hierarchische Verzeichnisstruktur hervor. Hier kann je-

der Benutzer die für seine Struktur notwendigen Unterverzeichnisse anlegen. Auch die Tiefe der Topologie ist egal, wodurch eine vielschichtige Verzeichnisstruktur gebildet werden kann (Red Hat, 2011).

Durch diesen Ansatz ist es möglich, die Dateien in natürlicher Weise zu ordnen. Jeder Benutzer besitzt somit ein mächtiges Instrument, um seine Arbeit zu strukturieren und dadurch die Dateiverwaltung erheblich zu erleichtern. Dies ist auch der Grund, warum fast alle modernen Dateisysteme in dieser Weise organisiert sind (Tuschl, 2008).

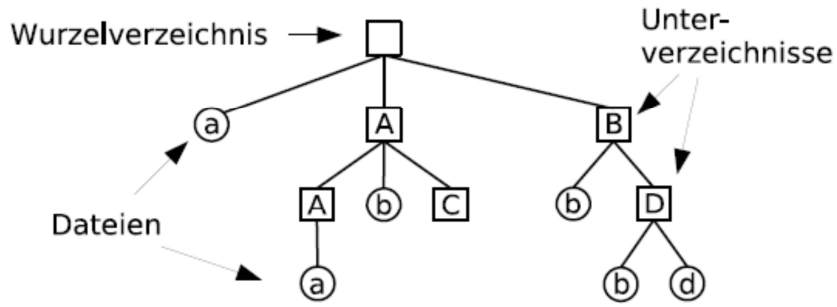

Abbildung 8: Hierarchisches Verzeichnissystem (Tuschl, 2008)

Die Hierarchie hat eine Baumstruktur. An der Spitze ist das Wurzelverzeichnis. Jede Datei im System ist als Blatt dargestellt und über einen einmaligen Pfadnamen ansprechbar. Der Pfadname beschreibt die Navigation von der Wurzel über allfällige Unterverzeichnisse, welche beliebig viele weitere Dateien und Unterverzeichnisse beinhalten können, bis hin zur Datei. Die Pfadnamen können *absolut* oder *relativ* ausgedrückt werden. Ein absoluter Pfadname bezeichnet immer den Weg von der Wurzel bis zur Datei. Die relative Pfadangabe zeigt den Pfad vom derzeitig verwendeten Verzeichnis zur gewünschten Datei (Silberschatz, Peterso, & Galvin, 1991).

Die einzelnen Komponenten des Pfades werden mit einem Separator, welcher von System zu System verschieden ist, getrennt. Das windowsspezifische Trennzeichen ist ein *Back-Slash* „\“. Unter Unix wird der nor-

male *Slash* „/“ dafür verwendet. Beispielsweise würde der Pfad der Datei d in Abbildung 8 unter Unix folgend aussehen: */B/D/d* (Tuschl, 2008).

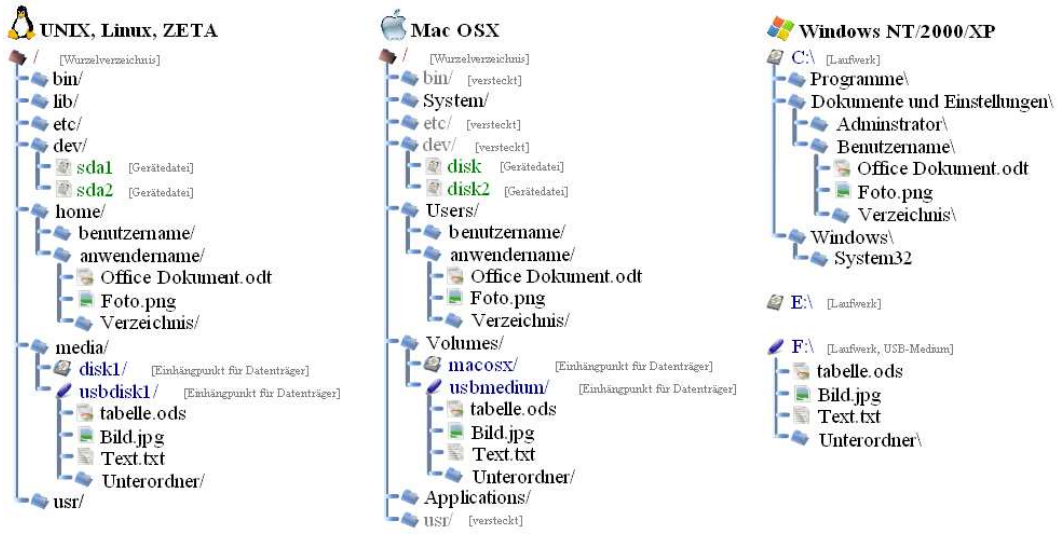

Abbildung 9: Verzeichnisstruktur verschiedener Betriebssysteme (Verzeichnisstruktur - Wikipedia, 2011)

2.1.2.5 Verzeichnisoperationen

Wie bei den Dateien, gibt es auch Operationen, die auf Verzeichnisse durchführbar sind. Auch hier gibt es systembedingte Unterschiede. Die wichtigsten Operationen sind (Tanenbaum, 2002):

- *erzeuge_verzeichnis(name):* Ein neues Verzeichnis wird angelegt.
- *lösche_verzeichnis(name):* Ein Verzeichnis wird gelöscht. Unter Unix kann nur ein leeres Verzeichnis gelöscht werden. Unter Windows können auch nichtleere Verzeichnisse, also samt deren Inhalt, gelöscht werden.
- *öffne_verzeichnis(name):* Ein Verzeichnis wird geöffnet um beispielsweise alle darin vorkommenden Dateien oder Verzeichnisse aufzulisten.
- *schließe_verzeichnis(name):* Ein Verzeichnis wird geschlossen.
- *lese_verzeichnis(name):* Die Einträge eines geöffneten Verzeichnisses werden zurückgegeben.
- *umbenennen_verzeichnis(name,neuer_name):* Ein Verzeichnis wird umbenannt.

2.1.3 Gegenüberstellung einiger bekannter Dateisysteme

Es gibt eine Menge Dateisysteme. In der folgenden Tabelle sollen zumindest die bekanntesten davon kurz gegenübergestellt werden.

Datei-system	Betriebs-system nativ	Technische Umsetzung	Länge Dateiname	Anzahl Dateien	Dateirechte-verwaltung
FAT16	MS-DOS 3.31	Tabelle	8.3	2^{16}-19	Nein
FAT32	Windows 95, 98	Tabelle	255 UTF16 Zeichen	2^{28}-19	Nein
NTFS	Windows NT, 2000, XP, Vista, 7	B+Tree	255 UTF16 Zeichen	2^{32}-1	ACL
HPFS	OS/2	B+Tree	255 Zeichen	unlimitiert	Nur in HPFS386
ext3	ab Linux 2.4.15	Tabelle, hashed B-Tree	254 Bytes	variabel	Unix Datei-rechte, ACL
ext4	ab Linux 2.6.19	Verkettete Liste, hashed B-Tree	256 Bytes	unlimitiert	POSIX
HFS +	Mac OS	B-Tree	255 UTF16 Zeichen	2^{32}-1	Unix Datei-rechte, NFSv4, ACL

Tabelle 1: Gegenüberstellung wichtiger Dateisysteme (Comparison File Systems - Wikipedia, 2011)

2.1.4 Warum hierarchische Dateisysteme - Verzeichnissysteme

Wie schon in den vorherigen Kapiteln beschrieben, wird das Verzeichnissystem aller gängigen Dateisysteme hierarchisch gestaltet.

Ein wichtiger Vorteil dieser Begebenheit ist, wie auch schon beschrieben, die Umgehung des Namenskonfliktes.

Bei den heutigen Datenmengen ist eine Strukturierung unumgänglich. Man muss sich nur vorstellen wie die Arbeit am PC aussehen würde, wenn die Dateien noch immer flach in einem Verzeichnis abgelegt würden. Schon bei privat benutzten Rechnern gehören Plattengrößen von mehreren Terabyte zur Grundausstattung. Bei diesen Speichergrößen würden sich ohne hierarchische Gestaltung des Systems mehrere Tau-

send Dateien auf einer einzigen Ebene befinden. Das Suchen nach speziellen Daten oder Informationen würde dementsprechend ineffizient werden. Ein gut strukturiertes Verzeichnissystem ergibt zwar auch einen gewissen Mehraufwand, resultiert jedoch in einem effektiven Suchsystem innerhalb dieser Struktur und einer immensen Erleichterung bei der täglichen Arbeit am PC. Selbst automatischen Suchsystemen wird die Arbeit stark erleichtert. Der Such-Algorithmus müsste sich ansonsten bei jeder Such-Operation durch die gesamte Anzahl der flach gespeicherten Dateien hangeln (Winkler a, 2007).

Was sich für Verzeichnissysteme in Betriebssystemen bereits lange bewährt, sollte sich natürlich auch für das Austria-Forum bewähren. Auch hier werden sich mit Fortbestand des Systems riesige Mengen an Informationen bzw. Daten ansammeln die vernünftig, durch ein hierarchisches Verzeichnissystem, strukturiert werden wollen. Somit gelten alle Eigenschaften die eine hierarchische Verzeichnisstruktur in Betriebssystemen befürworten auch hier.

2.2 Visualisierung

Wie in Kapitel 1.3.3 schon gesagt, eignen sich Bäume besonders gut um hierarchische Strukturen zu visualisieren. Daher möchte ich in diesem Kapitel auch grundlegend auf die Visualisierung in der Informationstechnologie eingehen.

„*Ein Bild sagt mehr als 1000 Worte*“

Genauso gut könnte in diesem Spruch das Wort *Bild* mit *Graphik* oder *Visualisierung* ersetzt werden. Die Zuhilfenahme graphischer Mittel zur Beschreibung von Informationen, Daten oder der Zusammenhänge von Dateien bringt sehr oft eine effektive Erleichterung, da das menschliche Wahrnehmungssystem stark an die effektive Verarbeitung von visuell kodierten Informationen angepasst ist. Dadurch wird es möglich - unter Vo-

raussetzung einer qualitativ hochwertigen visuellen Repräsentation - große Mengen von Information rasch interpretieren zu können (Ludwig, 2004).

Aufgrund der Generierung fließender dynamischer Bilder mithilfe von Computern, welche dem Verlauf der Informationen nachkommen, ist eine wesentliche Verbesserung dieser Vorzüge möglich. Leistungsstärkere Rechner und neue Technologien ermöglichen zudem immer bessere Visualisierungstechniken mit zunehmend höherem Grad an Interaktionsmöglichkeiten. Auch im Web können solche Visualisierungen dem User durch ständig wachsende Datenübertragungsraten zunehmend zugänglich gemacht werden (Ludwig, 2004).

2.2.1 Geschichte

Die graphische Übermittlung von Informationen ist im Grunde älter als die Übermittlung dieser mittels Schrift. Relikte dafür finden sich in vielen Höhlen weltweit, in welchen Menschen durch Wandmalereien Informationen ihrer Zeit hinterließen.

Abbildung 10: Höhlenmalerei aus der Höhle von Lascaux (Höhle Lascaux - Wikipedia, 2011)

Definierte man früher *Visualisierung* eher mit einem visuellen Konstrukt des Gedankens, so versteht man heute darunter auch eine graphische Darstellung von Daten oder Konzepten (Ludwig, 2004).

Es dauerte an die 5000 Jahre, um ein messbares Koordinatensystem mit den Variablen X und Y aus einem Koordinatensystem welches aus Variablen, die mit dem Namen der Himmelsrichtungen bezeichnet waren, zu

verwandeln. Im 15. Jahrhundert wurde durch florentinische Architekten die perspektivische Projektion entwickelt. Diese ist eine Erweiterung der zweidimensionalen Oberfläche und ermöglicht es, Objekte in einem geometrisch korrekten Kontext im dreidimensionalen Raum darzustellen (Ludwig, 2004).

Angenommen wird, dass die frühesten Arbeiten der Datengrafik, also die Verwendung von abstrakten visuellen Eigenschaften wie Linien und Bereiche um Daten darzustellen, aus der Zeit um Diderot und d'Alembert (Figurative System of Human Knowledge (1751)) (Andrews K. , 2008) und Playfair (1786) (Ludwig, 2004) stammen.

Diderot und d'Alembert stellten in einem Baum das System des menschlichen Wissens für ihre Enzyklopädie dar (siehe Abbildung 11).

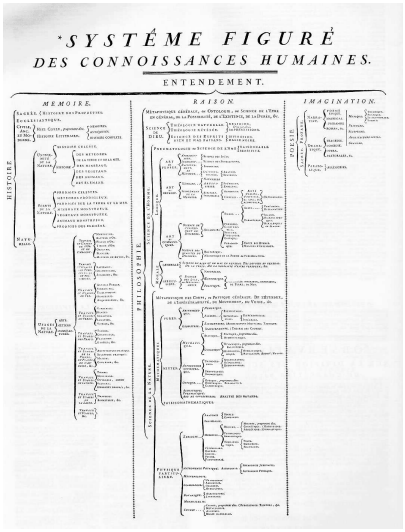

Abbildung 11: "Figurative System of Human Knowledge" (1751) oder "The tree of Diderot and d'Alembert" (Figurative human knowledge - Wikipedia, 2011)

Florence Nightingale entwickelte im Jahre 1858 das *Polar Area Diagram* (siehe Abbildung 12). In diesem dokumentierte sie die medizinische Versorgung der britischen Armee während des Krimkrieges. Sie gilt damit als Pionierin visueller Darstellung mathematischer Zusammenhänge.

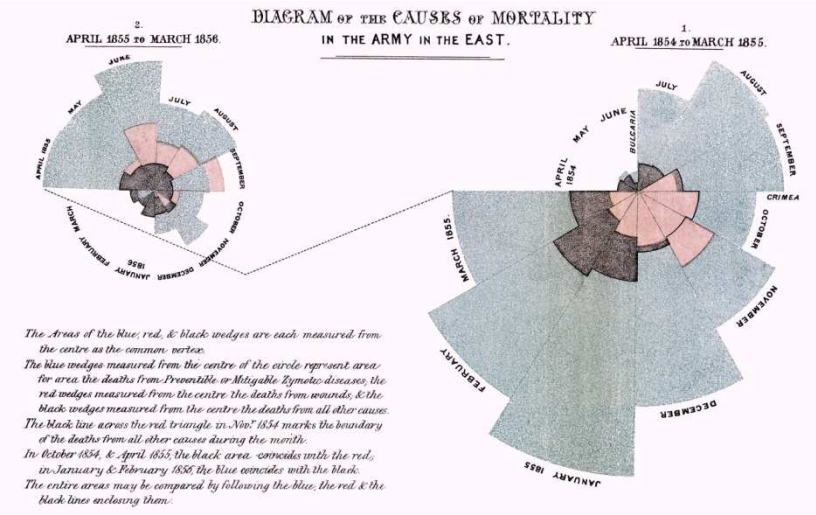

Abbildung 12: Polar Area Diagram von Florence Nightingale (1858) (Florence Nightingale - Wikipedia, 2011)

Charles Minard gilt ebenfalls als Pionier der grafischen Informationsvermittlung. Sein bekanntestes Werk war ein Diagramm aus dem Jahre 1861 (siehe Abbildung 13), in welchem der Russlandfeldzug Napoleons von 1812-1813 dokumentiert wird.

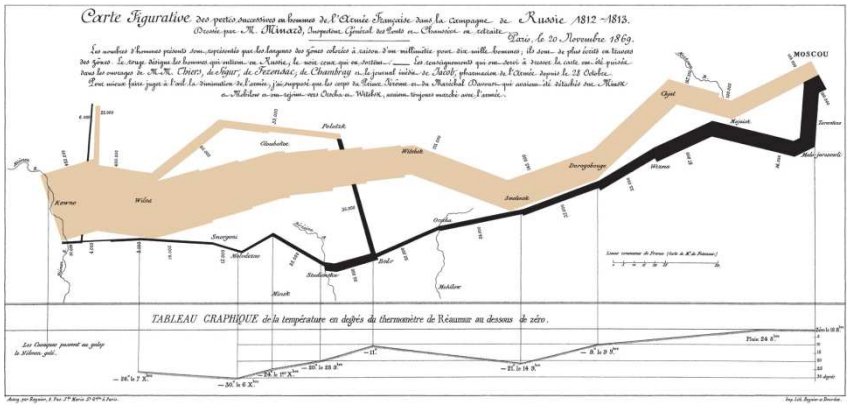

Abbildung 13: Minards Grafik über den Rußlandfeldzug (1861) (Charles Minard - Wikipedia, 2011)

Jacques Bertin war ein bedeutender französischer Kartograph, der mit seinem Buch *Semiologie Graphique* (1967) als Erster ein Standardwerk zur graphischen Theorie und zur Visualisierung allgemein verfasste (Jacques Bertin - Wikipedia, 2011).

1983 veröffentlichte Tufte eine Theorie der Datengrafiken, welche die Maximierung der Dichte von relevanter Information betonte. Die Theorien

von Bertin und Tufte waren der Hauptgrund, dass sich die Informations-Visualisierung als eigene Disziplin entwickelte. 1985 startete die National Science Foundation (NSF) die Initiative der *Scientific Visualization*. 1990 fand dann die erste IEEE Visualization Conference statt (Ludwig, 2004).

2.2.2 Begriff und Arten der Visualisierung

Im Grunde stellt die Visualisierung eine Verbindung zwischen einem abstrakten (mathematischen) Objekt und einem Gegenstand der realen Welt her. Eine wesentliche Aufgabe bei der Visualisierung ist das Finden von Darstellungen die der Erfahrung zugänglich sind. Liegt ein reales System vor, so wird eine solche Darstellung oft durch die Realität nahegelegt. Bei rein mathematischen Modellen oder physikalischen Prozessen, für die der Mensch keine Sinnesorgane besitzt, existiert kein direkter Bezug so dass Analogien zu bekannten Objekten und Vorgängen gefunden werden (siehe Abbildung 14). Die Visualisierung dient dabei sowohl der Darstellung der Ergebnisse, der Überprüfung der Modellvorstellung als auch der Vermittlung der Modelle und Schlussfolgerungen (Kuska, 2000).

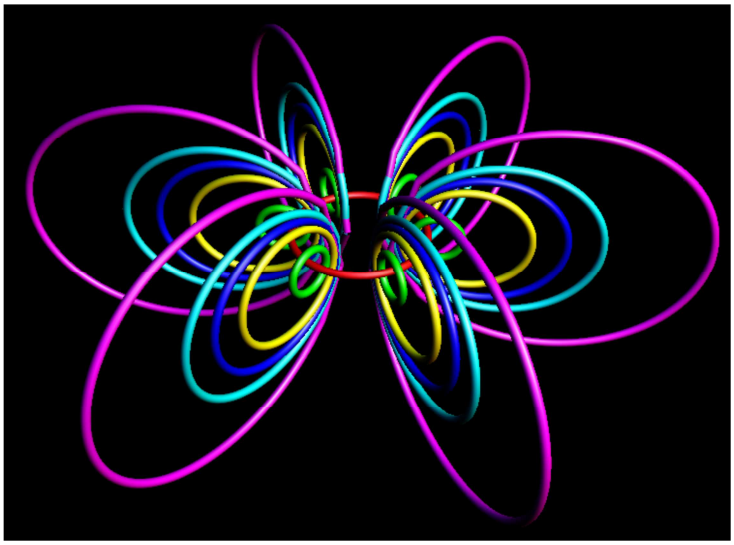

Abbildung 14: Visualisierung von Magnetfeldlinien einer Leiterschleife (Kuska, 2000)

Grundsätzlich kann man bei der Visualisierung von Daten zwischen Datenpräsentation und Datenexploration unterscheiden. Unter Präsentation

versteht man die Bereitstellung von geeigneten Präsentationsformen um bereits bekannte Fakten darzustellen. Mit der Exploration sollen durch angemessene Visualisierungen unbekannte Verknüpfungen zwischen Thematiken aufgedeckt werden. Visualisierungen, die sich den Computer zunutze machen haben sich als unabhängige Disziplin im Bereich der Mensch-Computer-Interaktion (Human Computer Interaction HCI) entwickelt. In den 80er Jahren wurden computerbasierte Visualisierungen vor allem in der Wissenschaft angewandt, daraus entwickelte sich die wissenschaftliche Visualisierung bzw. *Scientific Visualization*. Seit den 90ern kommen die Visualisierungen auch immer mehr in allgemeinen Bereichen vor, woraus sich die Informationsvisualisierung bzw. *Information Visualization* entwickelte (Ludwig, 2004).

2.2.2.1 Scientific Visualization

Als Scientific Visualization wird die Wissenschaft der Visualisierung oder Simulation von Daten bezeichnet, welche direkt physikalischen Prozessen zugeordnet werden können (Visualisierung, 2011).

Der Computer wird verwendet um Visualisierungen und Animationen (oftmals in 3D) der Eigenschaften von physischen Elementen (z.B.: menschlicher Körper, Gebäude, Naturphänomene, technische Konstruktionen, Moleküle oder die Erde) darzustellen (siehe Abbildung 15). Hier werden die modellierten Objekte detailliert studiert, geprüft und manipuliert um wissenschaftliche Hypothesen zu testen, eine Vorgehensweise virtuell auszuprobieren oder durch Verwendung von *Virtual Reality* virtuelle Rundgänge durchzuführen. Ziel ist es meist, das Innere eines Objektes zu erforschen und dadurch eine Vorbereitung vor der realen Durchführung zu haben (z.B.: Flugsimulationen, Crash-Test Simulationen in der Automobilindustrie usw.) (Ludwig, 2004).

Abbildung 15: Visualisierung der Wasseroberfläche nach Eintauchen eines Körpers (Scientific_visualization - Wikipedia, 2011)

2.2.2.2 Information Visualization

Ziel der Information Visualization bzw. Informationsvisualisierung ist es, abstrakte Daten graphisch so zu repräsentieren, dass strukturelle Zusammenhänge und relevante Eigenschaften intuitiv erfasst werden können. Die Daten müssen nicht notwendigerweise einen physikalischen Bezug aufweisen. Hierbei wird die große Leistungsfähigkeit des menschlichen visuellen Systems ausgenutzt, um die charakteristischen Eigenschaften einer Datenmenge zu erfassen. Weiter soll die Informationsvisualisierung, ein in hohem Maße, interaktiver Prozess sein, der visuelle und automatische Methoden verknüpft und dem Benutzer die Möglichkeit der Interaktion bietet. Dieser Zusammenhang kann durch das *Data State Reference Model* von Chi beschrieben werden (siehe Abbildung 16) (Schumann, 2004).

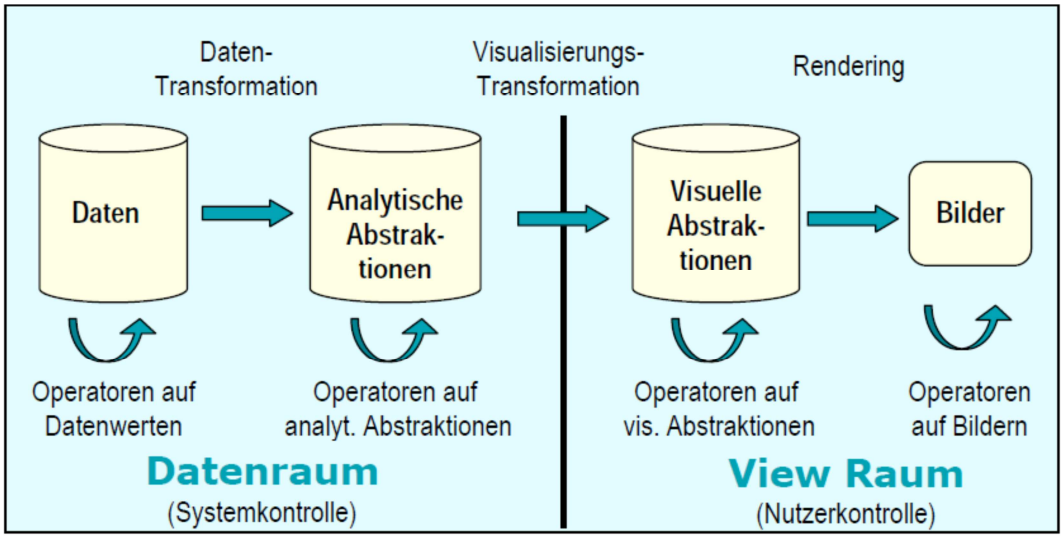

Abbildung 16: Data State Reference Model (Schumann, 2004)

Das Model beschreibt die stufenweise Abbildung der Daten über analytische und geometrische Abstraktionen auf Bilddaten sowie die Verknüpfung der Operatoren der unterschiedlichen Stufen welche zu einer datenflussorientierten Beschreibung des Visualisierungsprozesses führt, welche den meisten Visualisierungssystemen zugrunde liegt. Es bieten jedoch nicht alle Systeme dieselbe Unterstützung auf allen Stufen des Models an (Schumann, 2004).

Ein weiteres Schlüsselkonzept der Informationsvisualisierung ist die Skalierbarkeit. Es soll möglich sein, dass verschiedene Bereiche einer eventuell groß geratenen Visualisierung im Display gut darstellbar sind, also die Möglichkeit einer Detailansicht und Gesamtansicht gegeben ist. Des Weiteren sollte eventuell gleich ersichtlich sein, in welchem Kontext dieser Bereich dann steht (Thomas, 2004).

2.2.2.2.1 Visualisierung von Strukturen

Eine wichtige Teildisziplin der Informationsvisualisierung ist die Visualisierung von Strukturen, also die Analyse und Darstellung der Beziehung zwischen Datensätzen oder Variablen. Hauptsächlich beschäftigt man sich mit der Darstellung von Hierarchien (Schumann, 2004).

Grundsätzlich lassen sich die Möglichkeiten der Darstellung von Hierarchien in 3 Vorgehensweisen einteilen (Schumann, 2004):

- *2D oder 3D*
  Erfolgt die Darstellung in der Ebene oder im Raum.

- *Achsenparallel oder Radial*
  Erfolgt die Darstellung entsprechend den Achsen des Präsentationsraumes oder radial um die Wurzel.

- *Explizit oder Implizit*
  Werden die Kanten explizit dargestellt oder implizit durch entsprechende Anordnung der Knoten veranschaulicht. Also wird die Darstellung als *Graph* (z.B. Baum) oder als *Space Filling Methode* (z.B. Treemap) ausgeführt.

Üblicherweise werden heute, insbesondere für kleinere Knoten- und Kantenmengen, explizite Techniken, die so bezeichneten Node-Link-Diagramme eingesetzt. Ein weiterer Grund für die Beliebtheit dieser Technik (z.B. Bäume) ist, dass sie intuitiv leichter fassbar ist. Allerdings werden mit zunehmender Datengröße auch implizite Techniken immer populärer (Schumann, 2004).

2.2.2.2.1.1 Klassischer Baum

Klassische Bäume sind eine explizite, achsparallele Darstellungsform. Sie erscheinen uns als die natürlichste Möglichkeit der Darstellung für Hierarchien, da für viele die klassische Baumdarstellung mit Hierarchien äquivalent ist. Der Baum wächst oben von der zentral platzierten Wurzel nach unten. Die Beziehungen zwischen den einzelnen Knoten bzw. zwischen Eltern- und Kindknoten wird explizit durch Kanten dargestellt. Die Kanten können orthogonal oder auch diagonal mit verschiedensten Winkeln ausgeführt werden. In der Software-Technik ist diese Form aufgrund von einigen Problemen nicht so oft anzufinden, da sie sehr platzintensiv ist. El-

ternknoten können mitunter sehr viele Kindknoten haben. Bereits eine moderate Anzahl an Kindern führt platztechnisch schon zu großen Problemen. Um eine einigermaßen ästhetische Darstellung zu erreichen sollten sehr große Abstände zwischen den Elementen gewählt werden. Dies führt dazu, dass große Hierarchien evtl. nur Ausschnittsweise dargestellt werden können. Eine Orientierung in der Hierarchie über mehrere Ebenen ist schwierig, da der Kontext, in dem der aktuelle Ausschnitt liegt, schwer nachvollziehbar ist (Jetter, 2006).

Eine kompakte Darstellung könnte diesen Nachteil teilweise verbessern, doch sind die dafür notwendigen Algorithmen sehr rechenintensiv. Dies wirkt sich mitunter störend auf die sonstigen Manipulationen bzw. Interaktionen aus, da das System stark belastet wird (Winkler, 2006).

Der *Reingold-Tilford-Algorithmus* wäre einer dieser Ansätze, welcher versucht mit so klein als möglichen Zwischenräumen auszukommen. Dieser ist für Binärbäume konzipiert. Walker II erweiterte diesen Algorithmus dann auch für Bäume höheren Ranges (Nussbaumer, 2005).

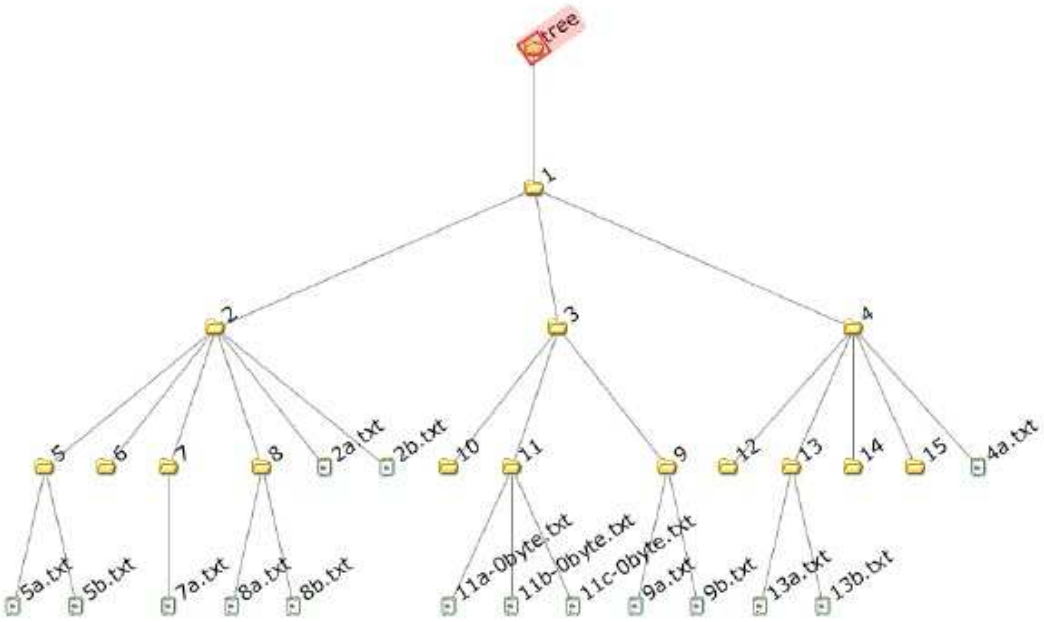

Abbildung 17: Klassischer Baum – Walker Layout (Andrews K. , 2008)

2.2.2.2.1.2 Liste

Die Interpretation als Liste ist eine der ältesten und einfachsten Visualisierungsformen für hierarchische Daten. Diese Methode ist mit ihrer Grundidee die Basis für andere Varianten zur Hierarchiedarstellungen. Jeder Knoten wird einfach als Zeile angezeigt, welche auch die Knoteneigenschaften, wie die Position in der Hierarchie über den Pfad, wiedergibt. Der Nachteil dieser Form ist, dass die Struktur der Hierarchie sowie der Kontext, indem die dargestellten Knoten sich befinden, kaum visualisiert sind. Dies muss der Benützer im Gedanken erledigen, um sich orientieren zu können (Winkler, 2006).

Bekannte Vertreter dieser Technik sind das Windows-Prompt *dir* und der UNIX-Befehl *ls*. Durch eine Vielzahl von Parametern hat man mit *ls* auch die Möglichkeit beispielsweise eine Auswahl von Attributen, den Ausschnitt der Hierarchie, des *Farbhighlightings* und viele andere Eigenschaften darzustellen. Will man damit arbeiten, muss die Syntax des Dateisystems bzw. der Kommandozeilen erlernt werden. Erfahrene Benutzer können damit jedoch schnell und gezielt im Dateisystem arbeiten und eine Hierarchie *explorieren* um darin zu navigieren (Jetter, 2006).

```
brw-r--r--    1 unixguy staff 64,  64 Jan 27 05:52 block
crw-r--r--    1 unixguy staff 64, 255 Jan 26 13:57 character
-rw-r--r--    1 unixguy staff     290 Jan 26 14:08 compressed.gz
-rw-r--r--    1 unixguy staff  331836 Jan 26 14:06 data.ppm
drwxrwx--x    2 unixguy staff      48 Jan 26 11:28 directory
-rwxrwx--x    1 unixguy staff      29 Jan 26 14:03 executable
prw-r--r--    1 unixguy staff       0 Jan 26 11:50 fifo
lrwxrwxrwx    1 unixguy staff       3 Jan 26 11:44 link -> dir
-rw-rw----    1 unixguy staff     217 Jan 26 14:08 regularfile
```

Abbildung 18: Listendarstellung mit Unix-Befehl „ls“ (LS_Unix – Wikipedia, 2011)

2.2.2.2.1.3 Tree Browser

Der Tree Browser ist eine Synthese aus Ansätzen des klassischen Baums und der Liste, ein sogenannter *Outliner*. Diese Form zählt zu den populärsten Techniken der Darstellung von Hierarchien, da z.B. der *Microsoft Windows Explorer* oder der *KDE Konqueror* Vertreter davon sind. Somit

wird diese Form von den meisten Leuten welche an Computern arbeiten benützt (Nussbaumer, 2005).

Tree Browser stellen ein intuitives und einfach zu verstehendes Interface zur Verfügung. Üblicherweise verfügen sie über zwei Fenster. Im Linken wird die Struktur der Hierarchie, die Baumtopologie, eingeblendet. In diesem Fenster können User durch die Hierarchie navigieren und die sogenannten *Subtrees* bzw. Verzeichnisse erweitern oder reduzieren (*expand* oder *collaps*). Beim Erweitern wächst die Struktur horizontal und vertikal. Sollte die Darstellung nicht in den verfügbaren Anzeigenbereich passen, werden *Scrollbars* als zusätzliches Navigationselement zur Verfügung gestellt. Klickt man beim Navigieren auf ein Verzeichnis, wird im rechten Fenster dessen Inhalt in einer Listendarstellung angezeigt (Putz, 2005).

Die kompakte Darstellung der Baumtopologie hat folgende Gründe. In der Baumansicht, im linken Fenster, werden nur Verzeichnisse (Knoten mit Kindern) jedoch ohne Dateien, also ohne Blätter, dargestellt. Die Verzeichnisse werden zeilenweise angezeigt, weshalb der platzsparende Effekt der Liste ausgenutzt wird. Die Baumtopologie wird durch orthogonale Kanten und Einrückungen dargestellt, wodurch ein visueller Effekt entsteht, mit welchem interagiert werden kann. Der Detailgrad kann durch das Erweitern bzw. Reduzieren von Knoten gezielt gesteuert werden.

Geht es darum Dateien oder Verzeichnisse auf derselben Ebene der Hierarchie auszumachen ist der Tree Browser sehr gut geeignet. Die Orientierung über mehrere Ebenen hinweg bzw. bei sehr tiefen Hierarchien ist jedoch manchmal schwierig, da der Baumcharakter nicht mehr auf einem Blick erfassbar ist.

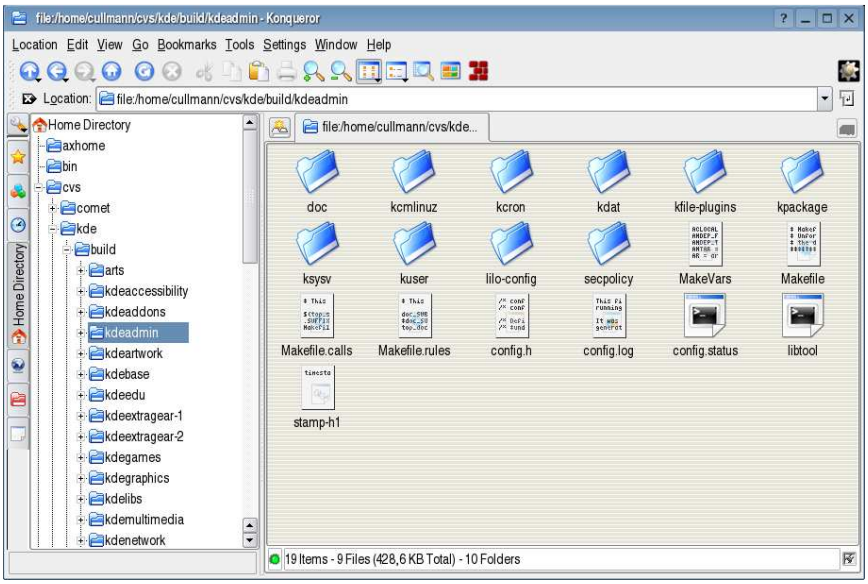

Abbildung 19: KDE Konqueror (Konqueror)

2.2.2.2.1.4 Treemap

Treemaps zählen zu den impliziten Darstellungsmethoden von Hierarchien in der Ebene, sind also eine sogenannte *Space Filling Methode*. Sie verwenden den komplett zur Verfügung stehenden Raum für die Darstellung der Hierarchie. Es wird alles (Struktur und Inhalt) in einem einzigen Panel visualisiert. Jeder Knoten wird als Rechteck dargestellt wobei die Kinder innerhalb der Eltern-Knoten angezeigt werden. Die Größe der Rechtecke ist abhängig vom Gewicht des jeweiligen Knotens. Das Gewicht wird wiederum von gewissen Attributen wie z.B. dem Erstellungsdatum oder der Dateigröße ermittelt (Putz, 2005).

Der ursprüngliche Algorithmus, ein sogenannter *Slice and Dice*-Algorithmus, wurde 1991 von Johnson und Shneiderman beschrieben. Die Wurzel, welche sich über die ganze Darstellungsfläche erstreckt, wird auf alle Kinder, abhängig von deren Gewicht, in vertikale Schlitze aufgeteilt. Diese wiederum werden auf deren Kinder, wieder abhängig von deren Gewicht, in horizontale Schlitze geteilt. Dies läuft rekursiv solange weiter, bis alle Knoten gezeichnet sind. Je mehr Knoten die Struktur hat, desto kleiner werden die Rechtecke. Dies führt oft dazu, dass die Flächen schwer zu erkennen sind. Diesem Problem wurde durch die *Squarified Treemaps* Methode versucht zuvorzukommen. Die Aufteilung der Knoten-

flächen erfolgt hier nicht mehr horizontal oder vertikal, sondern einer Kombination daraus, um die Flächen quadratischer zu gestalten. Dadurch ging allerdings die Ordnung der Struktur etwas verloren, wodurch einige Verbesserungen vorgenommen wurden, um etwas geordnetere *Maps* zu erhalten (Nussbaumer, 2005).

Treemaps erlauben es dem Benutzer schnell die größer gewichteten Elemente zu erkennen. Dem Benutzer wird auch gleich ein erster Überblick über die Struktur vermittelt. Intuitiv sind sie jedoch schwer erfassbar. Bei großen Hierarchien wird es auch immer schwieriger die einzelnen Elemente zu erkennen.

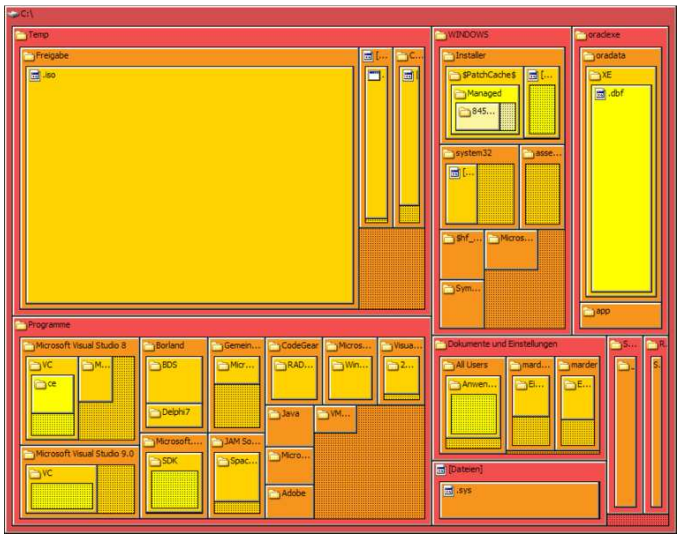

Abbildung 20: Darstellung eines Dateisystems als Treemap (Tree_structure - Wikipedia, 2011)

2.2.2.2.1.5 Hyperbolischer Browser

Der Hyperbolische Browser ist eine radiale und explizite Darstellungsmethode. Bei dieser Form wird versucht, sowohl den Gesamtkontext indem sich ein Knoten befindet, als auch die unmittelbaren Nachbarn mit Details darzustellen. Es wird immer der gesamte Baum dargestellt. Um nun die Integration der Gesamtübersicht und des Zooms in einer Ansicht zu bewältigen, wird eine hyperbolische Fläche verwendet. Im Zentrum dieser befindet sich der Fokusbereich, in dem die Knoten sehr detailliert dargestellt werden. Der Detailgrad nimmt nach außen hin zunehmend ab, die

Knotendichte wird hingegen größer. Dieser Effekt ist auch als Fisheye-Effekt bekannt (Winkler, 2006).

In Usability Tests fand der Hyperbolische Browser große Zustimmung im Hinblick auf effizientes Auffinden von Daten. Kleine Hierarchien waren allerdings mit einem Tree Browser schneller zugänglich (Andrews & Kasanicka, 2007).

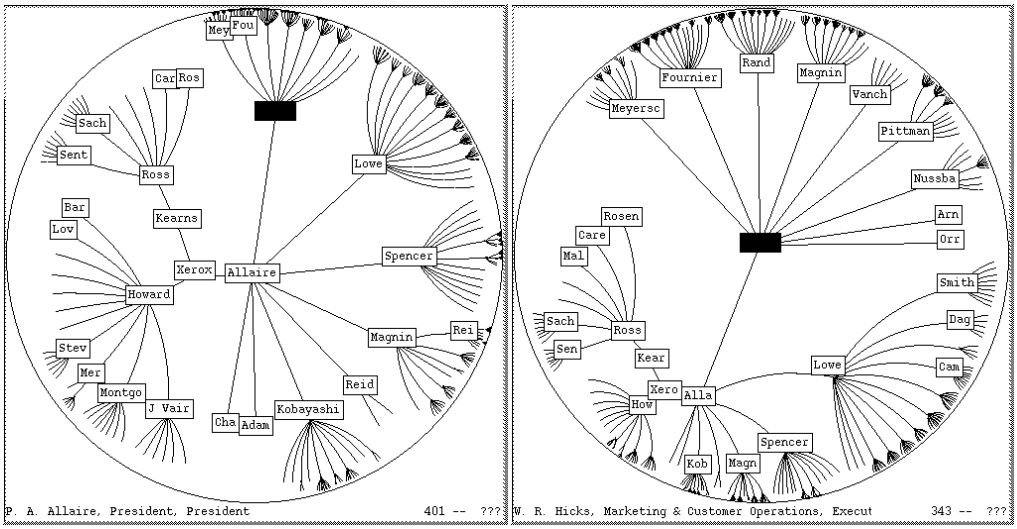

Abbildung 21: Hyperbolischer Browser: Änderung des Fokusbereichs (Xerox & Ramana, 1996)

2.2.2.2.1.6 Magic Eye View

Diese Technik ist ein anderer Ansatz des Versuches die Gesamtübersicht und den Zoom für höheren Detailgrad in einer Ansicht darzustellen. Zuerst wird die Hierarchie mit einem Algorithmus, ähnlich dem von Walker für die klassische Baumansicht, ausgelegt. Danach wird die Struktur auf die Oberfläche einer Halbkugel projiziert indem die kartesischen Koordinaten jedes einzelnen Knotens auf Kugel-Koordinaten abgebildet werden. Der verfügbare Platz auf der Hemisphäre nimmt Richtung *Äquator* zu wodurch der Platz für die Kindknoten mit zunehmender Tiefe steigt, da die Wurzel am *Pol* liegt, wo anfangs auch der Fokusbereich liegt, also im ursprünglichen Projektionsmittelpunkt. Wird dieser Projektionsmittelpunkt nun verschoben tritt der Effekt auf, dass die Knoten entgegen der Verschiebungsrichtung weiter auseinander und alle anderen näher zusammenrücken.

Somit wird ein neuer Fokusbereich für höheren Detailgrad festgelegt (Nussbaumer, 2005).

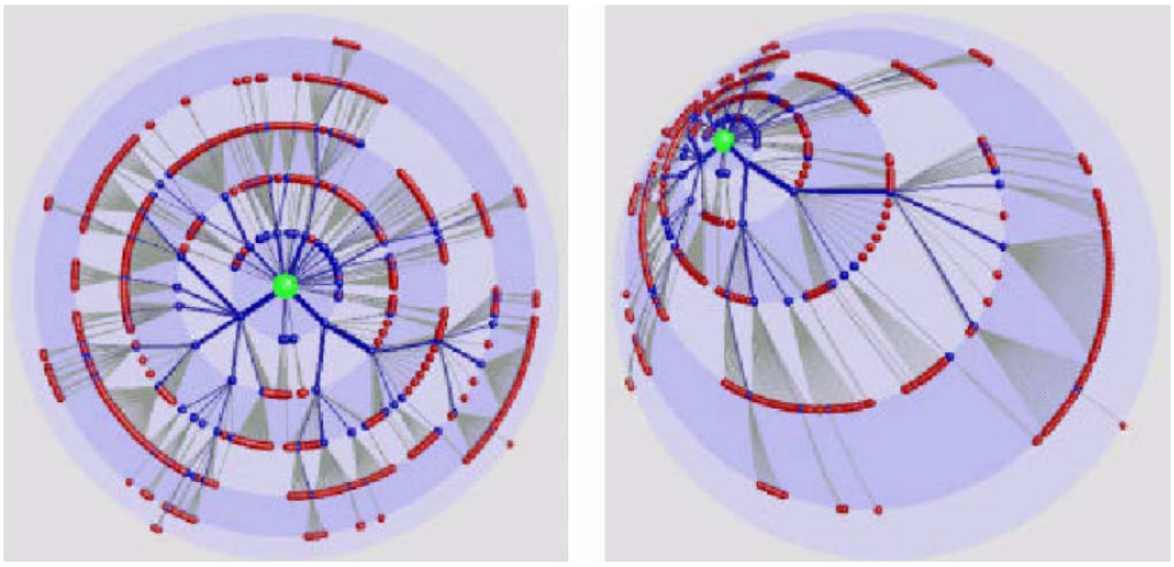

Abbildung 22: Magic Eye View: Änderung des Fokusbereiches (Zornow, 2004)

2.2.2.2.1.7 Cheops

Das aus einem französischen Projekt hervorgegangene System ermöglicht es mit sehr großen komplexen Hierarchien zu arbeiten. Hierbei werden die Knoten als Dreiecke dargestellt so wie der ganze Baum eine dreieckige Struktur darstellt, welche ähnlich wie der klassische Baum von oben nach unten verläuft. Einzelne Kindknoten können sich gegenseitig überlappen, wodurch die komprimierte Darstellung großer Hierarchien erst ermöglicht wird. Es muss gerade nur so viel von einem Dreieck sichtbar sein, dass die Existenz aller Knoten sichtbar ist. Dieser Umstand und dass die Auswahl eines Knotens nur im top-down Prinzip möglich ist, macht die Navigation innerhalb eines Cheops-Systems recht schwierig. Die Verwendung eines Farbschemas hilft beim Identifizieren des ausgewählten Knotens, dessen Kindern und Vorfahren (Egger, 2004).

In Abbildung 23 ist ein Cheops inklusive einer Legende des Farbschemas abgebildet.

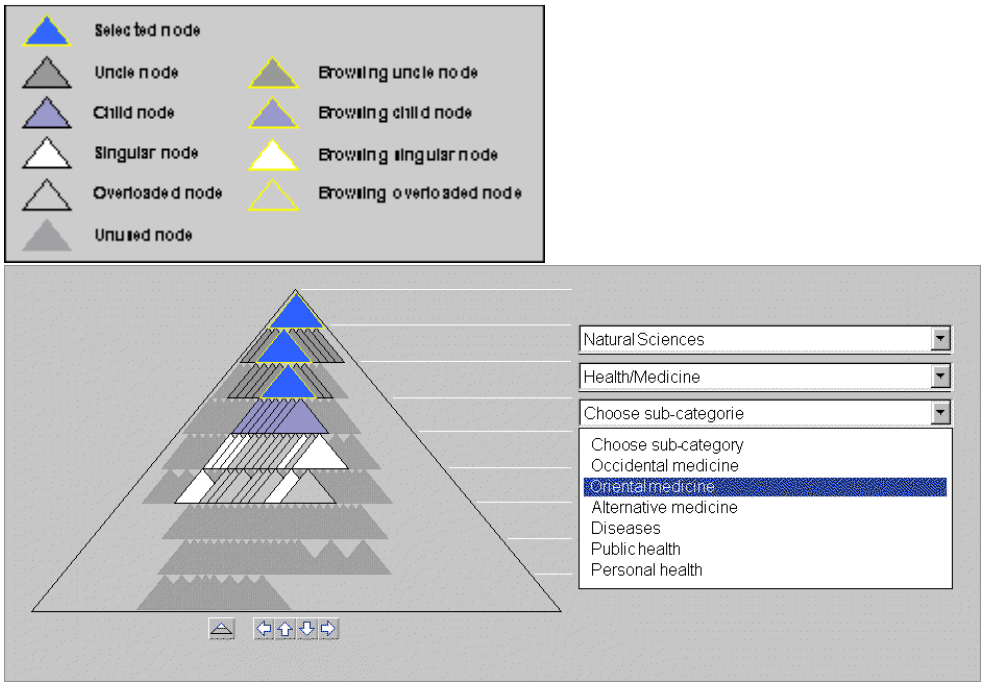

Abbildung 23: Cheops mit zugehöriger Legende (Beaudoin, Parent, & Vroomen, 1996)

2.2.2.2.1.8 InterRing

InterRing ist eine radiale, implizite Methode. Hierbei werden ausgehend von der Wurzel, welche im Zentrum liegt, kreisförmig alle Ebenen der Hierarchie aufgetragen. Jede Ebene der Hierarchie wird als neuer konzentrischer Kreisring dargestellt. Je tiefer ein Knoten in der Hierarchie liegt, desto weiter ist er vom Zentrum entfernt. Jeder Knoten einer Ebene erhält ein Segment im Kreisring. Kinder dieses Knotens grenzen im nächsten Kreisring diesem Segment an, damit die Struktur der Hierarchie erfassbar ist. Die Größe eines Segmentes hängt vom Verhältnis der Summe des Gewichtes eines Knotens und all seiner Kinder zur Summe der Gewichte aller Knoten der Struktur ab.

Das Problem hier ist, dass sehr tiefe Bäume zu schmalen Kreisringen führen und andererseits breite Bäume in zu klein aufspannenden Winkeln der Knotensegmente enden. Dafür werden dem Benutzer diverse Fokus und Kontext Methoden zur Verfügung gestellt. Diese können in einem weiteren Fenster oder manchmal sogar innerhalb oder außerhalb der Gesamtansicht dargestellt werden (Egger, 2004).

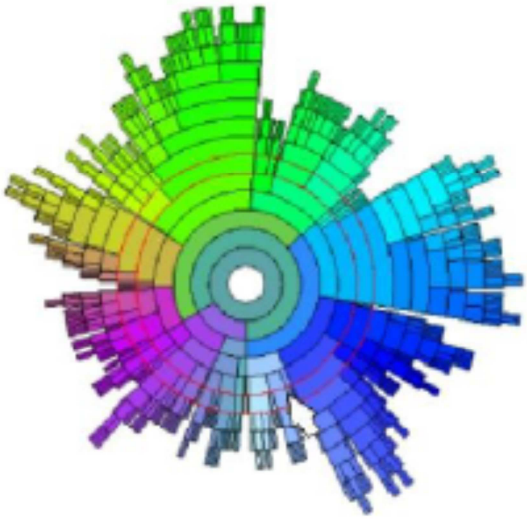

Abbildung 24: InterRing (Kerren, 2011)

2.2.2.2.1.9 Cone Trees

Der Cone Tree ist eine 3-dimensionale, explizite Darstellungstechnik. Im Grunde wurde hier der klassische Baum weiterentwickelt und auf 3D adaptiert. Ausgehend von der Wurzel werden die Kinder kreisrund unterhalb dargestellt. Dadurch erscheint der Baum insgesamt, wie auch alle Unterbäume kegelförmig. Die Knoten werden transparent dargestellt, damit die im Hintergrund befindlichen Verzeichnisse auch erfasst werden können. Wird ein Knoten angeklickt, drehen sich alle Ebenen von der Wurzel bis zum ausgewählten Knoten solange bis alle Knoten am Pfad im Vordergrund sind. Dadurch bleibt der Gesamtkontext immer im Blick. Die Höhe der Kegel hängt von der Tiefe des Baumes ab. Diese sind alle gleich hoch und ergeben sich aus der Gesamthöhe geteilt durch die Anzahl der Ebenen der Hierarchie (Winkler, 2006).

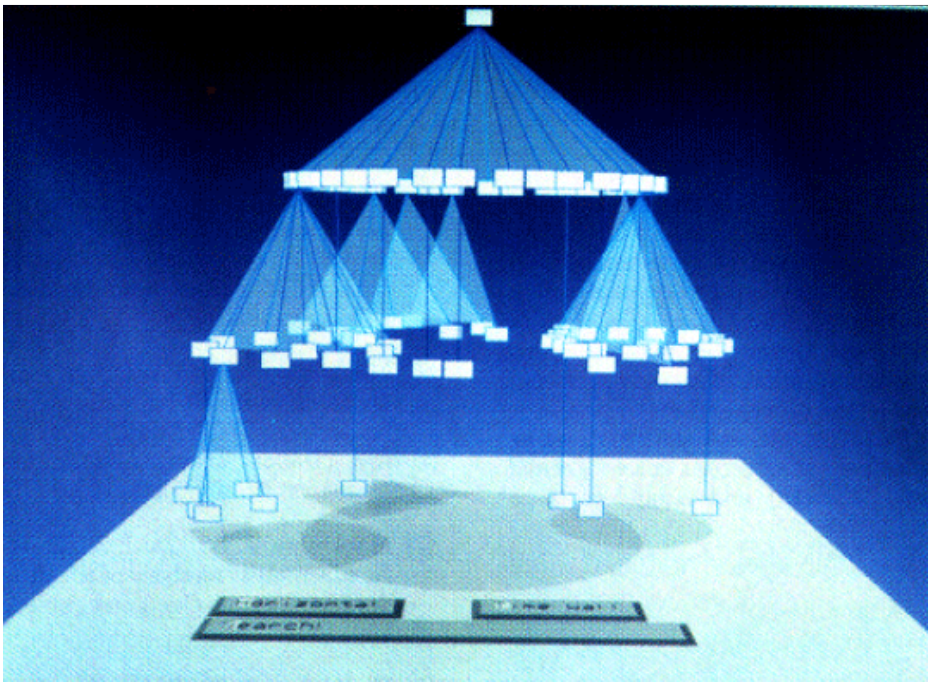

Abbildung 25: Cone Tree (Waloszek, 2008)

2.2.2.2.1.10 Beam Tree

Der Beam Tree ist eine 3-dimensionale implizite Darstellungsart. Sie ist im Grunde eine 3-dimensionale Weiterentwicklung der Treemap. Der Nachteil bei Treemaps ist, dass eigentlich nur die Blätter sichtbar sind. Bei diesem Ansatz werden einfach die Rechtecke, welche die Knoten darstellen, skaliert. Die Höhe oder die Breite der Rechtecke, abhängig von der Ebene, wird verkleinert, sodass die darunterliegende Struktur auch zum Vorschein kommt. Dadurch erhält man eine Tiefeninformation, welche zur 3D Visualisierung verwendet wird, wo die Knoten als Röhren (Beams) dargestellt werden (Knecht & Schwärzler, 2006).

In Abbildung 26 ist eine skalierte Treemap und in Abbildung 27 der daraus hervorgehende Beam Tree einer Hierarchischen Struktur zu sehen.

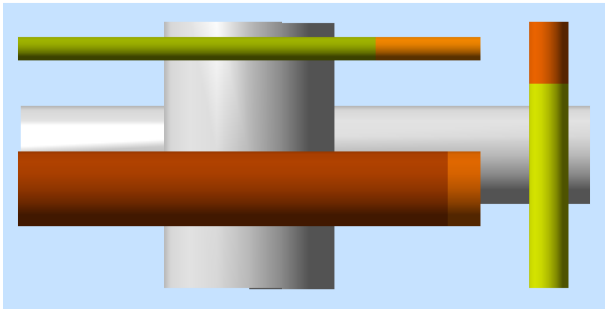

Abbildung 26: Skalierte Treemap – Beam Tree in 2D

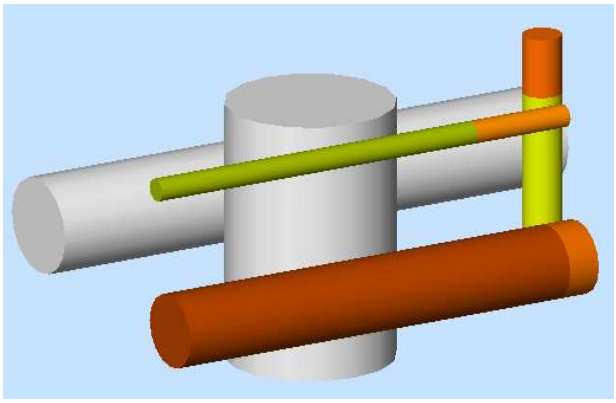

Abbildung 27: Beam Tree in 3D

2.2.2.2.1.11 Information Cube

Diese implizite Methode ist eine andere Adaptierung der Treemap in die dritte Dimension. Die Knoten werden als transparente Quader dargestellt. Die Größe ist von der Anzahl der Kinder abhängig. Die Kinderknoten werden als Quader innerhalb des *Vaterquaders* dargestellt. Um alle Quader herum ist der Wurzelknoten visualisiert. Auf die einzelnen Quader können zur Identifikation Labels gesetzt werden. Das große Problem bei dieser Art der Visualisierung ist die Unübersichtlichkeit bei großen Hierarchien (Egger, 2004).

Abbildung 28: Cone Tree (Egger, 2004)

2.2.2.2.1.12 Botanische Bäume

Diese Visualisierung von Hierarchien wurde von Kleiberg an der TU Eindhoven entwickelt. Sie verfolgt hauptsächlich die ästhetische Darstellung von Hierarchien und weniger die Interaktion mit der Hierarchie. Der Stamm des Baumes fungiert als Wurzelknoten, an dem jedes Kind, welches selbst Kinder besitzt, als abzweigender Ast dargestellt wird. Jedes Kind welches keine Kinder besitzt, wird am Ende des Stammes oder des Astes auf einer Kugel als Kegel dargestellt. In Abbildung 29 ist ein Botanischer Baum dargestellt (Egger, 2004).

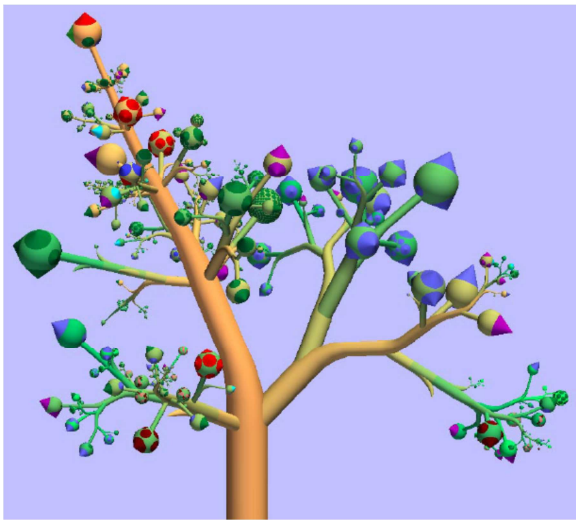

Abbildung 29: Botanischer Baum (Kleiberg, Wetering, & Wijk, 2001)

2.2.2.2.2 Auswahl der geeigneten Visualisierungsmethode der Dateienstruktur für das Austria-Forum

Da es sich beim Austria-Forum um eine webbasierte Enzyklopädie handelt, sollte eine Visualisierungsmethode verwendet werden, welche möglichst wenig Datenverkehr verursacht. Auch rechenintensive Methoden sollten vermieden werden, um den Server vor einer möglichen Überlastung zu bewahren. Weiters ist die Leistungsfähigkeit der clientseitigen Rechner auch ungewiss und könnte bei älteren Modellen zu langen Wartezeiten führen. Aus diesem Grund können alle 3D basierenden Techniken als auch rechenintensive Varianten, wie der Hyperbolische Browser, von vorn herein ausgeschlossen werden.

Als letzter wichtiger Aspekt ist die Platzeffizienz zu berücksichtigen. Es steht nur ein kleiner Anteil der Arbeitsfläche am linken Rand zur Verfügung. Um einen Vergleich der Platzverteilung der noch verbliebenen, auf 2D basierten Methoden der Hierarchievisualisierung zu bekommen, sind in den nächsten 3 Abbildungen die bereits vorgestellten Methoden des klassischen Baumes (siehe Abbildung 30), einer Treemap (siehe Abbildung 31) und des Tree Browsers (siehe Abbildung 32) für ein und dieselbe Dateienstruktur auf gleich großer Darstellungsfläche visualisiert.

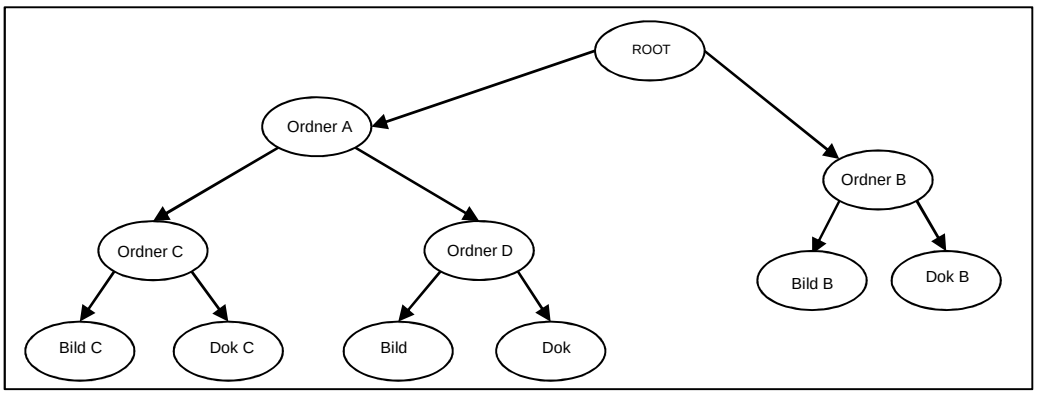

Abbildung 30: Darstellung als klassischer Baum

Abbildung 31: Darstellung als Treemap mit Gewichtung auf die Größe der Knoten

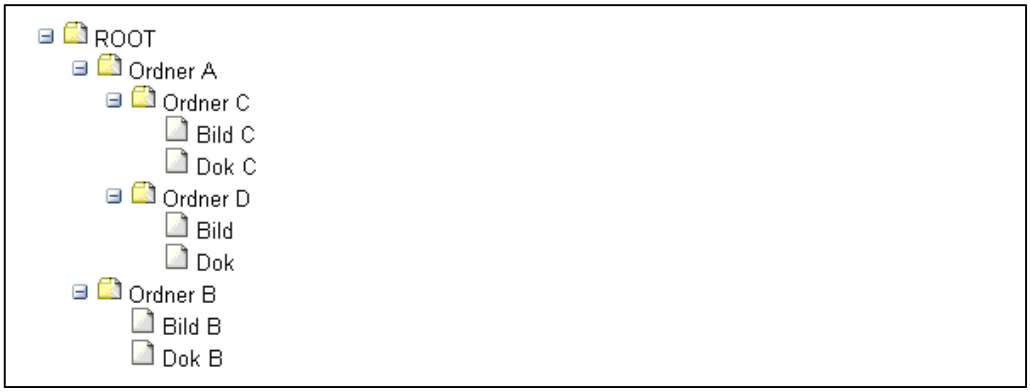

Abbildung 32: Darstellung als Tree Browser

Es ist zu erkennen, dass die Struktur der Hierarchie als klassischer Baum am schnellsten erfassbar ist. Solang sich die Anzahl der Knoten in Grenzen hält, wie in diesem Beispiel, ist dagegen nichts einzuwenden. Bei größeren Strukturen muss mit *Panning* und *Zooming* gearbeitet werden, um den Überblick zu bewahren. Diese Funktionen erfordern jedoch wieder Rechenleistung und Datentransfervolumen bei Webanwendungen.

Die Treemap verschafft einen schnell Überblick über die Gewichtung (in diesem Fall die Größe) der verschiedenen Ordner und Dateien. Durch die Einfärbung der Ebenen in einem Farbton, ist eine Übersicht über die Topologie der Hierarchie auch möglich, obwohl dazu schon etwas Übung notwendig ist. Wie beim klassischen Baum ist auch hier die Menge der Knoten ein wichtiger Faktor für die Lesbarkeit der Visualisierung. In Abbildung 20 ist zu sehen, dass bei steigender Anzahl der Knoten, die Recht-

ecke für deren Darstellung mitunter sehr klein ausfallen und diese daher leicht zu übersehen sind.

Durch die listenähnliche Darstellung findet der Tree Browser einen guten Konsens zwischen Lesbarkeit und Platzsparsamkeit. Wie in Abbildung 32 zu sehen ist, ist die Struktur trotz des geringen Platzbedarfes gut überschaubar. Größere Strukturen erfordern zwar mehr kognitive und interaktive Leistung vom User durch das Scrollen oder Auf- und Zuklappen der Unterverzeichnisse. Trotzdem ist diese Form noch die beste Lösung zur Visualisierung von Hierarchien, ganz besonders dann, wenn nur ein Teil am Rand der Darstellungsebene dafür zur Verfügung steht, wie es im Falle des Austria-Forums gegeben ist.

3. Wiki-Systeme

Wiki-Systeme zählen heute zu einem wesentlichen Bestandteil der Web-basierten Informationstechnologie. Viele Leute assoziieren heutzutage die Online-Enzyklopädie Wikipedia mit dem Begriff *Wiki*. Wikipedia ist zwar eine der meistverwendeten Enzyklopädien, welche auf dem System von MediaWiki basiert, es ist jedoch bei weitem nicht das einzige Wiki-System.

Eines dieser Wiki-Platformen ist JSPWiki, auf welchem das neue Austria-Forum basiert. In diesem Kapitel möchte ich grundlegend auf Wiki und einige bekannte Wiki-Systeme eingehen.

3.1 Was ist Wiki

Im Prinzip handelt es sich bei sogenannten Wikis um *Content Management Systeme* kurz *CMS*. Solche Systeme ermöglichen es dem Benutzer Inhalte zu verändern. Das Wort *Wiki* wiederum kommt aus dem hawaiianischen und bedeutet schnell oder sich beeilen (Hawaiian Dictionaries, 2003). Diese Definitionen lassen somit schon eine Beschreibung von Wikis zu, nämlich Web-Inhalte schnell und einfach zu produzieren, um sie so ebenfalls schnell einer großen Menge von Benutzern verfügbar zu machen. Als wesentlicher Unterschied zu anderen Content Management Systemen bietet Wiki-Software weniger Gestaltungsmöglichkeiten für Layout und Design der Webseiten. Wikis sind einfacher gehalten und sind dadurch auch für Neulinge leicht erlernbar (Wiki - Wikipedia, 2011).

Verschiedene Content Management Systeme können als Vorläufer der heutigen Wikis angesehen werden. Eines der ersten war das ZOG-Datenbanksystem, welches schon 1972 entstand. Dieses ermöglichte es mehreren Benutzern Textteile als Daten darzustellen und diese über Hyperlinks miteinander zu verbinden. Ab 1981 wurde es so erweitert, dass Änderungen einzelner Datensätze im gesamten Netzwerk sichtbar wurden. Das erste echte Wiki entwarf 1995 Ward Cunningham, welchem auch

der Name *Wiki* durch ein Erlebnis auf einem hawaiianischen Flughafen zu verdanken ist. Dort gibt es nämlich Schnellbusse, die WikiWiki-Bus genannt werden. Sein Wiki ging 1995 unter dem Namen *WikiWikiWeb* online (Semmernegg, 2009).

Der absolute Durchbruch der Wiki-Systeme gelang im Jahr 2001 durch die Einführung von Wikipedia, welche vom Unternehmer Jimmy Wales gemeinsam mit dem Philosophen Larry Sanger entwickelt wurde (Sen & Krömer, 2008).

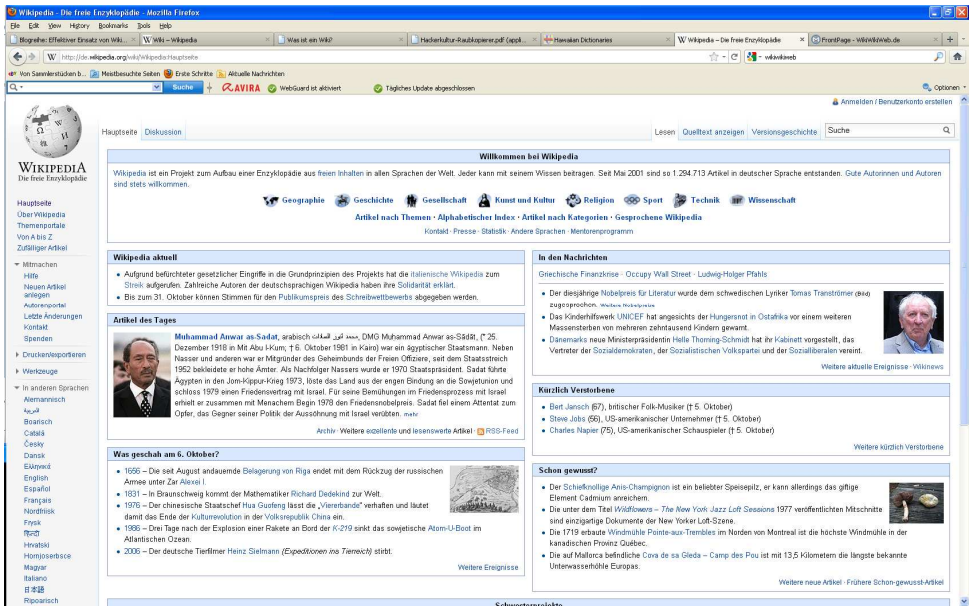

Abbildung 33: Startseite von Wikipedia[1]

Der Einsatz eines Wiki-Systems ist frei wählbar. Es kann am eigenen PC, als Desktop-Wiki und im Inter- sowie im Intranet eingesetzt werden. Hier kann es z.B. als Notizblock für einen Nutzer am PC oder als Kommunikations-Plattform für ein Projektteam unabhängig von Ort und Zeit oder als Informations-Plattform für Kunden oder Mitarbeiter eines Unternehmens oder ganz einfach als Enzyklopädie verwendet werden.

[1] (http://de.wikipedia.org/wiki/Wikipedia:Hauptseite) [Zugriff: 6.10.2011]

3.2 Das Wiki Konzept

WikiWikiWeb ermöglichte es schon jedem Besucher ohne Zugangsbeschränkungen beliebige Artikel zu erstellen, zu ändern oder zu löschen. Des Weiteren war die Syntax für jeden leicht erlernbar, sodass es für beinahe jedermann möglich war Beiträge zu verfassen. Dies sind auch die Grundkonzepte von Wikis. Cunninghams Meinung war es, das es keine Kontrollinstanz braucht, denn wenn es nur genug Benutzer gibt, die an einer Aufgabe mitwirken, würde sich alles selbständig regeln. Auch das Prinzip von Wikipedia basiert auf Selbstregulation. Jeder Benutzer kann beliebig Artikel ändern, jede Änderung wird in einer Art Historie festgehalten. Wenn ein Nutzer der Arbeit des anderen nicht zustimmt, kann er die alte Version wiederherstellen oder sie nach eigenem Ermessen verbessern (Sen & Krömer, 2008).

Diesem Konzept wurden anfangs von Kritikern anarchistische und chaotische Zustände bescheinigt. Aus Wikipedia entwickelte sich hingegen die größte Enzyklopädie der Welt. Waren es im Jahr 2001 noch knapp 5000 Artikel auf Englisch, hat Wikipedia mittlerweile mehr als 10 Mio. Artikel in mehr als 100 Sprachen. Auch große Zeitungen zitieren mittlerweile aus Wikipedia. Die Qualität der Artikel ist, entgegen der Meinung vieler Kritiker, qualitativ hochwertig. Mehr als 100.000 registrierte Benutzer sowie unzählige anonyme Mitgestalter und Administratoren pflegen die Enzyklopädie und erstellen deren Inhalte. Offensichtlicher Unfug oder sonstige Fehler werden dadurch entdeckt und korrigiert oder entfernt. Bei strittigen Themen kann sich eine Meinungsverschiedenheit entfachen, verschiedene Ansichten prallen aufeinander und es wird solange formuliert, bis am Ende pures Faktenwissen überlebt. Dies trägt maßgeblich zur kollektiven Intelligenz bei. Nebenbei sind die Texte online und somit ist Wikipedia äußerst aktuell und macht es zu einer ernsthaften Konkurrenz für bisher kommerziell verfügbare Enzyklopädien. All diese Fakten tragen wesentlich zur Bestätigung der Philosophie von Ward Cunningham bei (Sen & Krömer, 2008).

3.3 Architektur von Wikis

Auch hier lässt sich gleich vermuten, dass die Architektur, entsprechend der Wiki-Philosophie, schlank und einfach gehalten ist.

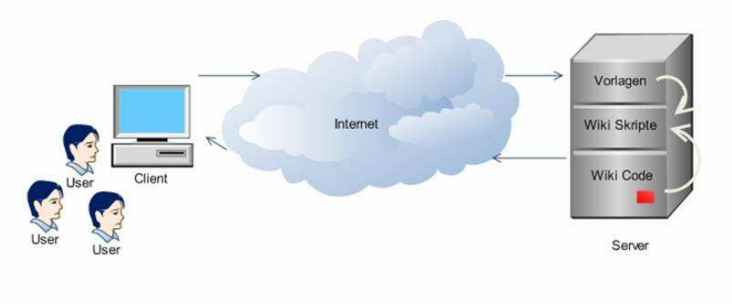

Abbildung 34: Darstellung der Wiki Architektur (Obertaxer, 2009)

Auf einem Server befindet sich eine Wiki-Software. Wird nun durch einen Benutzer ein Artikel aufgerufen, wird mittels Wiki-Skript der Wiki-Code aus einer Datenbank oder eines normalen Text-Dokuments, welches lokal am Server abgelegt ist, in HTML umgewandelt und in die Webpage eingebettet, sodass der Beitrag am Browser des Users dargestellt werden kann. Diese Skripte können z.B. PHP-, Perl- oder JSP-Skripte sein. Will nun jemand einen vorhanden Text bearbeiten, wird durch den Bearbeiten-Button eine neue Anfrage an den Server geschickt. Dieser sendet die angeforderte Seite nochmals, jedoch ohne den Artikel in HTML zu konvertieren. Somit erscheint clientseitig der Rohtext, also der Text dargestellt in der jeweiligen Wiki-Syntax, welcher dann bearbeitet werden kann. Nach dem Speichern wird die Änderung vom Server übernommen und der aktualisierte Text steht wieder zum Lesen oder Bearbeiten bereit. Weiters werden durch eine eigene Versionskontrolle auch die Änderungen gespeichert, sodass man den Artikel jederzeit auf eine Vorgängerversion zurücksetzen kann (Obertaxer, 2009).

3.4 Typische Wiki-Funktionen

Wiki-Systeme zeichnen sich auch durch typische Funktionen aus. Diese sind unter anderem die *Wiki-Syntax*, *Bearbeiten von Beiträgen*, *Versionie-*

rung oder History, *Differenzanzeige*, *Letzte Änderungen*, *Suchfunktion*, die *Übungsseite* bzw. *Sandbox*, *Upload*. Folgend möchte ich einige davon kurz beschreiben.

3.4.1 Syntax

Die Beitragserstellung in Wikis soll, wie bereits erwähnt, für so gut wie jedermann leicht zu bewältigen sein, deshalb bieten Wiki-Systeme eine relativ einfache Syntax an. Diese ist an HTML angelehnt und kann sich von System zu System leicht unterscheiden (Trattner a, 2009).

So würde z.B. der HTML-Tag für einen Hyperlink:

```
<a href="Seite">Seite</a>
```

in MediaWiki so ausschauen:

```
[[Seite]]
```

3.4.2 Bearbeiten von Beiträgen

Das Bearbeiten von Beiträgen ist wahrscheinlich die wichtigste Funktion von Wikis. Sie bietet meist eine Reihe von Elementen an, um neue Seiten zu erstellen oder zu verbessern. Bei den meisten Wikis findet sich auf der Editier-Seite ein Eingabefeld mit Funktionsknöpfen für die Wiki-Syntax, welche das Erstellen von Texten auch ohne großes Vorwissen ermöglichen (Trattner a, 2009).

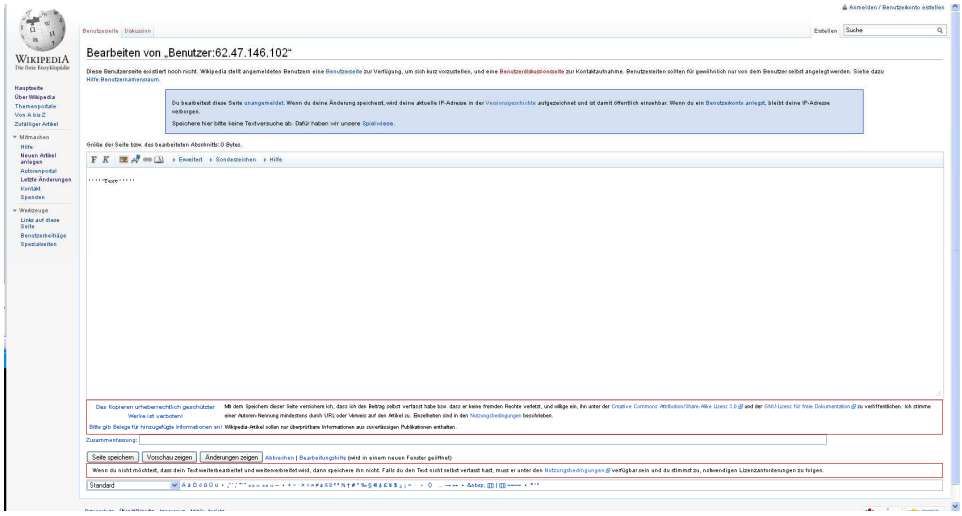

Abbildung 35: Die „Editierseite“ von Wikipedia

Sollte vor Erstellung des ersten Beitrages noch etwas Übungsbedarf bestehen, bieten fast alle Wikis eine Übungsseite, die sogenannte *Sandbox*, an.

3.4.3 Versionierung oder History

Diese Funktion ist wichtig um Änderungen an einer Seite rasch zu erkennen und gegebenenfalls Rückgängig zu machen. Meist reicht die Versionierung bis zum Erstellungsdatum des Beitrages zurück. Somit ist diese Funktion auch ein wichtiges Tool gegen Vandalen, die versuchen könnten die Seite komplett zu zerstören oder zu löschen (Trattner a, 2009).

3.4.4 Differenzanzeige

Um feststellen zu können, welche Details einer Seite von Version zu Version verändert wurden, bieten Wikis diese Funktion an. Sie bietet dem Benutzer die Möglichkeit durch farbliche Markierungen die Änderungen sofort zu erkennen. Diese Funktion ist gerade für Verfasser von Texten wichtig, um für sie interessante Änderungen oder ihren eigenen Werken feststellen zu können (Sayed, 2006).

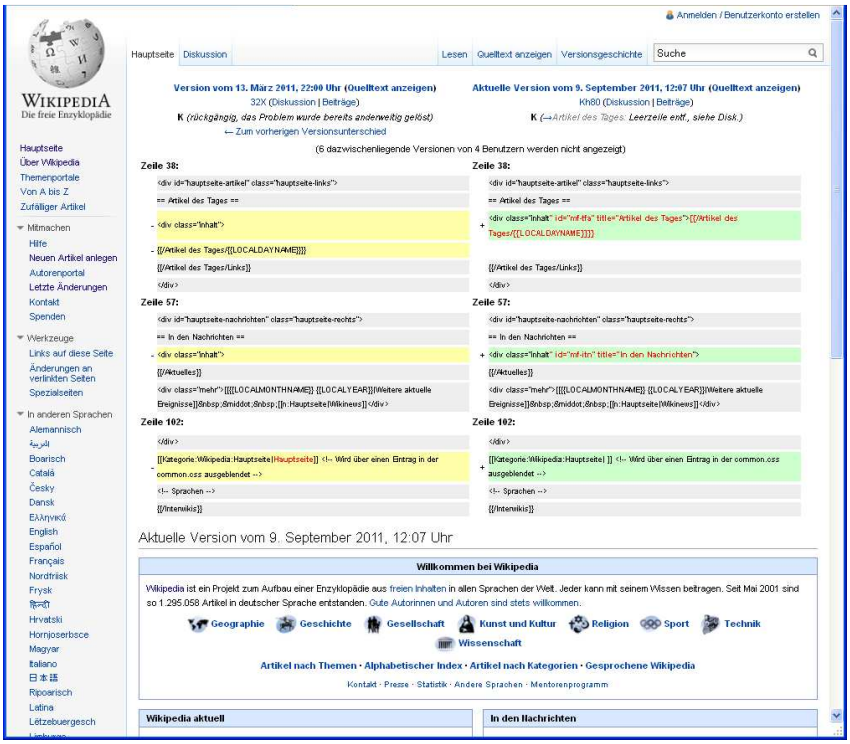

Abbildung 36: Anzeige der Unterschiede 2er Versionen in Wikipedia

3.4.5 Letzte Änderungen (Recent Changes)

Für Besucher, die sich regelmäßig in Wikis aufhalten, ist diese Funktion eine beliebte Anlaufstelle um zu sehen, was sich seit dem letzten Besuch geändert hat. Sie gibt einen chronologischen Überblick, wer wann welche Seite im Wiki geändert oder neu erstellt hat. Diese Funktion ist auch gegen Vandalismus sehr wertvoll, da der Administrator, oder sonst jemand aus der Community durch beobachten dieser Seite eventuell entstandenen Schaden gleich wieder zurücksetzen kann (Sayed, 2006).

3.4.6 Suchfunktion

Viele Wikis bieten eine klassische Volltext- oder Titelsuche für die enthaltenen Seiten an. Da die Anzahl der Beiträge eines Wikis ziemlich groß werden kann, ist diese Funktion sehr nützlich (Sayed, 2006).

3.4.7 Upload

Eine typische Funktion ist die Uploadfunktion. Damit können Beiträge durch zusätzliche Inhalte, wie Bildern, ergänzt werden. Weiters ist es dadurch möglich, das Wiki als Speichermedium für Dateien zu verwenden (Sayed, 2006).

3.4.8 Diskussion

Da gewisse Artikel heikle Themen aufarbeiten und es möglich ist, dass mehrere hundert User an diesem Artikel arbeiten, ist diese Funktion sehr nützlich, um eventuelle Kontroversen über das Thema aufzuarbeiten und zu diskutieren (Trattner a, 2009).

3.4.9 Verlinkung und Rückverweise

In Wikis kann jeder Beitrag auf einen anderen verweisen, sodass eine neue Hyperlinkstruktur entstehen kann. Die Verlinkung auf einen internen Artikel erfolgt meist über den Artikelnamen und nicht über dessen URL. Existiert ein interner Link von einer Seite auf eine andere, so ist es auch möglich, den Rückverweis herauszufinden. Mit dieser Funktion erhält man eine Liste aller Seiten innerhalb des Wikis, die auf eine spezielle Seite verweisen (Sayed, 2006).

3.4.10 Benutzerverwaltung

Ein Wiki ist vom Prinzip her für jeden offen. Daher spielt die Benutzerverwaltung nicht bei jedem Wiki eine Rolle. Trotzdem hat sich die Anmeldung mit Login-Funktion und Passwort durchgesetzt. Sollte ein System dies unterstützen, ist eine Registrierung notwendig, um bei der Erstellung bzw. Bearbeitung von Artikeln in diesem Wiki mitwirken zu können. Des Weiteren ist es auch möglich verschiedene Gruppen zu organisieren, denen verschiedene Rechte gewährt werden können (Sayed, 2006).

3.4.11 Daten Speicherung (Data Storage)

Die Speicherung und Verwaltung der Artikel und Beiträge ist primär für Administratoren des Wikis von Bedeutung, weniger für normale Benutzer. Hier gibt es zwei grundlegende Systeme, die Wikis verwenden:

- **Flat Files**

Dies sind einfache Text Dateien, welche einfach den Wiki-Text der Beiträge beinhalten. Sie werden im Verzeichnissystem am lokalen Speicher des Servers meist flach (in einem Verzeichnis ohne Unterverzeichnisse) abgelegt.

- **Relationale Datenbanken**

Hier werden Datenbanksysteme zum Verwalten der Beiträge verwendet, wie z.B. MySQL, PostgreSQL, Oracle, SQLite, BerkeleyDB usw.

Vorteile/Nachteile:

- Bei Flat Files ist keine zusätzliche Installation eines DB-Systems notwendig.
- Die Benützung von Flat Files erscheint viel natürlicher, da jeder Artikel eine einfache Datei im Dateisystem darstellt, welche einfach manipuliert und bearbeitet werden kann.
- Flat Files können einfach für Backups gezippt werden, dies ist ohne großes Vorwissen möglich. Backups sind zwar in Datenbanken auch kein großes Problem, jedoch nicht ohne gewisse Fertigkeiten, die man sich aneignen muss.
- Generelle Suchanfragen sollten in Datenbanken schneller abgearbeitet werden.
- Es kann auch erwartet werden, dass Datenbanken besser skalieren.

- Im Wesentlichen sind Datenbanksysteme schneller als das Arbeiten mit Dateien im Dateisystem.
- Nicht jeder Host lässt das Bearbeiten oder Manipulieren von Dateien aus Skripts zu, hier können dann nur DB-Systeme eingesetzt werden.

3.5 Wiki-Engines

Seit der Einführung des originalen WikiWikiWebs ist eine Großzahl verfügbarer Wiki-Implementierungen erschienen. Sie unterscheiden sich in ihrem Funktionsumfang, der Syntax und Semantik, der Speicherverwaltung oder des Installationsschwierigkeitsgrades voneinander. Die meisten sind dabei Open-Source Lösungen. Um nun das für sich persönlich richtige Wiki-Derivat zu finden, gibt es zahlreiche Hilfen im Internet wie z.B. die Wiki Matrix (Sayed, 2006).

Auf *Wiki Matrix* (www.wikimatrix.org) werden mehr als 100 verschiedene Wiki-Systeme gelistet und es bietet auch die Möglichkeit diese anhand verschiedener Features zu vergleichen. Somit können auch einige interessante statistische Aspekte abgerufen werden, z.B. die meist Aufgerufenen Wiki-Engines:

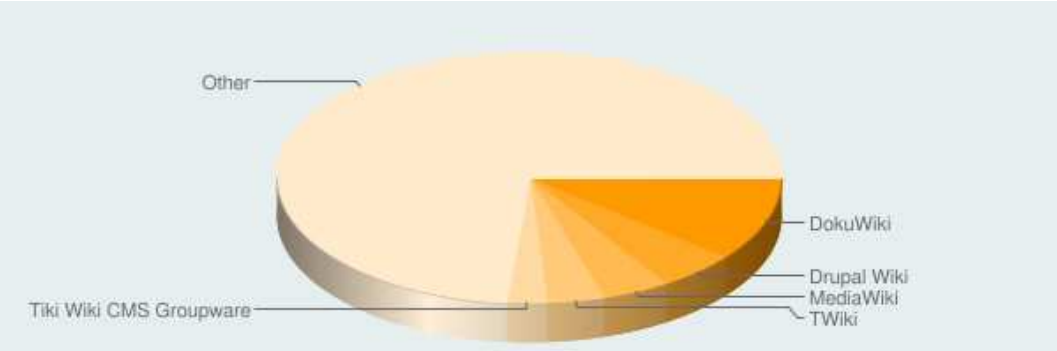

Abbildung 37: Meist aufgerufene Wikis auf wikimatrix.org[2]

[2] (http://www.wikimatrix.org/statistic/Most+Views) [Zugriff: 8.10.2011]

Wiki	Aufrufe
DokuWiki	6574
Drupal Wiki	3027
MediaWiki	2618
TWiki	2236
Tiki Wiki CMS Groupware	1550

Tabelle 2: Meist aufgerufene Wikis auf wikimatrix.org[3]

3.5.1 DokuWiki

DokuWiki ist ein standardkonformes, einfach zu verwendendes Wiki und zielt vor allem auf die Erstellung von Dokumentationen aller Art ab. Es richtet sich an Entwicklerteams, Arbeitsgruppen oder kleine Unternehmen. Es hat eine einfache und trotzdem mächtige Syntax, welches die Erstellung strukturierten Textes erleichtert. Daten werden als Text Dateien auf dem lokalen System abgelegt, daher ist auch keine Datenbank nötig.

Einige Features sind:

- Programmiert in PHP
- Daten Speicherung: Text Dateien
- Sicherheit: Page Permissions, Acces Control Lists (ACL)
- Grundlegende Eigenschaften: Vorschau, unlimitierte Seiten Revision, History, Differenzanzeige, Plugin-System, Letzte Änderungen
- Unterstützt ca. 55 Sprachen
- Unicode Unterstützung[4]

[3] (http://www.wikimatrix.org/statistic/Most+Views) [Zugriff: 8.10.2011]
[4] (http://www.wikimatrix.org/show/DokuWiki) [Zugriff: 10.10.2011]

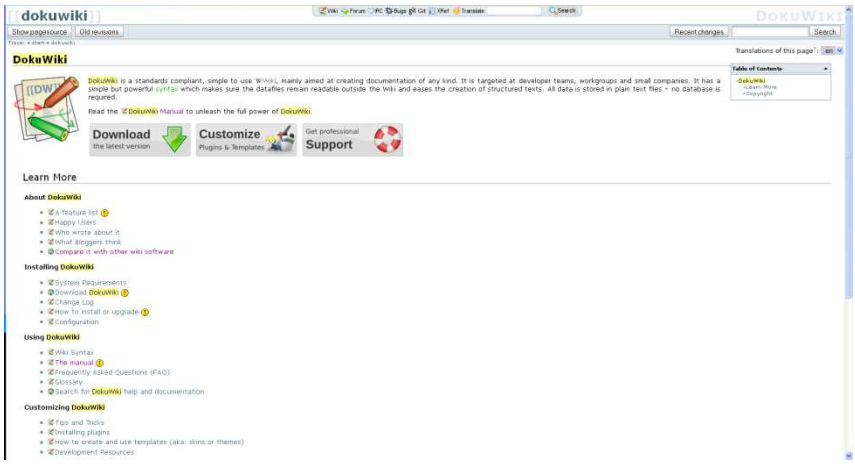

Abbildung 38: DokuWiki[5]

3.5.2 DrupalWiki

Drupal Wiki gehört zu den Wikis der sogenannten 3. Generation (3G). Es ist eine auf einem Web 2.0 Applikations-Framework basierende Engine und kombiniert *Enterprise 2.0 Funktionalität* und *Social Networking* mit den normalen kollaborativen Funktionen von Wikis. Es eignet sich besonders für technische Dokumentationen, SAP Integration von Daten, Qualitätsmanagement, Projektmanagement, Wissensdatenbanken, Blogs usw.

Einige Features sind:

- Programmiert in PHP
- Nicht frei verfügbar
- Daten Speicherung: MySQL, PostgreSQL
- Sicherheit: Page Permissions, ACL
- Grundlegende Eigenschaften: Vorschau, unlimitierte Seiten Revision, History, Differenzanzeige, Plugin-System, Letzte Änderungen
- Unterstützt ca. 12 Sprachen
- Unicode Unterstützung
- Spezielle Eigenschaften: Unterstützt Unicode, Kategorien, Volltextsuche, Strukturierte Daten[6]

---

[5] (http://www.dokuwiki.org/dokuwiki) [Zugriff: 10.10.2011]
[6] (http://www.wikimatrix.org/show/Drupal-Wiki) [Zugriff: 10.10.2011]

Abbildung 39: DrupalWiki[7]

3.5.3 MediaWiki

MediaWiki wird von Wikipedia genützt und ist deswegen die wahrscheinlich populärste Wiki-Engine. Weiters wird es auch von WikiQuote, WikiBooks, WikiTravel, UncycloPedia, WikiCities genützt.

MediaWiki basiert auf PHP. MySQL dient als Datenbanksystem. Die Syntax ist stark an der von UseModWiki angelehnt. Der Name kommt von WikiMedia, der Organisation, die die Wikipedia-Projekte organisiert.[8]

Einige Features sind:

- Programmiert in PHP
- Daten Speicherung: MySQL, PostgreSQL, Oracle, SQLite
- Sicherheit: Page Permissions, ACL
- Grundlegende Eigenschaften: Vorschau, unlimitierte Seiten Revision, History, Differenzanzeige, Plugin-System, Letzte Änderungen
- Unterstützt ca. 140 Sprachen

[7] (http://drupal-wiki.com) [Zugriff: 13.10.2011]
[8] (http://c2.com/cgi/wiki?MediaWiki) [Zugriff: 7.10.2011]

- Unicode Unterstützung
- Spezielle Eigenschaften: Unterstützt Unicode, Kategorien, Volltextsuche[9]

Abbildung 40: MediaWiki[10]

3.5.4 TWiki

TWiki ist ein flexibles, leistungsstarkes und einfach zu bedienendes *Enterprise-Wiki.* Es kann als Dokumenten Management System, als Wissensdatenbank oder als Raum für Projekt Entwicklungen herangezogen werden. Entwickler können die Funktionalität durch Plugins erweitern. Es gibt bereits mehr als 400 frei verfügbare Erweiterungen, mit denen man sein Wiki nach Belieben gestalten kann.

Einige Features sind:

- Programmiert in Perl, JavaScript
- Daten Speicherung: Text Dateien
- Sicherheit: Page Permissions, ACL
- Grundlegende Eigenschaften: Vorschau, unlimitierte Seiten Revision, History, Differenzanzeige, Plugin-System, Letzte Änderungen
- Unicode Unterstützung

---

[9] (http://www.wikimatrix.org/show/MediaWiki) [Zugriff: 10.10.2011]
[10] (http://www.mediawiki.org/wiki/MediaWiki/de) [Zugriff: 13.10.2011]

- Unterstützt ca. 19 Sprachen
- Spezielle Eigenschaften: Unterstützt Unicode, Kategorien, Volltextsuche, Strukturierte Daten[11]

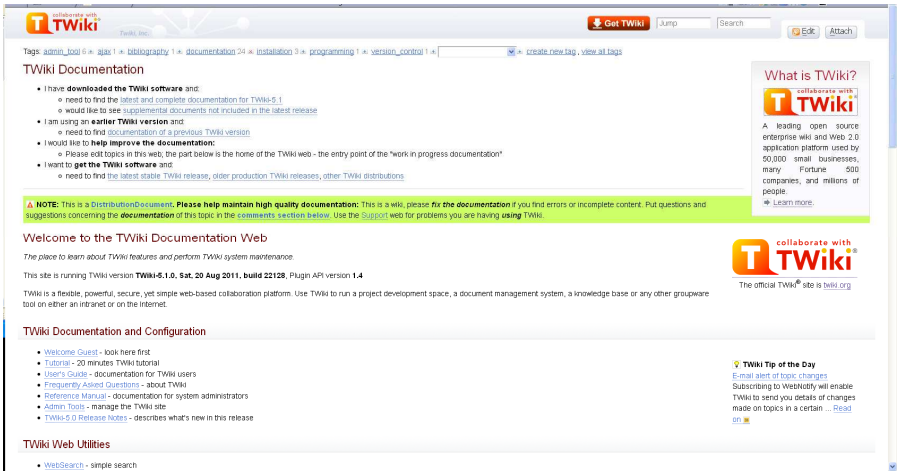

Abbildung 41: TWiki[12]

3.5.5 Tiki Wiki CMS Groupware

Wie der Name schon sagt, handelt es sich bei diesem Wiki um eine All-In-One-Lösung von Wiki + CMS + Groupware. Es können alle Arten von Webanwendungen, Wissensdatenbanken, Portale im Intra- sowie Internet erstellt werden.

Einige Features sind:

- Programmiert in PHP
- Daten Speicherung: MySQL
- Sicherheit: Page Permissions, ACL
- Grundlegende Eigenschaften: Vorschau, unlimitierte Seiten Revision, History, Differenzanzeige, Plugin-System, Letzte Änderungen
- Unicode Unterstützung
- Unterstützt ca. 19 Sprachen

[11] (http://www.wikimatrix.org/show/TWiki) [Zugriff: 10.10.2011]
[12] (http://twiki.org/cgi-bin/view/TWiki/WebHome) [Zugriff: 13.10.2011]

- Spezielle Eigenschaften: Unterstützt Unicode, Kategorien, Volltextsuche, Strukturierte Daten[13]

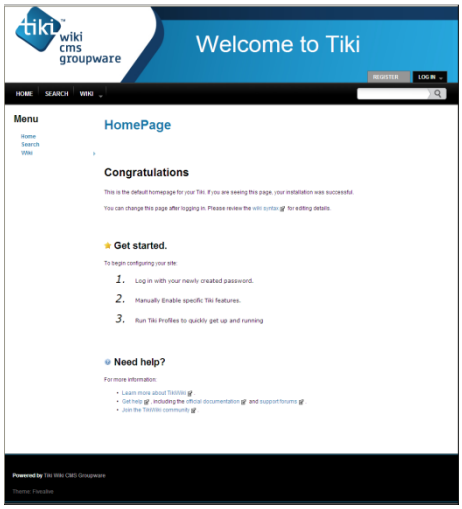

Abbildung 42: TikiWiki[14]

3.6 Vor- und Nachteile von Wiki-Systemen

Zusammenfassend sollen noch einige wesentliche Vor- und Nachteile von Wikis angeführt sein.

Die Vorteile die sie bieten, haben stark zur Verbreitung dieser Technologie beigetragen, einige davon sind (Semmernegg, 2009):

- Grundsätzlich kann jeder User Artikel verändern und verfassen.
- Es ist leicht erlern- und benutzbar
- Änderungen sind sofort ohne Verzögerung verfügbar
- Menschen können unabhängig von deren Aufenthaltsort am selben Dokument arbeiten.
- Durch die Versionierung der Beiträge, welche so gut wie alle Wikis unterstützen, ist es einfach, Änderungen zu verfolgen und gegebenenfalls rückgängig zu machen.

[13] (http://www.wikimatrix.org/show/Tiki-Wiki-CMS-Groupware) [Zugriff: 10.10.2011]
[14] (http://info.tiki.org/display71) [Zugriff: 13.10.2011]

- Es wird auch Menschen ohne HTML-Kenntnis oder sonstigen Internet-Techniken die Möglichkeit geboten, einfach im World Wide Web Artikel zu veröffentlichen.
- Wikis können für verschiedenste Themenbereiche eingesetzt werden, da es keine vordefinierte Struktur gibt.
- Es gibt eine Vielzahl von Open Source Projekten wodurch keine zusätzlichen Kosten entstehen.

Einige Vorteile von Wikis können auch als Nachteil ausgelegt werden:

- Da jeder Dokumente bearbeiten darf, sind Wikis für einige Anwendungen nicht geeignet, z.B. wenn Geheimhaltung wichtig ist. Zwar ist es möglich den Zugriff für gewisse Benutzer einzuschränken, jedoch liegt darauf nicht der primäre Fokus solcher Systeme.
- Wenn Wikis nicht verwaltet werden, ist die Gefahr von Vandalismus groß. Dies kann zwar durch Benutzerkonten etwas eingeschränkt werden, jedoch bleibt auch dann noch ein gewisses Maß an Verwaltungsaufwand bestehen.
- Ohne Internetanbindung kann man Wikis nicht nutzen. Die Möglichkeit den Inhalt in andere Formate umzuwandeln - wie z.B. pdf-Dateien - und so evtl. die gewünschten Artikel auch offline verfügbar zu machen oder auszudrucken, werden jedoch ständig verbessert.
- Da Wikis ohne Strukturvorgaben auskommen sind sie sehr flexibel, was jedoch auch zu unorganisiertem Wachstum führen kann, was schlussendlich in unübersichtlichen Dokumenten enden kann wo Informationen schwer auffindbar sind.

4. JSPWiki – Austria-Forum (Structured Wiki)

Wie einleitend schon geschrieben, wurde 1996 das österreichische Kulturinformationssystem AEIOU ins Leben gerufen. Im Jahr 2006 wurde dann am IICM ein *Content Management System* entwickelt, das den Inhalt von AEIOU übernehmen und unter dem Namen Austria-Forum weiterführen sollte. Aufgrund von technischen Mängeln stand im Jahr 2008 schließlich die Frage im Haus, dieses System entweder zu verbessern oder durch ein anderes System zu ersetzen. Nach mehreren Analysen wurde schlussendlich dieses System durch ein Wiki-System ersetzt, welches seit Oktober 2009 im Einsatz ist.

Das Austria-Forum ist ein Netzwerk Informationssystem mit einer großen Anzahl von Berichten über Österreich. Neue Beiträge sind einfach zu editieren, kontrollieren und zu veröffentlichen. Die Richtigkeit und Qualität der Artikel wird durch ausgewiesene Experten vom jeweiligen Wissensgebiet kontrolliert bevor sie eingefroren werden und am System abrufbar sind. Somit sind die Einträge zitierbar und können für Ausbildung, Forschung oder Journalismus verwendet werden. Der Inhalt bezieht sich immer auf Österreich, somit kann das Austria-Forum als eine österreichische online Enzyklopädie gesehen werden (Helic a, Saeed, & Trattner, 2009).

Welche Anforderungen wurden nun gestellt, dass die Entscheidung zur technischen Umsetzung des neuen Systems schlussendlich auf ein Wiki-System fiel (Trattner a, 2009):

- Das System soll frei verfügbar sein (OpenSource)
- Das System soll benutzerfreundlich sein
- Das System soll die Community als integralen Bestandteil ansehen
- Das System soll einfache Suchmethoden unterstützen (Volltext, Keyword)
- Das System soll einfache Methoden zur Content Erstellung anbieten (z.B.: WYSIWYG Editor, simple Syntax, Eingabe-Templates, etc.)

- Das System soll ein User Management und Control Modul anbieten
- Das System soll eine flexible Art der Kategorisierung anbieten
- Das System soll Drucken und Speichern von Beiträgen ermöglichen
- Das System soll ein leicht anzupassendes Datenspeicherungs-Modul anbieten
- Das System soll leicht erweiterbar sein

Nach Abarbeitung dieser Anforderungen fiel der Fokus schlussendlich auf die Wiki-Software *JSPWiki*, weshalb ich folgend einige technische Aspekte dieses Wiki-Clones erklären möchte.

4.1 JSPWiki

Janne Kalkonen startete die Entwicklung von JSPWiki im Jahre 2001. Das System ist in Java und JSP entwickelt und ist leicht zu installieren. Es hat die üblichen Wiki-Funktionen wie sie in Kapitel 3.4 beschrieben wurden und zeichnet sich durch folgende, für das Austria-Forum gut brauchbaren und erweiterbaren Features aus.

4.1.1 Plugins

Eine wesentliche Funktion der JSPWiki API ist die Plugin-Verwaltung. So kann die Eigenschaft und Unterstützung von JSPWiki nach eigenem Ermessen, vor allem funktional, erweitert werden, indem man seine eigenen Plugins schreibt oder bereits vorhandene verwendet. Diese sind einfache *Java Klassen,* welche das `com.ecyrd.jspwiki.plugin.WikiPlugin` Interface implementieren. Durch folgende Syntax können Plugins in eine Wiki-Seite eingefügt werden:[15]

```
[{INSERT <plugin class> WHERE<param1=value1>,<param2=value2>,...}]
```

[15] (http://www.jspwiki.org/wiki/JSPWikiPlugins) [Zugriff: 04.11.2011]

4.1.2 Filter

Seit JSPWiki 2.2 steht auch ein Filter-Modul zur Verfügung. Dieses bietet die Möglichkeit, die Informationen während des Flusses zwischen der Wiki-Engine und dem Browser automatisch vor bzw. nach zu bearbeiten.

Die vier Möglichkeiten diesen Filter zu setzen sind:

- Nach dem Holen der Seite vom Speicher, aber vor dem Übersetzen von WikiMarkup in HTML
- Nach dem Übersetzen in HTML, jedoch vor der Darstellung am Browser
- Vor dem Ablegen der Seite im Speicher
- Nach dem Ablegen der Seite im Speicher[16]

In JSPWiki vordefinierte Filter sind:

- *PingWeblogsComFilter*
  Dient zum filtern des Weblogs einer JSPWiki-Seite.[17]
- *ProfanityFilter*
  Hiermit können gewisse Inhalte von Seiten vor dem Anzeigen entfernt werden.[18]
- *SpamFilter*
  Damit ist es möglich fragwürdige Änderungen an Seiten zu blocken.[19]

4.1.3 Variablen

Mittels des Variablen-Moduls, können in JSPWiki bestimmte Werte oder Parameter auf einer Seite zur Anzeige gebracht werden. Hierbei kann in JSPWiki zwischen folgenden Variablen unterschieden werden:

---

[16] (http://doc.jspwiki.org/2.4/wiki/PageFilters) [Zugriff: 04.11.2011]
[17] (http://doc.jspwiki.org/2.4/wiki/PingWeblogsComFilter) [Zugriff: 04.11.2011]
[18] (http://doc.jspwiki.org/2.4/wiki/ProfanityFilter) [Zugriff: 04.11.2011]
[19] (http://doc.jspwiki.org/2.4/wiki/SpamFilter) [Zugriff: 04.11.2011]

- *Vordefinierte Variablen (oder auch Konstanten bzw. System Variablen)*
  Hiermit kann auf einer Seite z.B. der Name des Wikis, die Basis-URL des Wikis, der Login-Status, der Seitenname, die Gesamtanzahl der Seiten usw. zur Anzeige gebracht werden.
- *Kontext Variablen*
  Stehen mit anderen Anwendungen bzw. Plugins in Verbindung um diese zu Konfigurieren oder Werte auszulesen.
- *Property Variablen*
  Hiermit können Einstellungen aus dem *jspwiki.properties* File ausgelesen werden.
- *Page Variablen*
  Hiermit können eigene Variablen auf einer Seite definiert werden.[20]

4.1.4 Forms

Hiermit können Wiki-Seiten, wie aus HTML bekannt, mit Formular-Funktionen erweitert werden.
Folgende Elemente werden unterstützt:

- *FormOpen*
  Dient zum Öffnen einer JSPWiki-Form.
- *FormClose*
  Dient zum Schließen einer JSPWiki-Form.
- *FormSet*
  Dient zum Setzen gebräuchlicher Parameter für die Übermittlung (verborgen).
- *FormOutput*
  Hiermit kann der Händler (Plugin) für das Formular definiert werden.

[20] (http://doc.jspwiki.org/2.4/wiki/Wiki.How%20To.Use%20Variables) [Zugriff: 04.11.2011]

- *FormInput*
  Dient zur Eingabe von z.B. Text, Passwörtern usw.
- *FormSelect*
  Hiermit kann eine Drop-down Auswahlliste erstellt werden.
- *FormTextarea*
  Stellt ein mehrzeiliges Textfeld zur Verfügung.[21]

4.1.5 Templates

Um seine eigenen Layouts zu gestalten, unterstützt JSPWiki Templates. Templates sind ein Satz von HTML und JSP Dateien, welche das Aussehen der Wiki-Seiten definieren. Der standardmäßige Template-Ordner lautet *default*. Die selbst entwickelten Templates sollten in einen eigenem Ordner abgelegt werden. Durch Änderung des `jspwiki.templateDir` in der `jspwiki.properties` Datei auf den Ordner mit den persönlichen Templates steht JSPWiki nach Neustart dieses Aussehen zur Verfügung.[22]

4.1.6 Sicherheit

JSPWiki beinhaltet vielfältige und flexible Sicherheitseigenschaften. Dadurch ist JSPWiki für den Einsatz in *Stand-Alone-Systemen* oder als Teil eines großen Firmen-Intranets geeignet.

4.1.6.1 Authentifizierung

JSPWiki unterstützt mehrere Stufen der Authentifizierung. So können Benutzer anonym, mit so genannter *versicherter* Identität, durch *cookies* authentifiziert oder als Administrator gelten.

- *Anonym*
  Der Benutzer ist nicht eingeloggt und hat auch kein *cookie* bereitgestellt. Im Benutzerbereich wird nichts angezeigt.

---

[21] (http://doc.jspwiki.org/2.4/wiki/Wiki.Forms) [Zugriff: 04.11.2011]
[22] (http://www.jspwiki.org/wiki/JSPWikiTemplates) [Zugriff: 07.11.2011]

- *Versichert*

  Der Browser des Benutzers hat ein so genanntes `JSPWikiAssertedName` – *cookie*. Im Benutzerbereich wird folgendes angezeigt: Benutzername (nicht eingeloggt).
- *Authentifiziert*

  Der Benutzer ist mit seiner Identifikation und dem Passwort eingeloggt. Im Benutzerbereich wird folgendes angezeigt: Benutzername (authentifiziert). In dieser Stufe gibt es noch die Unterscheidung zwischen Administrator und normalen Benutzer.

4.1.6.2 Zugriffskontrolle

JSPWiki fördert wie alle Wikis die Offenheit und Zusammenarbeit. So erlaubt es grundsätzlich allen Benutzern Seiten anzusehen, zu erstellen und zu ändern. Dies ist jedoch nicht immer förderlich. So gibt es auch in JSPWiki die Möglichkeit die Zugriffsrechte für jede Seite gesondert einzustellen. Um dies zu regeln, werden entweder *Zugriffskontroll-Listen (ACLs - Access control lists)* oder für eine allgemeinere Zugriffskontrolle die JAVA2 Security Policy mit JAAS Konfigurationsdateien verwendet. In ACLs wird spezifiziert, welchen Benutzern oder Gruppen welche Aktionen erlaubt sind.

- *ACL*

  Diese erlauben es Benutzern, die Rechte einzelner Seiten zu regeln. Hierfür stehen fünf Privilegien zur Verfügung: *view*, *edit*, *comment*, *rename* und *delete*.

  Soll auf einer bestimmten Seite z.B. der Nutzer Hans das Recht erhalten die Seite zu ändern, ist folgende Syntax der Wiki-Seite hinzuzufügen: `[{ALLOW edit Hans}].`
- *JAAS*

  Um eine Vielzahl von Seiten zu verwalten, ist eine globale Administration manchmal sinnvoller als eine auf Pageebene. Hierfür wird dieses Modul zusätzlich angeboten. Zugriffsregeln der JAAS Konfi-

gurationsdateien haben höhere Priorität als solche auf Seitenebene.[23]

4.1.7 Datenspeicherung

Das für die Implementation des Ajax-Tree und die Hierarchisierung des Verzeichnissystem bedeutendste Feature, ist die Datenspeicherung. JSPWiki bietet dafür ein leicht zu änderndes bzw. anzupassendes Modul, welches mit ein Grund für die Auswahl dieses Systems als Basis des neuen Austria-Forums war. Auf Codeebene werden ausgehend vom Interface *WikiPageProvider,* alle Speicherungskonzepte abgeleitet (Trattner a, 2009).

Grundsätzlich kann in JSPWiki zwischen

- *Datei-Provider* und
- *Anhangs-Provider*

unterschieden werden. Wobei sich der Datei-Provider, wie der Name schon sagt, um die Verwaltung der Dateien der einzelnen Seiten kümmert und der Anhangs-Provider für die Verwaltung der Anhänge der jeweiligen Seiten zuständig ist. JSPWiki betrachtet die Anhänge nicht als globales Objekt wie z.B. in MediaWiki, sondern ordnet jeder Wiki-Seite ihre eigenen Anhänge zu (Trattner a, 2009).

JSPWiki speichert die Wiki-Seiten als reine Text-Dateien auf der Festplatte außerhalb des Servlet-Containers mit allen dazugehörigen Anhängen ab. Der Ordner hierfür kann in der `jspwiki.properties` Datei festgelegt werden. Für die Dateien-Verwaltung stehen standardmäßig drei Datei-Provider zur Verfügung:[24]

[23] (http://doc.jspwiki.org/2.4/wiki/Wiki.Admin.Security) [Zugriff: 07.11.2011]
[24] (http://doc.jspwiki.org/2.4/wiki/PageStorage) [Zugriff: 08.11.2011]

- *FileSystemProvider*

  Ist der einfachste der drei Provider. Hier werden die Dateien flach in einem Ordner ohne Versionsverwaltung gespeichert. Das heißt, ältere Dateien werden einfach überschrieben. Somit ist eine Revision zu einer älteren Version oder eine Differenzanzeige nicht möglich. Deshalb ist ein Einsatz dieses Providers nicht anzuraten, denn die Revisionskontrolle ist eines der wesentlichsten und wichtigsten Features eines modernen Wiki-Systems z.B. beim Schutz vor Zerstörung einzelner Wiki-Seiten.[25]

- *VersioningFileProvider*

  Dieser Provider bietet im Gegensatz zum FileSystemProvider eine Versionskontrolle an. Hier werden die Dateien, im für die Speicherung der Dateien konfigurierten Ordner, *flach* (auf einer Ebene) abgelegt. Zusätzlich werden ältere Versionen im Verzeichnis *OLD*, in Ordnern welche den Namen der Wiki-Seite tragen, abgelegt. Die-Verzeichnis- bzw. Dateien-Struktur sieht dann folgendermaßen aus:

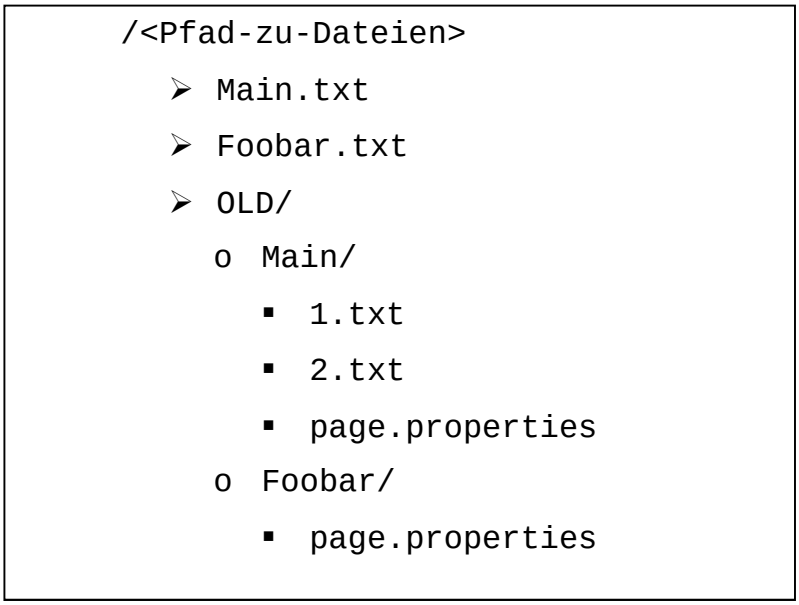

Abbildung 43: Struktur auf Verzeichnisebene mit VersioningFileProvider

[25] (http://doc.jspwiki.org/2.4/wiki/FileSystemProvider) [Zugriff: 08.11.2011]

In diesem Fall hat die Seite *Main* drei Versionen, welche einfach numerisch inkrementiert werden. Die Seite *Foobar* hat währenddessen nur eine Version. Die Datei *page.properties* beinhaltet zur Seite zugehörige Metainformationen wie Autor, Erstellungsdatum usw.[26]

- *RCSFileProvider*
  Dieser ist eine Erweiterung des FileSystemProvider, dem eine Seiten-Versionierung mittels RCS (Revision-Control-System) zugefügt wurde. Laut JSPWiki-Homepage ist von diesem Provider auch abzuraten, da RCS bei großem Datenaufkommen unstabil werden kann.[27]

Für die Verwaltung der Anhänge gibt es standardmäßig einen Provider:

- *BasicAttachmentProvider*
  Wie schon gesagt ist dieser Provider für die Verwaltung von an den Wiki-Seiten angehängten Dateien wie Bilder oder Videos (im `jspwiki.properties` File können die zulässigen Dateiformate für die Anhänge eingestellt werden) zuständig und beinhaltet auch eine Versionskontrolle.
  Gehen wir vom Beispiel des FileSystemProviders vorhin aus und hängen der Datei *Main* ein Bild mit dem Namen *picture.jpg* an. Der Provider erzeugt nun in einem festgelegten Ordner (wird im `jspwiki.properties` File definiert) einen Ordner mit dem Namen *Main-att* in welchem wieder ein Ordner *picture.jpg-dir* angelegt wird. Die Datei wird dann in diesem Ordner mit dem Namen 1.*jpg* angelegt, da es die erste Version dieses Bildes ist. Unter *attachment*.*properties* werden wiederum Metainformationen über die angehängte Datei gespeichert (Trattner a, 2009).

[26] (http://doc.jspwiki.org/2.4/wiki/VersioningFileProvider) [Zugriff: 08.11.2011]
[27] (http://doc.jspwiki.org/2.4/wiki/RCSFileProvider) [Zugriff: 08.11.2011]

Die Struktur auf Verzeichnisebene sieht nun folgendermaßen aus:

```
/<Pfad-zu-Dateien>
  ➢ Main.txt
  ➢ Foobar.txt
  ➢ Main-att/
    o picture.png-dir/
      ▪ 1.png
      ▪ attachment.properties
  ➢ OLD/
    o Main/
      ▪ 1.txt
      ▪ 2.txt
      ▪ page.properties
    o Foobar/
      ▪ page.properties
```

Abbildung 44: Struktur auf Verzeichnisebene mit VersioningFileProvider und BasicAttachmentProvider

4.2 Austria-Forum

Wie schon öfters erwähnt, ist das Austria-Forum eine online Enzyklopädie über Österreich, mit Inhalten welche von einem *Editorial-Board* mit ausgewählten Experten kontrolliert bzw. verfasst werden und die Artikel dadurch zitierbar sind. Technisch basiert es auf dem Wiki-System *JSPWiki*, jedoch mit vielen technischen Verbesserungen und neuen Features wie z.B.:

- Serverseitige Größenanpassung von hochgeladenen Bildern und automatische *Thumb* Generierung
- Erweiterung des Editors z.B. um Tabellen leichter zu erstellen oder Texte markieren zu können
- Die Möglichkeit des Hochladens von Dateien wird jetzt schon innerhalb des Editors angeboten

- Qulitätsgütesiegel – *Aprooved* – gesperrte Beiträge werden damit eingefroren und können nur noch auf Systemebene geändert werden.
- Ein eigenes *Section Tagging*-System
- Einige neue Filter und Plugins wurden hinzugefügt
- Viele Fehler des alten Systems wurden verbessert

Das System des Austria-Forums, in Folge AFWiki genannt, zeichnet sich auch als sogenanntes *Structured Wiki* aus. Dies bedeutet, dass die Inhalte durch diverse technische Neuerungen und Features strukturiert dargestellt werden und so eine bessere Übersicht über die Inhalte des Wikis und somit ein komfortableres Navigieren durch die Angebote dieser Online-Enzyklopädie ermöglicht wird.

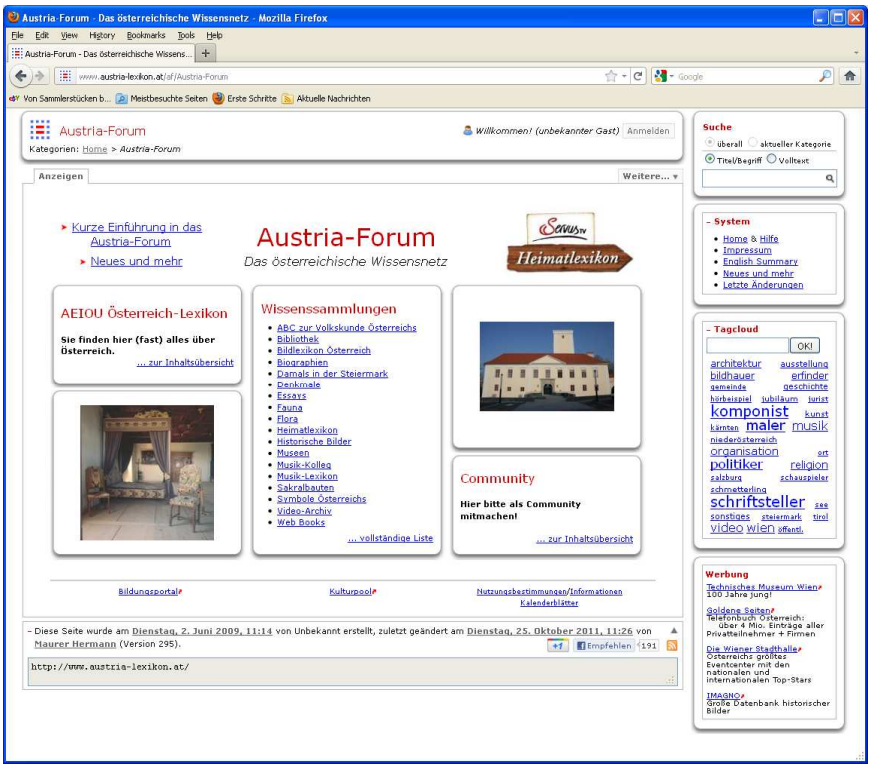

Abbildung 45: Startseite des Austria-Forum mit den 3 Hauptkategorien

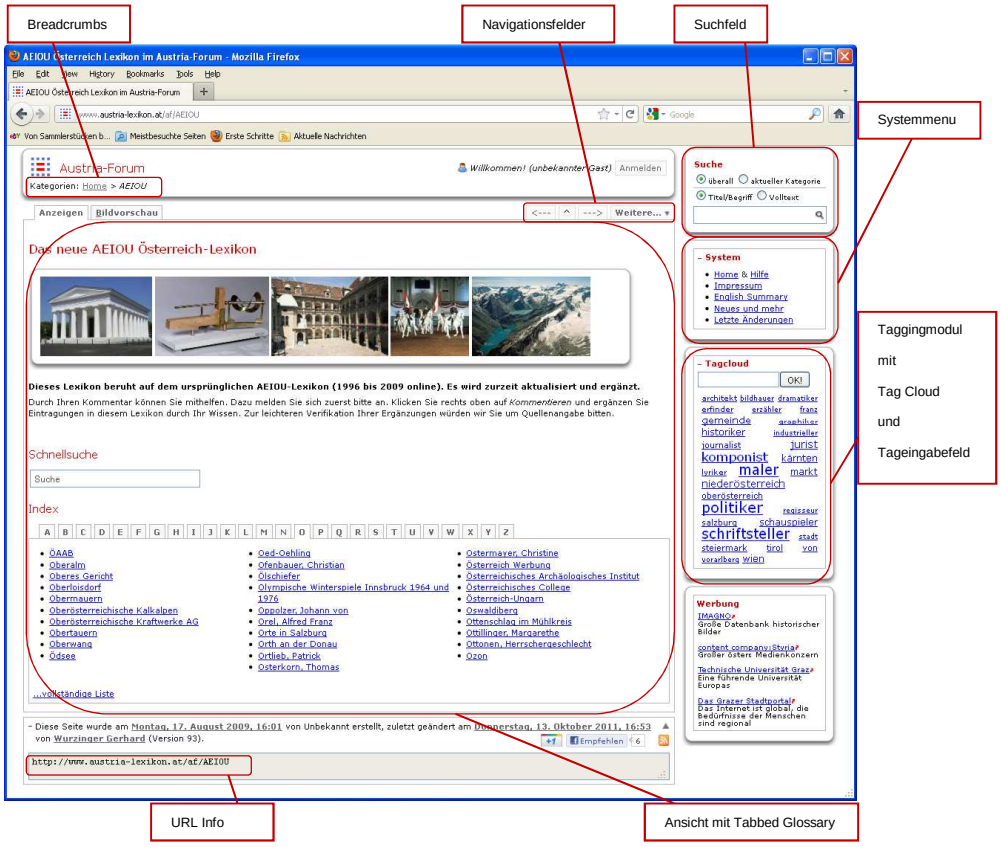

Abbildung 46: Strukturierung im Austria-Forum

Auf der Startseite des Austria-Forums sind gleich die drei Hauptkategorien sichtbar (Abbildung 45). In Abbildung 46 sind die wichtigsten strukturellen Elemente des AFWiki abgebildet, welche eine Navigation durch das System erheblich erleichtern:

- *Hierarchische Breadcrumbs*
  Erleichtert die Navigation um einfach wieder in eine übergeordnete Kategorie zu gelangen oder um die Orientierung im System zu behalten.

- *Navigationsfelder*
  Um eine Seite im Alphabet vor oder zurück oder in die übergeordnete Hierarchieseite zu gelangen.

- *Systemmenu*

  So sind schnell wichtige Systemseiten wie z.B. die Hilfe erreichbar.

- *Taggingmodul*

  Um über ein Eingabefeld Dokumente zu kommentieren oder mittels *Tag Cloud* artverwandte Dokumente ausfindig zu machen und evtl. dorthin zu navigieren.

- *Suchfeld*

  Für Titel/Begriff- oder Volltextsuche in der aktuellen Kategorie oder überall.

- *URL Info*

  Für eine leichtere Seitenreferenzierung oder Orientierung im System.

Im Ansichtsfeld können weiter strukturelle Features eingebunden werden, welche die Übersicht und Navigation erleichternd beeinflussen wie z. B.:

- *CategoryIndexPlugin*

  Hiermit kann ein alphabetisches Inhaltsverzeichnis über eine Kategorie erzeugt werden:

Abbildung 47: Der CategoryIndexPlugin im Austria-Forum

- *GlossaryPlugin*

  Damit kann ein Index, der den jeweiligen Buchstaben zugeordneten Begriffen erstellt werden:

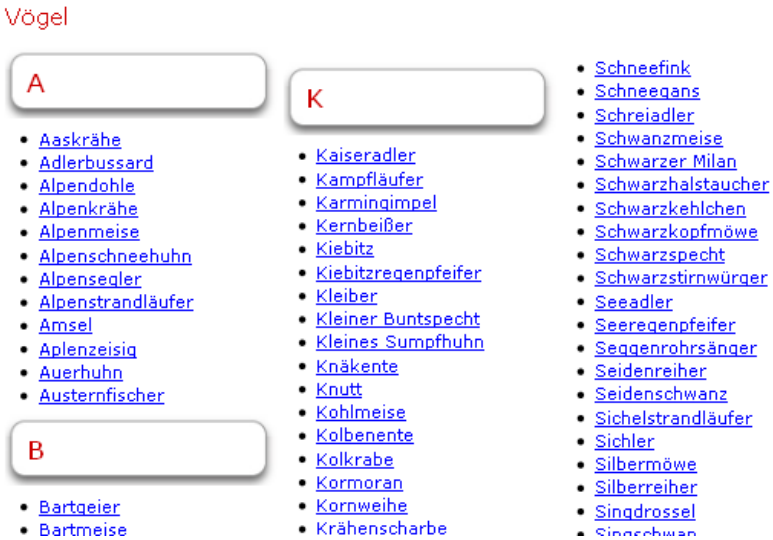

Abbildung 48: Der GlossaryPlugin im Austria-Forum

- *TabbedGlossaryPlugin*

  Dient zur Erstellung eines Inhaltsverzeichnisses, welches in Rastern unterteilt ist, welche den einzelnen Anfangsbuchstaben der Artikel zugeordnet sind:

Abbildung 49: Der TabbedGlossaryPlugin im Austria-Forum

- *Bildvorschau mit ListCategoryThumbs*

  Erstellt eine kleine Vorschau der Bilder in einer Kategorie:

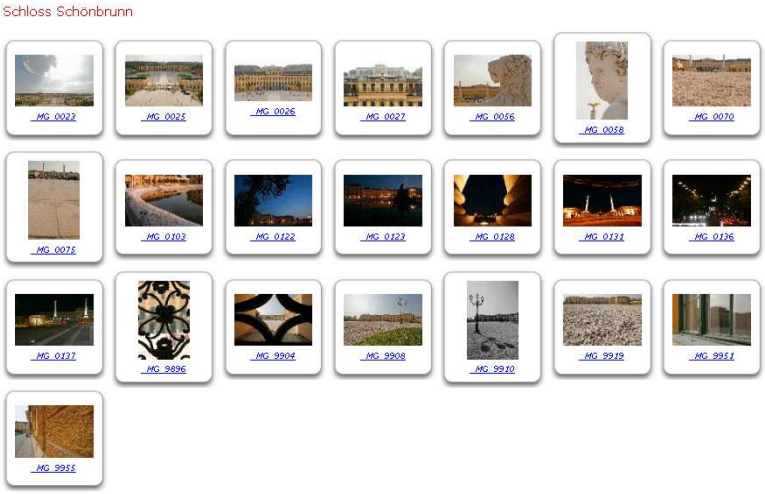

Abbildung 50: Bildvorschau mit ListCategoryThumbs im Austria-Forum

- *Bildfolge mit SlideShowPlugin*

  Hiermit kann eine Slide-Show über alle Bilder der Kategorie erstellt werden:

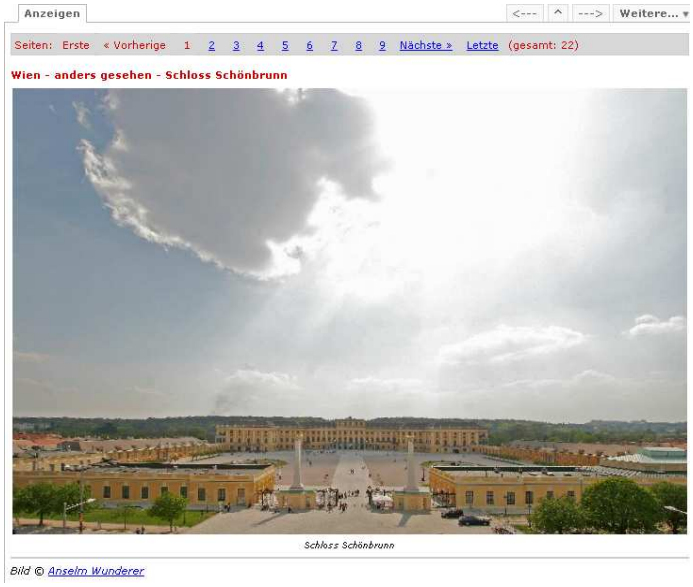

Abbildung 51: Bildvorschau mit ListCategoryThumb im Austria-Forum

Insgesamt wurden ca. 20 sogenannte *strukturgebende* Plugins entwickelt, um einen einfacheren Umgang mit den Inhalten von Dokumenten und dessen Anhängen zu gewährleisten (Trattner, Helic, Hasani-Mavriqi, & Leitner, 2010).

4.2.1 Strukturierung der Daten

All den technischen Neuerungen, welche eine Strukturierung des Inhaltes und dadurch ein effizienteres Navigieren ermöglichen, geht die Umstellung auf eine strukturierte Kategorisierung und Dateienspeicherung auf Systemebene voraus.

Wie in den bisherigen Erläuterungen schon ersichtlich, unterstützt AFWiki ein Konzept zur Kategorisierung der Beiträge. Dieses System ist als *sub-paging* bekannt und wird auch in Systemen wie MediaWiki, PHPWiki oder Twiki eingesetzt. Allerdings ist es in diesen Systemen auf nur eine Unterkategorie limitiert, welches im AFWiki nicht der Fall ist (unlimitiert). Die Kategorien werden vom Editorial-Board erstellt. Diese bilden eine hierarchische Struktur und sind somit auch hierarchisch durchsuchbar. Kategorien können auch verschoben oder umbenannt werden wobei ein eigenes Referenzierungsmodul auch die innere Linkstruktur in den Dokumenten aufrecht hält (Trattner, Helic, Hasani-Mavriqi, & Leitner, 2010).

Jeder Beitrag ist also Teil genau einer Kategorie, jede Kategorie hat genau eine *Eltern*-Kategorie und kann *Eltern*-Kategorie von vielen Unterkategorien und Seiten sein. Dadurch ergibt sich ein einfaches strukturiertes bzw. hierarchisches Adressierungsschema der URL:

```
http://www.austria-lexikon.at/<category-page>/<page>
http://www.austria-lexikon.at/<category-page>/<category-page>/<page>
.
.
```

Um z.B. alle Beiträge der Kategorie Fauna zu erhalten ist folgende Notation zu benützen:

```
http://www.austria-lexikon.at/af/Wissenssammlungen/Fauna
```

Will man nun eine Stufe in der Hierarchie weiter und z.B. den Beitrag des Dachs lesen ist folgende Notation von Nöten:

```
http://www.austria-lexikon.at/af/Wissenssammlungen/Fauna/Dachs
```

(Trattner b & Helic, 2009)

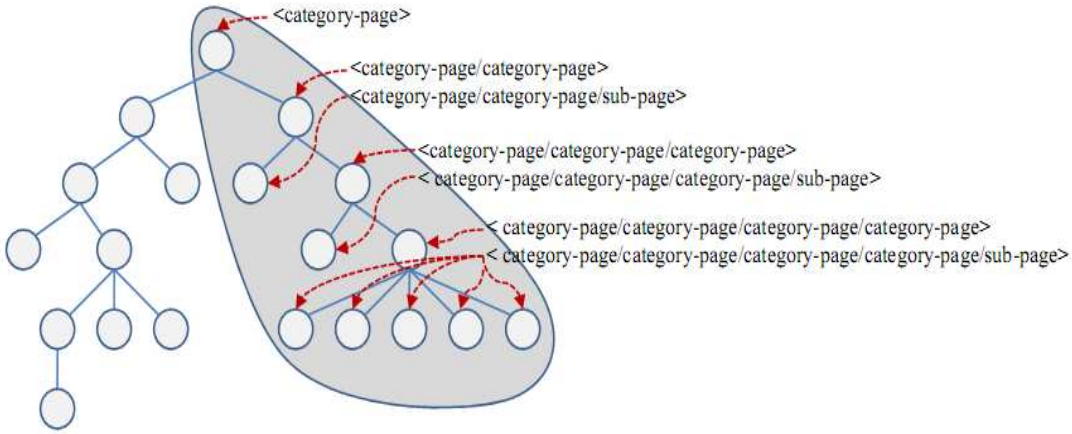

Abbildung 52: Hierarchisches Adressierungsschema im AFWiki (Trattner c, Helic, & Strohmaier, 2011)

Auch auf Systemebene wurde dieses hierarchische System übernommen. Hier bilden die Kategorien einfach Verzeichnisse bzw. Ordner, welche wiederum Unterverzeichnisse und Dateien beinhalten, die die Textdateien der Artikel oder dessen Anhänge darstellen.

Die Pfadangabe auf Verzeichnisebene kann folgend definiert werden:

```
<Pfad-zu-Dateien>/<Category>
<Pfad-zu-Dateien>/<Category>.txt
<Pfad-zu-Dateien>/<Category>/<Page>.txt
<Pfad-zu-Dateien>/<Category>/<Category>
<Pfad-zu-Dateien>/<Category>/<Category>.txt
<Pfad-zu-Dateien>/<Category>/<Category>/<Page>.txt
.
.
```

Im Gegensatz zu JSPWiki werden in AFWiki die Dateien der Seiten (Verzeichnis *pages*) und die Dateien der Anhänge (Verzeichnis *attach*) in getrennten Ordnern abgelegt. Die Dateien der Versionskontrolle sind wiederum im Ordner *OLD* im Verzeichnis *pages* abgelegt. Diese können jetzt hierarchisch in mehreren Ebenen gespeichert werden, im Gegensatz zur flachen Speicherung der Dateien in einer Ebene in JSPWiki. Die Verzeichnisebenen sind im Ordner *pages* und *attaches* dieselben. Im Beispiel aus ist ersichtlich, dass *Category* eine Unterkategorie der Hauptkategorie

Main ist. In *Main* ist zusätzlich noch ein Beitrag *Page1* enthalten und die Kategorie *Category* beinhaltet einen Beitrag namens *Page2*. Weiters hat *Main* und *Category* jeweils ein Bild als Anhang dabei.

Die Struktur auf Verzeichnisebene sieht dann folgendermaßen aus:

```
/<Pfad-zu-Dateien>
   ➢ attach/
      • Main/
         o Category/
            ▪ Page2-att/
               ➢ Pic_Pc.jpg-dir/
                     • 1.jpg
                     • attachment.properties
      • Main-att/
         o Pic_Main.jpg-dir/
            ▪ 1.jpg
            ▪ attachment.properties
   ➢ pages/
      • Foobar.txt
      • Main.txt
      • Main/
         o Category.txt
         o Page1.txt
         o Category/
            ▪ Page2.txt
      • OLD/
         o Foobar/
            ▪ 1.txt
            ▪ page.properties
         o Main/
            ▪ 1.txt
            ▪ Category/
               ➢ 1.txt
               ➢ Page2/
                     • 1.txt
                     • page.properties
               ➢ page.properties
            ▪ Page1/
               ➢ 1.txt
               ➢ page.properties
            ▪ page.properties
```

Abbildung 53: Struktur auf Verzeichnisebene des AFWiki

4.2.1.1 Wichtigste Änderungen auf Codeebene für Hierarchisierung

In Kapitel 2.1 wurde aus informationstechnischer Sicht schon auf die Gründe für eine Umstellung auf eine hierarchische Strukturierung des Verzeichnissystems eingegangen. Hier möchte ich nun die wichtigsten, dafür notwendigen Änderungen auf Codeebene des Wiki-Systems erläutern, um solch eine Verzeichnisstruktur fürs AFWiki zu ermöglichen.

In der Klasse *MarkupParser* wurde die Konstante `LEGACY_CHARS_ALLOWED` um die Parameter `/´®†!“„-?'~'©` erweitert. Diese Konstante bezeichnet alle Punktationszeichen, welche in der Wiki Markup zulässig sind. Vor allem das / (Slash) ist hier äußerst wichtig um z.B. InterWiki-Links in Texten, welche in AFWiki ja hierarchisch sind, zu bezeichnen.
Dieselbe Erweiterung erfuhr auch die Konstante `PUNCTUATION_CHARS_ALLOWED`. Diese listet alle Punktationszeichen auf, welche für Seitennamen zulässig sind. Auch hier ist das Slash die wichtigste Erweiterung, um eben die gesamte Hierarchie eines Seitennamens bezeichnen zu können.

In der Klasse *AbstractFileProvider* wurden einige Methoden hinzugefügt bzw. geändert:

- `public Collection getAllPages() throws ProviderException`

  Da die Dateien jetzt hierarchisch in Ordnern angelegt sind wurde die Funktion so abgeändert, dass die Ordner rekursiv nach Pages durchsucht werden um diese aufzulisten.

- `private int getPageCount(String dir)`

  Diese Methode wurde um die Funktion `getPageCountHelp` erweitert, welche ebenfalls die Ordner rekursiv nach Seiten durchsucht und diese somit zählt.

- `public Collection findPages(QueryItem[] query)`
  Auch diese Funktion wurde so geändert, dass die Ordner rekursiv durchsucht werden um Seiten mit dem gewünschten Suchbegriff zu finden.

- `public void deletePage(String pageName) throws ProviderException`
  Da eine Seite in AFWiki auch eine Kategorie sein kann wurde diese Funktion so geändert, dass die Kategorie auch gelöscht wird.

- `public boolean deleteCategory(String pageName) throws ProviderException`
  Diese Funktion wurde hinzugefügt, um eine komplette Kategorie inklusive Unterkategorien und deren Seiten zu löschen.

- `public Collection getSubPages(String category) throws ProviderException`
  Diese Funktion ist auch neu und listet alle Unterseiten einer Kategorie ohne die Seiten der Unterkategorien einzubeziehen.

- `public Collection getAllCategoryPages(String category) throws ProviderException`
  Diese Funktion listet alle Seiten inklusive der Seiten der Unterkategorien auf. Sie durchsucht die Ordner dabei auch rekursiv.

- `public int getTotalCategoryPageCount(String category) throws ProviderException`
  Wird verwendet, um alle Seiten einer Kategorie zu zählen. Diese Funktion wurde auch für AFWiki neu hinzugefügt.

- `public void createCategory(String page)`
  Erstellt eine neue Kategorie und auch ein Verzeichnis für die Unterkategorien und Dateien dieser Kategorie auf Systemebene.

- `public boolean isCategory(String pagename) throws ProviderException`
  Wird dazu verwendet, um zu kontrollieren ob eine Seite auch eine Kategorie ist.

Um die hierarchische Dateienverwaltung auch für die Speicherung der Anhänge der einzelnen Beiträge zu ermöglichen, wurde die Klasse *AustriaForumAttachmentProvider* neu implementiert, welche vom Interface *WikiAttachmentProvider* abgeleitet ist.

5. Implementation des webbasierten Tree Browser

Nach all den bereits getätigten Maßnahmen zur Strukturierung und dadurch leichteren Bedienung bzw. Navigation durch das System des Austria-Forums, sollte noch eine Erleichterung dafür hinzugefügt und auf Nutzen für das bereits bestehende System getestet werden, nämlich eine Visualisierung des Verzeichnissystems, mit welcher auch diverse Datenmanipulationen zur Administration vorgenommen werden können. In Kapitel 2.2 wurde schon über mögliche Visualisierungsmöglichkeiten von Verzeichnissystemen diskutiert. Die Entscheidung ist aufgrund des eigentlich geringen Platzangebots am Bildschirm für dieses Tool auf den sogenannten *Tree Browser* gefallen. Aufgrund der Tatsache, dass diese Methode einen relativ guten Überblick über das Verzeichnis auf doch geringem Raum bietet, und nebenbei sehr schlank in der Implementation ist, ist diese Visualisierungsmöglichkeit auch eine der am häufigsten verwendeten in webbasierten Systemen.

5.1 Aufgabenstellung

Um die am Institut für Informationssysteme und Computer-Medien (IICM) entstandene Softwarelösung für die technische Realisierung der Community-behafteten Online-Enzyklopädie Austria-Forum in struktureller Hinsicht an die Anforderungen des Austria-Forums vorzubereiten, wurde das File-Providermodul angepasst, wobei Daten (Kategorien/Beiträge) hierarchisch auf Verzeichnissystemebene abgelegt werden (die Umsetzung wurde im vorherigen Kapitel besprochen – Gründe hierfür wurden schon in Kapitel 2.1 beschrieben).

Um die Daten-Darstellung/Manipulation dem User auch über das Webinterface zugänglich zu machen, soll ein DHTML-File-System-Tree implementiert werden. Die Darstellung soll einer herkömmlichen Explorer-Ansicht entsprechen (die Gründe hierfür wurden in Kapitel 2.2 beschrieben). Des Weiteren sollen Dateien *on demand* geladen und visualisiert

werden können. Neben der Darstellung sollen die Funktionen *DELETE*, *RENAME* und *MOVE* via *Drag and Drop* implementiert werden, wobei nicht nur einzelne Dateien, sondern ganze Folders verschoben, gelöscht und umbenannt sollen werden können. Zudem ist das JSPWiki-Security-Modul mit einzubinden.

Hier möchte ich nochmal in Punkten die Anforderungen an das zu implementierende Tool auflisten:

- Darstellung des Verzeichnissystems von AFWiki im Webinterface
- Explorer ähnliche Ansicht (Tree Browser)
- Dynamische Inhalte (DHTML)
- Manipulationen im Verzeichnissystem ermöglichen (Delete, Rename und Move sowohl einzelner Dateien als auch kompletter Verzeichnisse)
- Drag and Drop
- Laden der Daten on demand
- Einbeziehen des Sicherheits-Moduls.

5.2 Technologieauswahl

Um die Anforderungen umzusetzen, sind sowohl serverseitig als auch clientseitig Module zu implementieren. Natürlich wäre es auch möglich, rein serverseitig zu arbeiten. Dies birgt jedoch den Nachteil, dass bei einer Änderung der Ansicht des zu entwickelnden *Tree Browsers* nicht nur der angeforderte Teil neu zu generieren ist, sondern der gesamte Inhalt des Webfensters neu geladen werden muss. Dies führt wiederum zu einem höheren Datenverkehrsvolumen.

Grundsätzlich suchen wir Technologien, die folgendes anbieten:

Clientseitig:

- Dynamische Darstellung von Webpageinhalten aufgrund der vom Server zur Verfügung gestellten Daten
- Kommunikation mit serverseitigem Interface

Serverseitig:

- Kommunikation mit Client
- Auswertung der Anfragen des Clients
- Erstellung der Antworten und zur Verfügung stellen dieser in einem für den Client erfassbarem Format

Welche Technologien stehen hier zur Verfügung bzw. können hier sinnvollerweise zum Einsatz gebracht werden?

5.2.1 Serverseitig

Wie in Kapitel 4 beschrieben, basiert AFWiki auf dem WikiWiki Klon JSPWiki. JSPWiki ist in Java und JSP entwickelt, weshalb serverseitig die Technologieauswahl vorgegeben ist und nicht weiter diskutiert werden muss. Obendrein sind alle gestellten, serverseitigen Anforderungen mit dieser Technik lösbar.

5.2.2 Clientseitig

Hier wird nicht über die Auswahl der Technologie zur Ansicht der statischen Webinhalte diskutiert (ist durch JSP auch schon vorgegeben), sondern mit welcher Technologie die dynamischen Inhalte und Daten generiert werden bzw. welche Technologie zur Kommunikation mit dem serverseitigen Interface herangezogen werden soll.

In JSPWiki und auch AFWiki wird für die Darstellung dynamischer Elemente und der Kommunikation mit dem Server JavaScript/Ajax verwendet.

Dies ist allerdings nicht alleinige Voraussetzung für das Verwenden dieser Technologie auch für das zu entwickelnde Tool.

Mögliche Techniken dafür wären z.B.:

- JavaScript/Ajax (wie schon erwähnt)
- Flash
- Ruby on Rails
- JavaFX
- VBScript
- Silverlight
- Flex

VBScript wurde von Microsoft entwickelt und hat eine geringe Verbreitung außerhalb von Microsoft-Produkten. Nebenbei endet die Unterstützung von Microsoft – es kann also von einer toten Programmiersprache bzw. Script gesprochen werden, weshalb diese Technik von vornherein ausgeschlossen werden kann.

Alle anderen Techniken außer JavaScript/Ajax sind Plugin-basiert, erfordern also die Installation eines Zusatzprogrammes, damit darin erstellte Anwendungen am Client lauffähig sind. Außerdem bieten alle ähnliche Möglichkeiten, wie z.B. die Erstellung von Webseiten mit 3D-Effekten, Animationen, Unterstützung diverser Videoformate und vor allem die Bereitstellung reichhaltiger Interaktionsmöglichkeiten für den User. Deshalb möchte ich nur Flash als den wohl bekanntesten Vertreter stellvertretend für alle Plugin-basierten Techniken herausnehmen und mit JavaScript/Ajax bzw. auf JavaScript basierenden Frameworks zu vergleichen.

5.2.2.1 Flash vs. JavaScript

	Flash	JavaScript	Vorteil
File Size	möglicherweise größere Dateien	kleine Dateigröße	JavaScript
Kompatibilität	nicht mit allen Browsern und Smartphone kompatibel	mit fast allen Browsern und vielen Smartphone kompatibel	JavaScript
Erlernbarkeit	schwerer erlernbar als JavaScript	Leichter erlernbar als Flash	JavaScript
Features	sehr umfangreich	weniger umfangreich	Flash
Graphik	gute Unterstützung	schlechter als bei Flash	Flash
Performance	bei komplexen Features wie z.B. 3D besser als JavaScript	schlechter als Flash bei komplexen Features	Flash
Multimedia	sehr gut unterstützt	nicht so gute Unterstützung	Flash
Verfügbarkeit	Plugin muss zuerst installiert werden.	ist Bestandteil aller wichtigen Browser	JavaScript
Interaktivität mit Webpage-Elementen	limitiert	sehr gut unterstützt	JavaScript

Tabelle 3: Entscheidungstabelle Flash vs. JavaScript (Lyckman, 2009)

Laut Bewertung in Tabelle 3 hat JavaScript knapp die Nase vorne. Flash ist eine gute Wahl bei der Einbindung von komplexen Animationen, Grafi-

ken oder Multimediaelementen, kann aber bei den für dieses Projekt wesentlichen Faktoren nicht punkten. Nimmt man die für uns wichtigsten Eigenschaften wie die Kompatibilität der jeweiligen Technik und das zur Verfügung stellen von Interaktivität an Elemente einer Webpage heraus, geht die Selektion eindeutig in Richtung JavaScript. Nebenbei hat JavaScript schon den vorher erwähnten Bonuspunkt, dass AFWiki, für dynamische Webpageelemente clientseitig, bereits JavaScript und Ajax einsetzt. Zusätzlich ist noch zu erwähnen, dass Flash zur Nutzung clientseitig die Installation eines Plugins erfordert. JavaScript muss im Browser zwar auch erst aktiviert werden, trotzdem ist es standardmäßiger Bestandteil aller bekannten Browser, obwohl sich bei der Interpretation des Codes nicht alle an einen Standard halten.

5.2.2.2 Ajax

Da der Begriff schon öfters gefallen ist noch eine kurze Erklärung was dahinter steckt. Ajax ist keine eigene Programmiersprache, sondern bezeichnet eine Technologie zur asynchronen Datenübertragung zwischen einem Client (Browser) und einem Server (Web-Server). Dabei stellt sich der Name aus den zugrundeliegenden Techniken zusammen: AJAX → ***A****synchronous* ***J****avaScript* ***A****nd* ***X****(HT)ML*. Durch Einsatz dieser Technolgoie ist es möglich, HTTP-Anfragen (mittels XMLHttpRequest-Objekt) durchzuführen und die angezeigte HTML-Seite nur an den erforderlichen Stellen zu verändern. Somit muss nicht die ganze Seite neu geladen werden (Garret, 2005).

Durch den Einsatz dieser Technologie können Anwendungen erzeugt werden, die ein desktopähnliches Verhalten aufweisen.

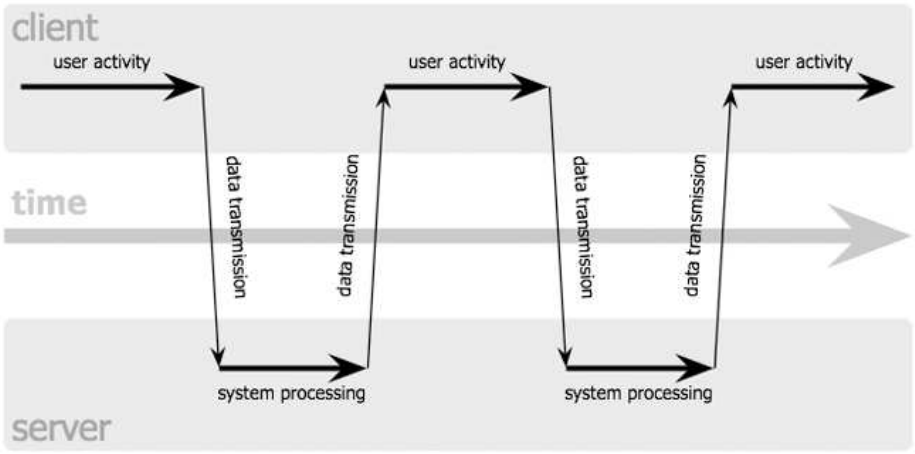

Abbildung 54: Klassisches (synchrones) Modell einer Web-Anwendung

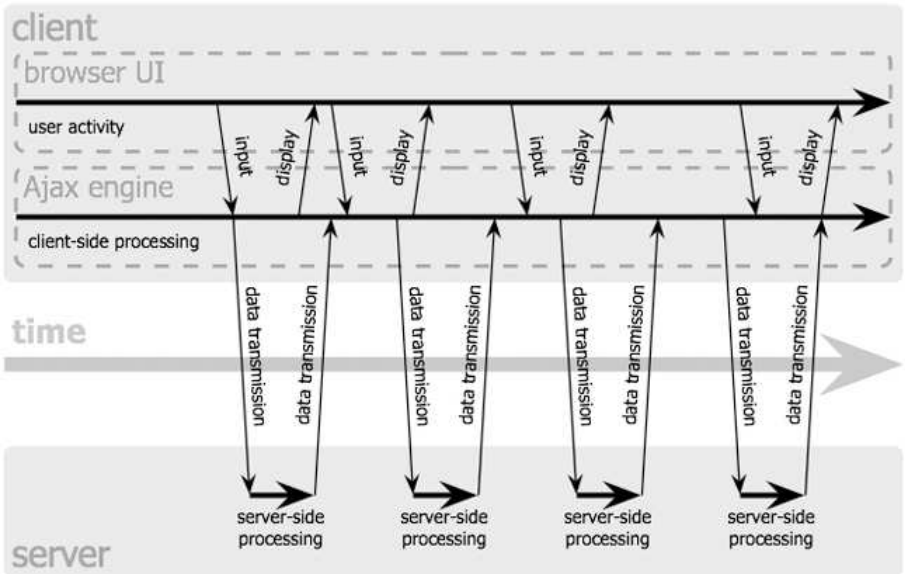

Abbildung 55: Ajax (asynchrones) Modell einer Web-Anwendung (Garret, 2005)

5.3 Frameworks

JavaScript/Ajax wird nach der vorher beschriebenen Selektion für die clientseitige Implementation dynamischer Inhalte und zur Kommunikation mit dem Server verwendet. Die nächste zu klärende Frage ist, ob es bereits vorhandene Frameworks oder sonstige Projekte gibt, welche die wichtigsten Eigenschaften dieses Projektes auf Basis der gewählten Technik anbieten und sich den Bedürfnissen hier anpassen lassen.

Folgende drei Projekte bzw. Frameworks, welche einen webbasierten Tree Browser implementieren, fanden den Weg in die engere Auswahl.

- *‚Folder tree with drag and drop'* von *'dhtmlgoodies'* (http://www.dhtmlgoodies.com/index.html?whichScript=drag-drop-folder-tree)

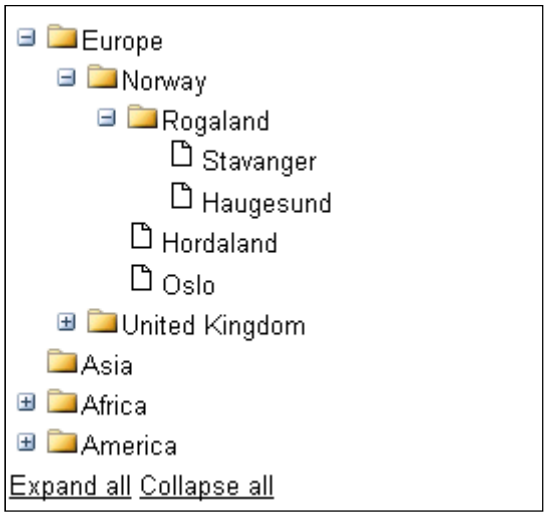

Abbildung 56: "Folder tree with drag and drop" von "dhtmlgoodies"[28]

- *'JavaScript Tree Component with Drag-and-Drop Capabilities'* von *'dhtmlx'* (http://www.dhtmlx.com/docs/products/dhtmlxTree/index.shtml?pl)

Abbildung 57: "JavaScript Tree" von "dhtmlx"[29]

[28] (http://www.dhtmlgoodies.com/scripts/drag-drop-folder-tree/drag-drop-folder-tree.html) [Zugriff: 29.11.2011]
[29] (http://www.dhtmlx.com/docs/products/dhtmlxTree/index.shtml?pl) [Zugriff: 29.11.2011]

- *‘JSP WikiTree’* auf *‘JSPWiki’* (http://www.jspwiki.org/wiki/JSPWikiTree)

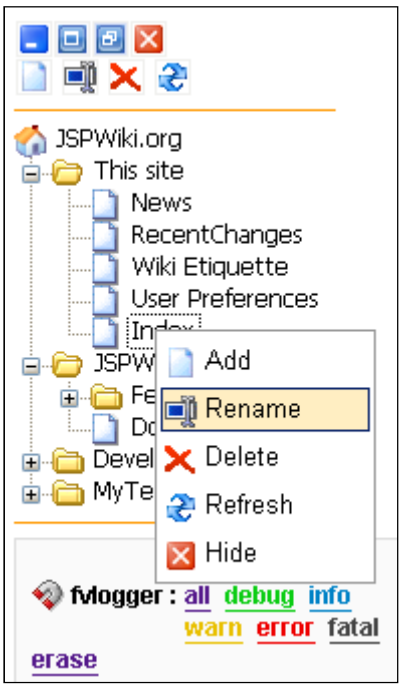

Abbildung 58: JSP WikiTree[30]

Alle drei Systeme bieten bereits einige brauchbare Features an, um als Grundlage für dieses Projekt zu dienen. Folgend ist die Bewertungstabelle dargestellt, die zur Entscheidungsfindung diente:

[30] (http://www.jspwiki.org/wiki/JSPWikiTree) [Zugriff: 29.11.2011]

	dhtmlgoodies	dhtmlx	JSP Wi-kitree
Tree Browser im Webinterface	ja	ja	ja
Dynamische Webinhalte	ja	ja	ja
Manipulationen im Verzeichnissystem Ermöglichbar	ja	ja	ja
Drag and Drop	ja	ja	nein
Laden der Daten on demand	nein	nein	nein
Drop-Down Menu	ja	ja	ja
Erweiterungs- und Änderungsmöglichkeit	leicht möglich	möglich	möglich
Lizenz	LGPL	GPL	LGPL

Tabelle 4: Bewertungstabelle für Frameworks

Aus der Tabelle ist ersichtlich, dass alle drei Methoden als gute Basis dienen könnten. Die Entscheidung viel schlussendlich auf das Tool von *dhtmlgoodies*. Dieses System wirkt für unsere Bedürfnisse wenig überladen, wodurch ein Erweitern bzw. Ändern des vorhandenen Systems leicht möglich ist. Obendrein ist es, gleich wie JSPWiki und somit auch AFWiki, als LGPL (GNU Lesser General Public Licence) lizenziert.

Der JSP WikiTree bietet von Haus aus kein Drag and Drop an. Obendrein bietet es serverseitig Module zur Aufarbeitung der Daten vom Server. Hierfür wurde eine Datenbank auf Basis von HSQLDB im *‚In-Process (Standalone) Mode‘* [31] entwickelt, aus der die zur Datengenerierung nötigen Module die Antwort für den Client erstellen. Dies ist zwar eine interessante Herangehensweise an das Problem, setzt jedoch voraus, dass bei

[31] (http://hsqldb.org/doc/guide/ch01.html#N101A8) [Zugriff: 29.11.2011]

jedem Neustart des Wikis die Datenbank neu mit Daten befüllt werden muss. Außerdem muss bei jeder Datenänderung auch dessen Datenbankeintrag geändert werden. All diese Umstände bedingen einen zeitlichen und speichertechnischen Mehraufwand.

Der Tree von *dhtmlx* bietet ähnliche Möglichkeiten wie der von *dhtmlgoodies*, ist jedoch in GPL (GNU General Public Licence) lizenziert. Ein Blick in die Statuten besagt, dass Programme, die Teile von GPL-Software enthalten als abgeleitete Werke betrachtet werden und somit die ganze restliche Software automatisch auch dieser Lizenz unterliegt, egal welche Lizenz diese Software vorher gehabt hat. Dies würde bedeuten, dass bei Einsatz dieses Tools, AFWiki automatisch der GPL unterzogen wird. Dies wäre im Grunde kein Problem, da sowohl GPL als auch LGPL Lizenzen zum Schutz freier Software sind, jedoch stellt LGPL keine Forderungen bezüglich abgeleiteter Software und ist nicht so streng, was das Hinzufügen von Kommentaren bei allfälligen Änderungen betrifft. Somit müsste jedes verwendete Plugin und sonstige verwendete Software von AFWiki kontrolliert werden, ob es die Bedingungen der GPL erfüllt.[32]

5.4 Architektur

JSPWiki, das als Basis für AFWiki dient, wurde nach dem Konzept des *MVC-Design-Pattern* entwickelt (MVC steht für **M**odel-**V**iew-**C**ontroller), welches für Informationssysteme bevorzugt eingesetzt wird. Ein Design-Pattern ist grob umschrieben ein von Experten erstelltes Prinzip zur Lösung eines wiederkehrenden Problems in der Softwaretechnologie. Sozusagen eine Vorlage für die Vorgehensweise für bestimmte Softwarelösungen.

[32] (http://www.gnu.org/licenses/lgpl.html) [Zugriff: 29.11.2011]

Das MVC-Modell wird in drei Bereiche eingeteilt:

- *Model*
  Beinhaltet die Module für die Speicherung sowie Bereitstellung von Daten für die anderen Bereiche.

- *View (Ansicht)*
  Beinhaltet die Module, welche für die Ansicht zuständig sind wie z.B. Benutzeroberfläche, Ansicht im Browser usw.

- *Controller*
  Ist die Schaltstelle und beinhaltet die Module, welche für die Kontrolle des Datenflusses zuständig sind.

Unser Tool dient als Erweiterung des AFWiki und setzt auf dieses auf, womit ein Design im MVC-Prinzip naheliegt.

In Abbildung 59 ist die Prozessansicht für den webbasierten Tree Browser ersichtlich. Die einzelnen Klassen und Files, welche dem jeweiligen Bereich des Entwicklungsmodells angehören, sind in den darin enthaltenen Containern eingetragen. Somit kann ein allgemeiner Überblick über das Zusammenspiel der einzelnen Programmteile im jeweiligen Bereich gegeben werden.

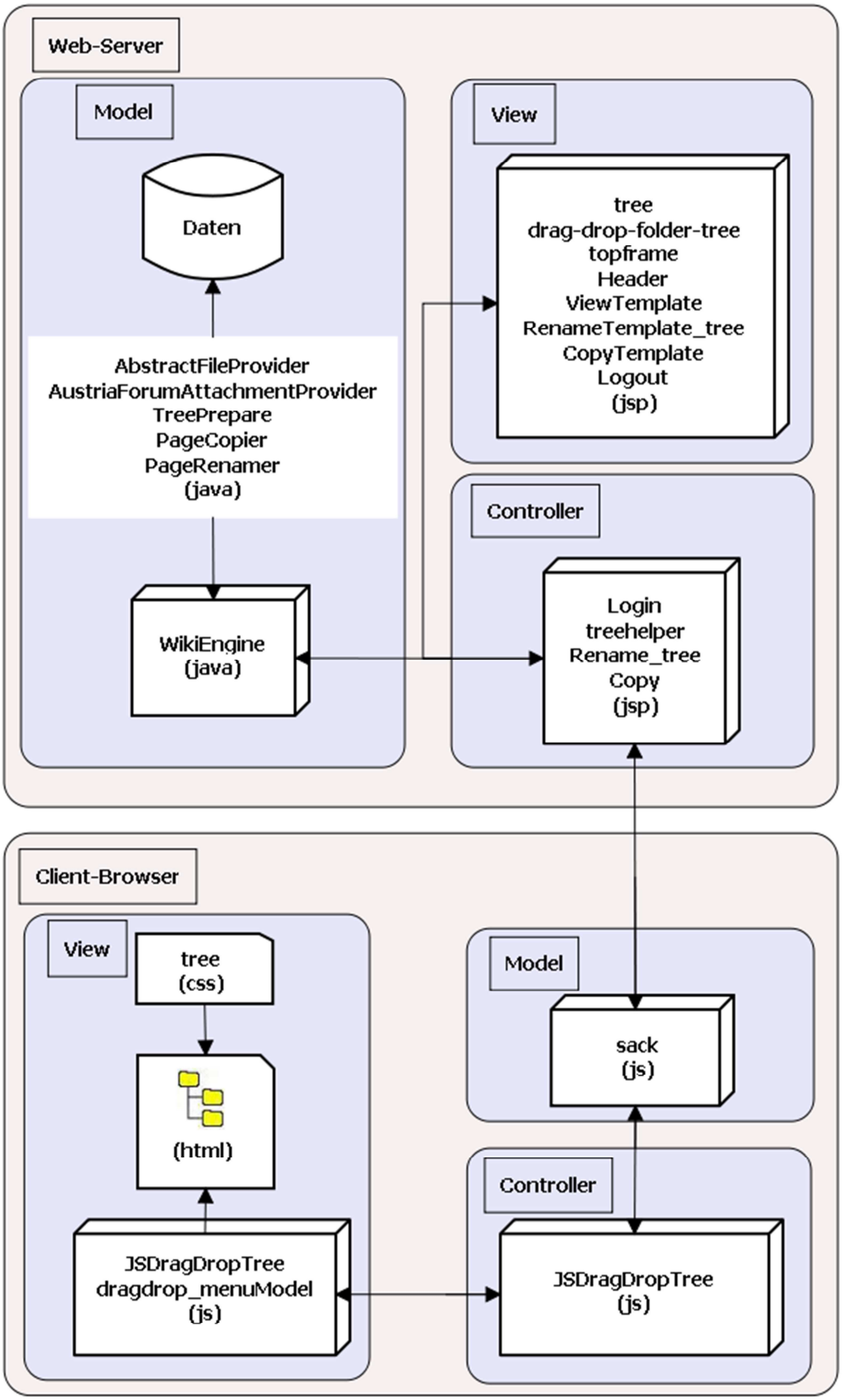

Abbildung 59: Prozessansicht des webbasierten Tree Browser

Wie schon in den vorherigen Kapiteln beschrieben, eignet sich JavaScript hervorragend um diverse dynamische Webinhalte als auch User-Interaktionen clientseitig zu verarbeiten. Einzelne JavaScript-Klassen lassen sich jedoch nicht eindeutig nur einem Bereich dieses Design-Pattern zuordnen, da sie sowohl Aufgaben für die Ansicht, als auch Kontrollfunktionen beinhalten, so wie hier zum Beispiel die JavaScript-Klasse `JSDragDropTree` (js).

Ausgehend vom Datenspeicher auf der Serverseite stellen die Java Klassen `AbstractFileProvider` (java), `AustriaForum-AttachmentProvider` (java), `TreePrepare` (java), `PageCopier` (java), `PageRenamer` (java) und `WikiEngine` (java) Funktionen zur Verfügung, um die für diese Implementation relevanten Daten vom Speicher aus- bzw. einzulesen als auch zur Aufbereitung dieser zur Übertragung an den Client. Die Abläufe werden von den JSP-Klassen `Login` (jsp), `treehelper` (jsp), `Rename_tree` (jsp) sowie `Copy` (jsp) gesteuert. Die Generierung des HTML-Codes für den Browser beim Client übernehmen die JSP-Klassen `tree` (jsp), `drag-drop-folder-tree` (jsp), `topframe` (jsp), `Header` (jsp), `ViewTemplate` (jsp), `RenameTemplate_tree` (jsp), `CopyTemplate` (jsp), sowie `Logout` (jsp) wobei die endgültige Formatierung von den Cascading Style Sheets (CSS) welche die Datei `tree.css` enthält durchgeführt wird. Weiters sind auf der Seite des Clients die JScript-Klassen `JSDragDropTree` (js) als auch `dragdrop_menuModel` (js) zur Interaktion mit einzelnen DOM-Knoten zuständig (DOM bedeutet **D**ocument **O**bjekt **M**odel). Die Daten werden von der JScript-Klasse `sack` (js) bereitgestellt als auch zur Übertragung entgegen genommen. Diese Klasse stellt AJAX-Funktionen (XMLHttpRequest) zur Verfügung. Die Steuerung am Client wird auch von der JScript-Klasse `JSDragDropTree` (js) übernommen.

5.4.1 Bemerkung zu Klassen

Ich spreche hier immer wieder von Klassen, egal ob es sich um Java, JSP oder JavaScript handelt. In Java ist dies berechtigt, hier wird auch das Schlüsselwort *class* zur Deklaration eines Klassenobjektes benützt. Auch in JSP darf von einer Klasse gesprochen werden, da der JSP Code durch den Java-fähigen Webserver immer in eine Servlet-Klasse umgewandelt wird.
In JavaScript gibt es die Bezeichnung jedoch nicht. Hier wird das *klassenähnliche* Objekt *Prototype* benützt. Damit lassen sich die wesentlichen Eigenschaften von Klassen in JavaScript umsetzen und ich werde weiter auch bei JavaScript von *Klassen* sprechen, auch wenn dies aus fachlicher Sicht nicht ganz korrekt ist (Meyer).

Zur Kennzeichnung einer Klasse wird in Klammer immer die Technologie angegeben zu welcher sie gehört, also z.B.:

- `NameDerJavaKlasse` (java)
  Bezeichnet eine Java-Klasse
- `NameDerJavaServerPageKlasse` (jsp)
  Bezeichnet eine Servlet-Klasse in JSP
- `NameDerJavaScriptKlasse` (js)
  Bezeichnet eine Klasse bzw. ein *Prototype*-Objekt in JavaScript

5.5 Implementation

Im vorherigen Kapitel wurden schon alle Klassen aufgelistet, welche gegenüber dem vorhandenen AFWiki (Stand Januar 2010) entweder neu hinzugefügt wurden, eine Änderung oder eine funktionale Erweiterung erfahren haben um die Funktion des Tree Browsers zu gewähren. Ein grober Überblick ist somit schon geschaffen. Dieses Kapitel soll einige genauere Einblicke ermöglichen.

5.5.1 Aufbau

In Abbildung 60 ist die Ansicht von AFWiki inklusive Tree Browser dargestellt.

Die Arbeitsfläche ist in drei Teil-Frames aufgeteilt:

- Austria-Forum
- Baumansicht
- Adressleiste

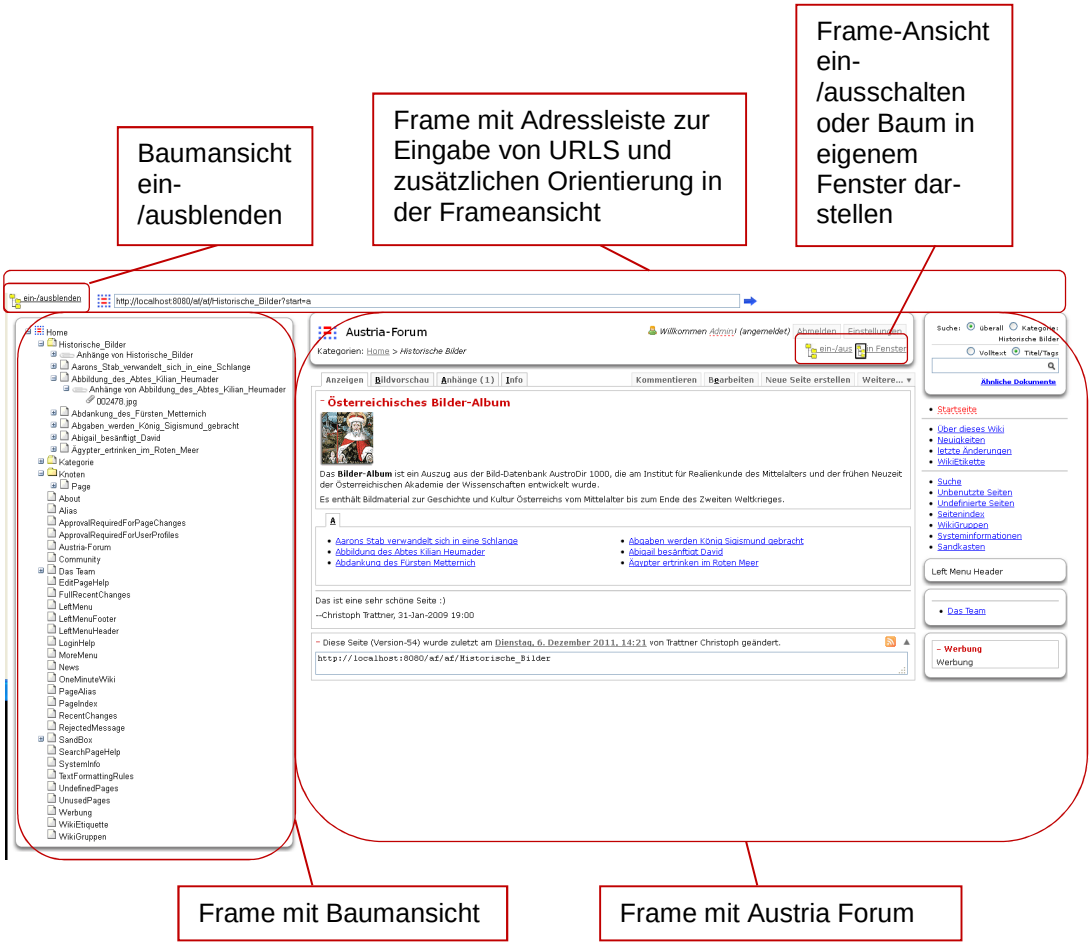

Abbildung 60: Arbeitsflächen des Austria-Forum mit Tree Browser

Am linken Rand der Adressleiste ist ein Button zum Ein- und Ausblenden des Tree Browsers zu sehen. Somit kann schnell ein größerer Ansichtsbereich für die Seiten des Austria-Forum geschaffen werden. In Abbildung 61 unten sieht man die Arbeitsfläche mit ausgeblendeter Baumansicht.

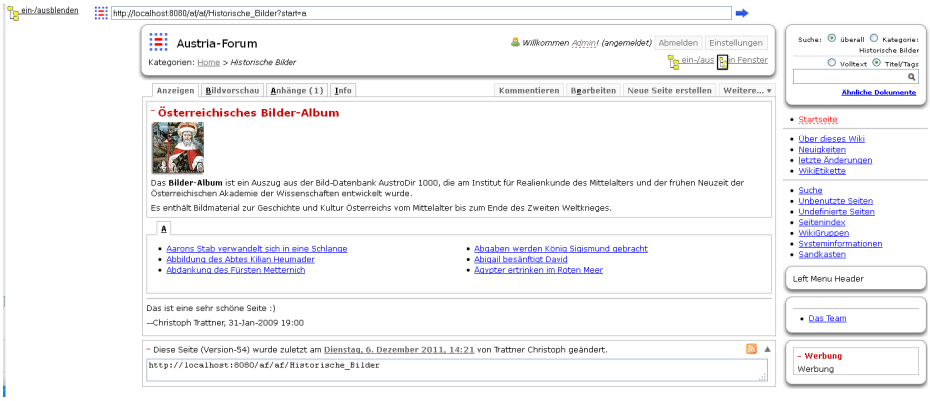

Abbildung 61: Arbeitsflächen des Austria-Forums mit ausgeblendetem Tree Browser

Falls jemanden die Darstellung in Frames nicht zusagt, bietet die Schaltfläche *ein-/aus* am rechten unteren Rand des Headers die Möglichkeit, in die originale Ansicht ohne Frames zu schalten.

Abbildung 62: Arbeitsfläche des Austria-Forums mit ausgeschalteter Frame-Ansicht

Ebenfalls an dieser Stelle befindet sich der Schalter *in Fenster*, der dazu dient, die Baumstruktur in einem eigenen Fenster darzustellen.

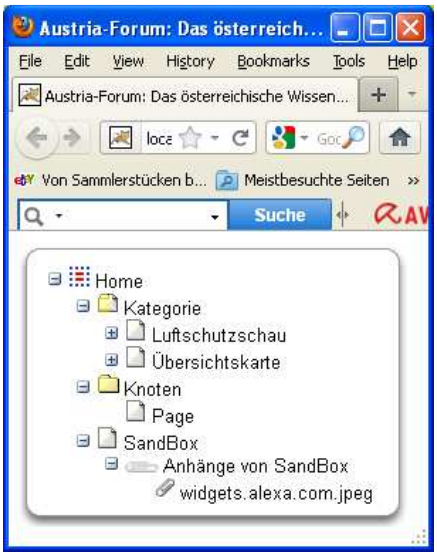

Abbildung 63: Baum in eigenem Fenster

Realisiert wird die Frame-Ansicht im JSP-File `tree.jsp` durch folgendes *FRAMESET*:

```
<FRAMESET ROWS="30,*" valign="top" BORDER="no"
BORDERCOLOR=#ffffff SPACING="0" ONLOAD="frame(<%=this.frame%>)" >
<FRAME SRC="topframe.jsp" NAME="Menue" valign="top" >
      <FRAMESET COLS="315,*" BORDER="5" SPACING="0" valign="top">
        <FRAME SRC="" NAME="Navigation" ID="Navigation" >
        <FRAME SRC="" NAME="Dates" ID="Dates" >
      </FRAMESET>
  <NOFRAMES>
    <BODY ONLOAD="frame(1)">
     <p><a href="<%=this.redirect%>">Austria-Forum</a></p>
    </BODY>
  </NOFRAMES>
</FRAMESET>
```

Listing 1: Code für Frameset in tree.jsp

Die drei Frames haben die Namen *Menu,* welche die Adressleiste beinhaltet, *Navigation* für den Tree Browser und *Dates*, um das ursprüngliche Austria-Forum anzuzeigen. Die Adressleiste und der JavaScript-Code, welche die Funktion der Leiste ermöglichen, sind in `topframe` (jsp) integriert. Des Weiteren ist dort auch die Funktion zum Aus- und Einblenden der Baumansicht enthalten.

5.5.1.1 Warum doch Frames in diesem Projekt

Frames sind im heutigen Webdesign eher ein Tabuthema. Immer mehr Webentwickler versuchen ohne diese Technik auszukommen. Zum Beispiel kann durch `<Table>` oder `<div>` Tags und der JSP-Direktive (im Fall von JSPWiki ist es eine Wiki-Direktive) `<wiki:Include>` sowie einer CSS-Formatierung ähnliches in einem Dokument geschaffen werden.

Die meistgenannten Kritikpunkte gegen Frames sind:

- *Suchmaschinen*
  Suchmaschinen arbeiten mit sogenannten *robots* oder *spiders* um Webseiten zu indizieren und in die Datenbank aufzunehmen. Diese Programme suchen nach Inhalt wie etwa Überschriften, Titel oder Meta-Tags. Solche Inhalte sind auf Frameseiten kaum vorhanden.

- *Bookmarking*
  Die dargestellte URL in der Adressleiste des Browsers bleibt durch Verwendung eines *Framesets* immer die gleiche, da sich das Hauptdokument, in welchem das Set definiert ist, nicht ändert. Will man nun irgendeine Seite vermerken, wird die URL zur Hauptseite gespeichert und nicht zum gewünschte Dokument.

- *Orientierung*
  Die Adresse in der Leiste des Browser dient vielen als Hilfsmittel zur Orientierung im Web. Dies fällt durch die bereits unter dem Punkt *Bookmarking* erwähnten Eigenschaften von *Framesets* weg.

Was waren nun die Beweggründe, diese Technik zum Einsatz zu bringen? Wie gesagt wäre, wie oben beschrieben, alternativ der Einsatz eines `<Table>` Elementes mit `<wiki:Include>` möglich. Das Problem ist, dass mit dieser Technik bei jedem Aufruf einer neuen Seite auch der Baum neu geladen werden muss. Dies bedeutet mehr Datenverkehr und mehr Rechenaufwand am Server und beim Client.

Wird nun eine neue Seite geladen, soll auch der Baum danach wieder die gleiche Struktur haben. Also muss vor dem Entladen der alten Seite eine JavaScript Funktion die Struktur des soeben gezeigten Baumes aufzeichnen. Demnach muss es den Baum durchscannen und alle erweiterten Knoten speichern, also solche mit einem Minus (-) vorne. Diese werden dann in einem Cookie abgespeichert. Beim Laden der neuen Seite wird dieses aufgerufen und alle erweiterten Knoten müssen on demand, mittels Ajax, wieder am Server nach Kindknoten untersucht werden. Diese werden dann wieder zum Client geschickt, wo die Struktur mittels JavaScript wieder hergestellt wird. Ist die Anzahl der abzufragenden Knoten etwas größer, kann das zu langen Ladezeiten führen. Noch dazu ist nicht abzusehen mit welcher Hardware und mit welcher Internetverbindung clientseitig gearbeitet wird. Dies kann mitunter am Nervenkostüm des Users zerren und diesen bald zum Verlassen des Angebotes bewegen.
Mit Frames muss der Baum nicht neu geladen werden und bewirkt weniger Datenverkehr und weniger Rechenaufwand. Dies ist auch der Hauptgrund, warum vor Jahren diese Technik entwickelt wurde.

Nun noch ein paar Worte zu den bereits erwähnten Nachteilen von Framesets und was zur Abschwächung dieser getan wurde:

- *Suchmaschinen*
  Die Ansicht in Framesets und somit die Darstellung des Verzeichnisbaumes ist kein Muss. Sie kann wie oben bereits beschrieben, komplett ausgeschaltet werden. Außerdem ist sie nur für registrierte Benutzer freigeschalten. Alle anonymen User navigieren in der normalen *framelosen* Ansicht durchs System. Somit steht auch einer Indexierung diverser Seiten durch Suchmaschinen und einer Verbesserung des *Rankings* nichts im Wege. Obendrein wurden in der Seite, welche das Frameset aufruft, suchmaschinenfreundliche Merkmale gesetzt wie z.B. ein aussagekräftiger Titel, Metainformationen und Hinweise im NOFRAME-Bereich.

- *Bookmarking*

  Wie schon erwähnt, ist die Ansicht in Frames abschaltbar, wodurch dann wie üblich ein Seitenvermerk gesetzt werden kann.

  Einige Browser, wie z.B. der *FireFox,* bieten auch die Möglichkeit durch Klick auf die rechte Maustaste den Inhalt der einzelnen Frames zu verzeichnen.

- *Orientierung*

  Durch die Integration des Tree Browser ist im Grunde eine Navigation und Orientierungshilfe bereits gegeben. Außerdem wurde im oberen Bereich eine Adressleiste hinzugefügt (siehe Abbildung 60, Abbildung 61, Abbildung 62), welche die Adresse der gerade angezeigten Seite wiedergibt. Sie kann auch zur Eingabe der gewünschten URL verwendet werden, um auf diesem Wege zur gewünschten Seite zu gelangen.

  Dazu kommt noch, dass für Zwecke der Orientierung AFWiki bereits *Breadcrumbs* im Header anbietet und auch die URL der gerade besuchten Page im unteren Bereich angezeigt wird.

5.5.1.2 Grundstruktur des Baumes

Der Baum basiert auf einer (X)HTML Aufzählungsliste, also eine Liste, die mit `<UL>-` und `<LI>-Tags` aufgebaut ist (UL = unordered list, also unsortierte Liste – LI = list item, also Listeneintrag), welcher durch Hinzufügen von `<IMG>-Tags`, `<A>-Tags` und das Setzen diverser Attribute, Mouse-Events und CSS-StyleSheets das endgültige Aussehen und die endgültigen Funktionen verliehen wird. Die Generierung direkt in eine (X)HTML Aufzählungsliste wurde gewählt, um so ein weiteres Umwandeln zu umgehen um die endgültige Ansicht darzustellen. Außerdem sind in JavaScript als auch Java gute Klassen integriert, um HTML-DOM Elemente anzusprechen und zu bearbeiten.

Der Aufbau der Liste erfolgt server- als auch clientseitig. Das erste Grundgerüst wird am Server erstellt. Dies enthält nur die nötigsten Informationen um nicht zu viele Daten übers Netz schicken zu müssen. Aus der am Server generierten Grundstruktur kann der Rest des Baumes clientseitig fertiggestellt werden. Der übrige Aufbau am Client wird allerdings nicht gleich nach Eintreffen der Daten in einem Schritt erledigt. So werden nur die für die richtige Ansicht des Baumes nötigsten Elemente und Attribute *initial* am Client erstellt, somit ist ein schnellerer Aufbau der Seite gewährt. Die endgültigen Elemente und Attribute werden bei einem allfälligen *onmouseover-Event* der einzelnen Einträge durchgeführt.
Diese stufenartige Generierung bietet eine ausgewogenere Arbeitsverteilung wodurch Leistungsspitzen etwas abgeflacht werden können und den Aufbau des Baumes beschleunigen.

5.5.1.3 Aufbau eines Listenelementes <LI>

Nehmen wir nun ein `<LI>`-Element heraus um die Aufbaustufen zu veranschaulichen:
In Listing 2 sieht man den Code eines Listeneintrages wie er am Server generiert und zum Client geschickt wird. Es sind alle notwendigen Informationen vorhanden, damit der Client den Eintrag weiterverarbeiten kann.

```
<li parentId="/af/af/" id="Historische_Bilder">
      <img src="/af/images/tree/_plus.gif" />
      <img src="/af/images/tree/both.png" />
      <a>Historische_Bilder</a>
</li>
```

Listing 2: Listenelement generiert vom Server

Nachdem die Daten am Client angekommen sind werden alle Elemente gescannt und die wichtigsten Event-Attribute hinzugefügt, damit mit dem Tree Browser schon gearbeitet werden kann.

```
<li parentId="/af/af/" id="Historische_Bilder">
      <img onmouseover="highlightText();"
           src="/af/images/tree/_plus.gif"/>
      <img onmouseover="highlightText();"
           src="/af/images/tree/both.png" />
      <a onmousedown="initDrag();"
         onmousemove="moveDragableNodes();"
         onmouseup=" dropDragableNodes();"
         onmouseover="highlightText();">
         Historische_Bilder</a>
</li>
```

Listing 3: Listenelement nachdem es am Client angekommen ist und die ersten Events hinzugefügt wurden

Wie in Listing 3 zu sehen, wird ein *onmouseover*-Event hinzugefügt, welches bei diesem Ereignis die JavaScript-Funktion `highlightText()` aufruft um den Endzustand des Elementes herzustellen wie es in Listing 4 zu sehen ist. Diese Funktion wird nur beim ersten *mousover*-Event eines Elementes durchgeführt. In diesem Zustand hat das Listenelement alle wichtigen Eigenschaften erhalten um alle darauf notwendigen Operationen durchführen zu können.

```
<li parentid="/af/af/" id="Historische_Bilder">
    <img onclick="showHideNode('false','Historische_Bilder');"
  onmouseover="highlightText();"src="/af/images/tree/_plus.gif"/>
    <img onmouseover="highlightText();"
         src="/af/images/tree/both.png"/>
    <a target="Dates" href="/af/af/Historische_Bilder"
       onmousedown="initDrag();"
       onmousemove="moveDragableNodes();"
       onmouseup=" dropDragableNodes();"
       onmouseover="highlightText();">
       <span style="background-color: transparent;"
           onmouseover="this.style.backgroundColor='gold';"
           onmouseout="this.style.backgroundColor='transparent';">
           Historische_Bilder</span></a>
</li>
```

Listing 4: Listenelement in der Endversion

5.5.1.4 Gegenüberstellung diverser Generierungsvarianten

Ein kurzer Test sollte die oben genannte aufgeteilte Variante zur Generierung der einzelnen Listenelemente untermauern. Verglichen wurde mit:

- Kompletter Generierung des Baumes serverseitig
- Generierung des Baumes in einem Schritt clientseitig.

Getestet wurde auf einem technisch normal ausgerüstetem Labtop über eine Verbindung mit 700 Kbit/s. Als Browser wurden der Internet Explorer - da mir dieser bei vorherigen Beobachtungen als der langsamste erschien wenn es um Generierung von JavaScript-Code ging – und der Mozilla FireFox - da dieser einer der beliebtesten Browser ist - verwendet. Als Testobjekt diente ein Baum mit ca. 1000 Knoten. Gestoppt wurde die Zeit vom Anfordern bis zur fertig geladenen Baumansicht. Das Ergebnis ist in folgender Tabelle abgebildet.

	komplett serverseitig	**ein Schritt clientseitig**	**aufgeteilte Generierung**
Internet Explorer	12 Sek.	20 Sek.	6 Sek.
Mozilla FireFox	10 Sek.	7 Sek.	5 Sek.

Tabelle 5: Testauswertung verschiedener Generierungsmodelle

Der Internet Explorer sackt in der Geschwindigkeit bei der Generierung in einem Schritt am Client stark ab und schneidet bei der zum Einsatz gebrachten Methode (aufgeteilte Generierung) am besten ab. Der Mozilla FireFox verzeichnet nicht so gravierende Geschwindigkeitsschwankungen, erlangt aber auch bei der eingesetzten Methode das beste Ergebnis.

5.5.1.5 Aufbau eines Baumes

Folgend soll noch anhand eines Beispieles die zugrunde liegende <UL><LI> Listenstruktur und der daraus resultierende Tree Browser mit und ohne CSS-Formatierung gezeigt werden.

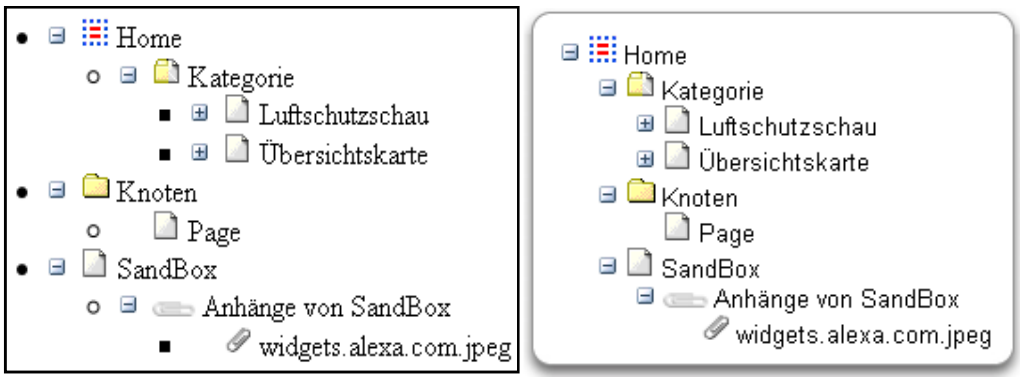

Abbildung 64: Links: Baumansicht ohne CSS. Rechts: Baumansicht mit CSS

```
<ul id="dragdrop_tree" class="dragdrop_tree">
 <li id="/af/af/" class="home.gif" nodrag="true" nodelete="true"
     norename="true">
     <img onclick="showHideNode('false','/af/af/');"
          src="/af/images/tree/_minus.gif" />
     <img src="/af/images/tree/home.gif" />
     <a> Home </a>
  <ul style="display: block;" id="tree_ul_1">
     <li parentid="/af/af/" id="Kategorie">
        <img src=" /af/images/tree/_minus.gif" />
        <img src="/af/images/tree/both.png" />
        <a>Kategorie</a>
        <ul style="display: block;" id="tree_ul_2">
          <li parentid="Kategorie" id="Kategorie/Luftschutzschau">
             <img src="/af/images/tree/_plus.gif" />
             <img src="/af/images/tree/file.png" />
             <a>Luftschutzschau</a></li>
          <li id="Kategorie/Übersichtskarte">
             <img src="/af/images/tree/_plus.gif" />
             <img src="/af/images/tree/file.png" />
             <a>Übersichtskarte</a></li>
         </ul></li></ul>
     <li parentid="/af/af/" id="Knoten">
        <img src=" /af/images/tree/_minus.gif" />
        <img src="/af/images/tree/foldericon.png" />
        <a>Knoten</a>
        <ul style="display: block;" id="tree_ul_3">
           <li parentid="Knoten" id="Knoten/Page">
              <img src="/af/images/tree/_plus.gif"
                   style="visibility: hidden;" />
              <img src="/af/images/tree/file.png" />
              <a>Page</a></li>
           <li style="display: none;" id="Knoten/noPage"></li>
        </ul></li>
```

```
    <li id="SandBox">
       <img src=" af/images/tree/_minus.gif" />
       <img src="/af/images/tree/file.png" />
       <a>SandBox</a>
       <ul style="display: block;" id="tree_ul_4">
          <li parentid="SandBox" id="att-SandBox">
             <img src="/af/images/tree/_minus.gif">
             <img src="/af/images/tree/attachments.png">
             <a>Anhänge von SandBox</a>
             <ul style="display: block;" id="tree_ul_5">
                <li id="atts-SandBox" parentid="SandBox"
                    atts="true">
                    <img src="/af/images/tree/_plus.gif"
                         style="visibility: hidden;" />
                    <img src="/af/images/tree/attachment_s.png" />
                    <a>widgets.alexa.com.jpeg</a></li>
</ul></li></ul></li></li></ul>
```

Listing 5: (X)HTML Grundstruktur der oben abgebildeten Baumstruktur nach Generierung am Server

5.5.1.6 Erklärung der Icons und Dateitypen

Zur Kennzeichnung der verschiedenen Dateitypen und Zustände werden Icons eingesetzt. Folgend sind die Wichtigsten aufgelistet:

-  *Kategorie*

  Bezeichnet eine Kategorie, also eine Seite, die noch weitere Unterseiten oder Unterkategorien aufweisen kann. Auf Filesystemebene bedeutet das, dass auf gleicher Ebene ein Ordner und ein Textfile einer Page des Systems mit gleichem Namen vorhanden ist. Das Textfile speichert die Informationen der jeweiligen Page in Wiki-Syntax. Im Ordner sind alle Unterseiten und Unterkategorien enthalten. Auf Codeebene wurde dafür die Bezeichnung *Both* (Ordner und Seite) verwendete.

-  *Ordner oder Folder*

  Bezeichnet einen Knoten bzw. Ordner der weitere Unterseiten oder Unterkategorien beinhalten kann, selbst jedoch keine eigene Page ist. Auf Filesystemebene ist im Gegensatz zum vorherigen Punkt kein Textfile vorhanden, sondern nur der Ordner, welcher alle dazugehörigen Unterseiten und Unterkategorien enthält.

-  *Seite oder Page*
  Bezeichnet eine einzelne Page, also ein normales Textfile auf Filesystemebene.
- *Container Anhänge*
  Bezeichnet den Container aller Anhänge einer Wiki-Seite oder einer Kategorie.
-  *Anhang*
  Bezeichnet einen einzelnen Anhang einer Wiki-Seite.
-  *Plus Icon*
  Bezeichnet einen zugeklappten Knoten. Ein Klick bewirkt das Erweitern des Knoten, also das Abrufen der Unterknoten vom Server.
-  *Minus Icon*
  Bezeichnet einen erweiterten Knoten. Ein Klick darauf bewirkt das Zuklappen des Knoten, also das Ausblenden der Unterknoten

5.5.2 Funktionen

In diesem Kapitel sollen die wesentlichen Funktionen erläutert werden sowie deren Ablauf und dadurch auch die wichtigsten Klassen und Methoden der Implementation abgehandelt werden.

5.5.2.1 Wichtige Klassen um Funktionen des Baumes zu ermöglichen

In Kapitel 5.4 wurden alle Klassen, welche für dieses Tool neu implementiert bzw. verändert wurden bereits erläutert. Hieraus sind die Klassen `TreePrepare` (java), `treehelper` (jsp) sowie `JSDragDropTree` (js) hauptverantwortlich um die Funktion des Baumes zu gewährleisten.

`TreePrepare` (java) stellt serverseitig die meisten Funktionen zur Verfügung, um Anfragen vom Client zu bearbeiten oder diesem Daten bereitzustellen. Dazu wird sie von den Klassen `AbstractFileProvider` (java) und `AustriaForumAttachmentProvider` (java) maßgeblich unter-

stützt, welche Funktionen zur Behandlung der Dateien des Systems beinhalten.

Auf Seiten des Clients ist in `JSDragDropTree` (js) alles enthalten um Anfragen (Requests) an den Server zu formulieren, die dadurch erhaltenen Daten (Responses) zu verarbeiten und die nötige Dynamisierung der Webseiteninhalte zu ermöglichen. Alle Funktionen für die AJAX-basierte Kommunikation finden sich in der Klasse `sack` (js).

Als Schaltzentrale zwischen Server und Client, also zwischen den beiden oben genannten Klassen, dient `treehelper` (jsp). Hier werden die Anfragen des Clients mit den notwendigen Funktionen am Server verbunden um die resultierenden Antworten an den Client zu retournieren. Hierfür wird in der URL der Parameter *what* mitgegeben. Dieser dient als ID, welcher über eine *if*-Abfrage in `treehelper` (jsp) zum richtigen Funktionsaufruf in `TreePrepare` (java) führt.

5.5.2.2 Vordergrund- Hintergrundprozess

In diesem System gibt es eigentlich keine Funktionen, die richtig zeitkritischen anzusehen wären. Das Erweitern von Knoten kann als ‚quasi' Echtzeitprozess bezeichnet werden. Der User möchte beim Navigieren durch das Angebot beim Klick auf das ‚Plus'-Icon einer Kategorie die Artikel natürlich so schnell wie möglich aufgelistet haben. Das Löschen, Kopieren und Verschieben von Kategorien mit einer großen Anzahl von Beiträgen kann mitunter sehr lange dauern. Hier sind nicht nur die Dateien und deren Anhänge auf File-System-Ebene zu ändern. Es müssen auch alle Referenzen auf die Artikel der Kategorie und deren Attribute einzelnen angepasst werden. Weiters muss bei jeder Datei die Rechtevergabe kontrolliert werden.

Somit wurde nur das Erweitern als Vordergrundprozess implementiert. Das Löschen, Verschieben, Umbenennen oder Kopieren wird über eine

Warteschlange im Hintergrund durchgeführt. JSPWiki bietet dafür die Klasse `WikiBackgroundThread` (java) an.

5.5.2.3 Authentifizierung und Initialisierung

Beginnen möchte ich bei der Authentifizierung und Initialisierung des Baumes. Der grundsätzliche sequenzielle Ablauf ist im folgenden Diagramm ersichtlich.

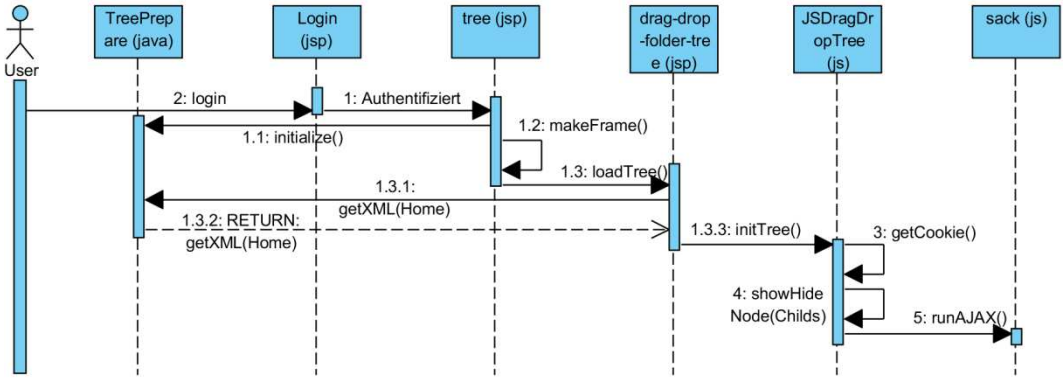

Abbildung 65: Sequenzieller Ablauf der Authentifizierung und Initialisierung

Abhängig von der Usergruppe hat nicht jeder die gleichen Berechtigungen. In dieser Phase geht es als erstes darum, wem Zugang zur Baumansicht gewährt wird. Laut Vorgabe soll dies jeder registrierte Benutzer erhalten. Hier ist zuerst mal wichtig, ob der Tree Browser initialisiert wird oder nicht.

Um dies nun zu gewähren, wurde in die Klasse `Login` (jsp), bei gegebener Authentifizierung, eine Umleitung zur Klasse `tree` (jsp) eingebaut, in welcher das schon beschriebene Frameset mit der Baumansicht aufgebaut wird.

```
if( wikiSession.isAuthenticated()){
   viewUrl = wiki.getURLConstructor().makeURL(WikiContext.NONE,"",
   false,null)+"tree.jsp?redirect="+redirectPage+"&settree=true";
   response.sendRedirect(viewUrl);
}
```

Listing 6: Umleitung nach Authentifizierung zu tree.jsp in Login.jsp

Wie in Listing 6 zu sehen, enthält die URL zur Umleitung die zwei Parameter `redirect` und `settree`. Der Erste gibt die Seite an, welche als erstes im Hauptframe angezeigt werden soll. Durch den Parameter `settree` wird angegeben, ob die Baumansicht gestartet wird oder nicht. So ist es möglich, durch manuellen Eintrag der Adresse und das Setzen dieses Parameters auf `false` eine *Normalansicht* zu erhalten.
Um sicherzustellen, dass nichtregistrierte User bei händischer Eingabe der URL nicht zur Baumansicht gelangen, erfolgt auch in `tree` eine Abfrage der Registrierung.

Vor Erzeugung der Frames wird über die Funktion `isStarted()` erfragt, ob die Klasse `TreePrepare` bereits initialisiert ist oder dies gegebenenfalls noch durchgeführt werden muss (siehe Listing 7).

```
if(TreePrepare.isStarted() == false){
      log.info("Initialize Tree");
      TreePrepare tree = new TreePrepare( wiki );
}
```

Listing 7: Abfrage und Initialisierung von TreePrepare in tree.jsp

Diese Abfrage garantiert, dass die Initialisierung serverseitig nur einmal durchgeführt wird und nicht bei jedem neuen User wiederholt wird. Bei der Initialisierung dieser Klasse werden einige wichtige Konstanten vom System abgerufen und gesetzt. So wird das anfängliche Grundgerüst der `<UL><LI>`-Liste mit dem Root-Element erstellt, der Pfad zum *Home*-Ordner der Textdateien für die Wiki-Seiten sowie deren Anhänge und die URL zur Startseite abgefragt und in einer Variable gespeichert. Der Hintergrund Thread `PageDeleteUpdater` (java), welcher eine innere Klasse von `TreePrepare` (java) ist und zum Löschen von Kategorien mit mehreren Unterseiten bzw. Unterkategorien im Hintergrund verwendet wird, wird ebenfalls gestartet. Zuletzt wird noch die Variable `started = true` gesetzt, um die nun erfolgte Initialisierung zu signalisieren.

Die Klasse `drag-drop-folder-tree` (jsp) ist für die Erzeugung der Baumansicht zuständig. Initial wird die in `TreePrepare` (java) erstellte und gespeicherte Grundstruktur des Baumes mit der Funktion `getXML(Home)` geholt. Folgend wird die Klasse `JSDragDropTree` (js) initialisiert. Dabei werden wichtige URLs zu den für die Kommunikation mit dem Server benötigten JSP-Klassen, den weiteren JavaScript-Klassen und den Icons gesetzt. Anschließend wird die Usergruppe sowie die ID des Baumes gesetzt und alle benötigten Drop-Down Menüs initiiert, welche durch Drücken der rechten Maustaste auf ein Listenelement sichtbar werden und für verschiedene Dateitypen und User verschiedene Optionen zur Verfügung stellen.
Als letzter Schritt der Initialisierung soll noch die letzte Baumansicht wieder hergestellt werden. Hierfür wird im *Cookie* nachgeschaut, ob die in der letzten Darstellung erweiterten Knoten eingetragen wurden. Falls ja, werden diese in der folgenden Tree-View wieder erweitert dargestellt.

5.5.2.4 Navigieren, Erweitern und Zuklappen

Die Implementation eines Tree Browsers bedeutet aufgrund der dadurch zur Verfügung gestellten Möglichkeit der Hierarchischen Suche (siehe Kap. 1.4 Hierarchische Suche) auch eine Erleichterung der Navigation durch das Angebot des Systems. Die Artikel werden durch Anklicken eines Listenelementes im rechten Frame angezeigt.

Um dies zu unterstützen, ist das Erweitern und Zuklappen von einzelnen Kategorien mit Unterkategorien bzw. Unterseiten eine sehr wichtige Funktion. Der prinzipielle Ablauf des Erweiterns eines Knotens ist im unteren Sequenzdiagramm gezeichnet.

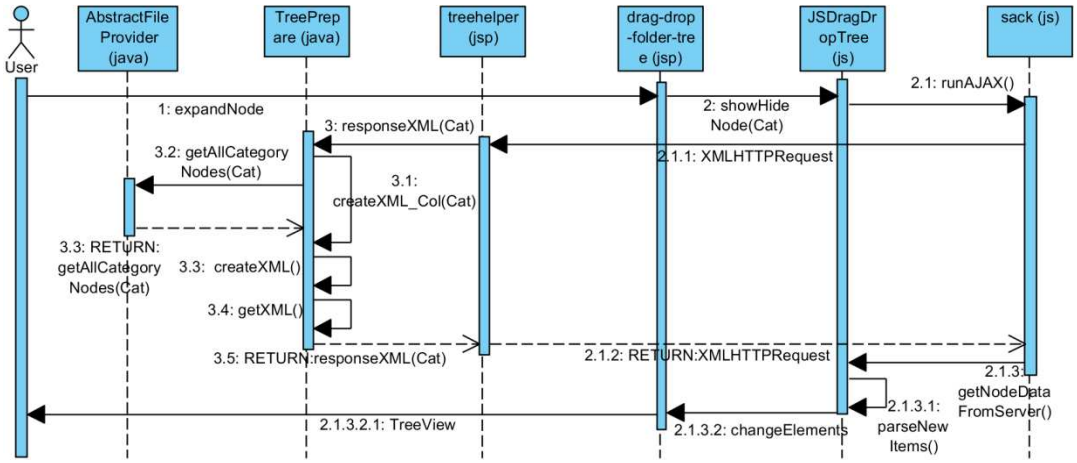

Abbildung 66: Sequenzieller Ablauf - Erweitern einer Kategorie

Vorgabe hier war, dass die Daten beim Erweitern on demand vom Server geladen werden.

Wie in Listing 4 aus Kapitel 5.5.1.3 ‚Aufbau eines Listenelementes <LI>' ersichtlich, wird die Erweiterung oder das Zuklappen eines Knotens durch einen Mausklick auf das ‚+' bzw. ‚-' -Icon gestartet und die Funktion `showHideNode()` in der Klasse `JSDragDropTree` (js) aufgerufen. Dieser Funktion wird das aufrufende Element übergeben. Hieraus lassen sich notwendige Informationen lesen. So ist es wichtig zu erfahren, um welchen Dateityp es sich handelt (Kategorie, Seite mit Anhang oder der Container für die Anhänge), welchen Zustand das Element hat (erweitert oder eventuell doch zugeklappt) und natürlich die ID, um zu wissen welches Element die Anforderung gestellt hat.

Der Dateityp und der Zustand des Elementes lassen sich aus den Icons ablesen. Die ID, welche in unserem Fall der Name der Kategorie ist, wird dem Element als Attribut mitgegeben und die URL kann aus dem `<href>`-Tag generiert werden.

Wie schon angesprochen ist unser Baum als `<UL><LI>`-Liste aufgebaut. Nehmen wir eine stark vereinfachte Liste her, um den grundsätzlichen Vorgang bildlich darzustellen.

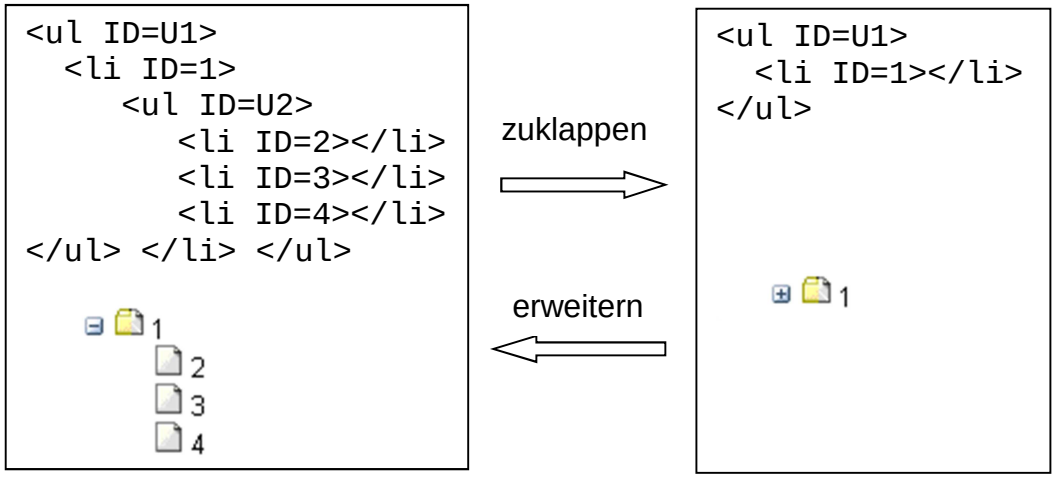

Abbildung 67: Links: Baum mit erweiterten Listenelement - Rechts: Baum mit zu zugeklappten Listenelement

Wie in Abbildung 67 zu sehen, ist einem erweiterten Listenelement einfach das beinhaltende <UL>-Element wegzunehmen bzw. auszublenden um in den zugeklappten Zustand zu kommen und umgekehrt wieder einzublenden bzw. ein neu generiertes <UL>-Element an der Stelle des alten Elementes hinzuzufügen.

Fahren wir in der Funktion `showHideNode()` fort um dies nun genauer zu erklären. Mittlerweile sind die wichtigsten Parameter bekannt, so auch der momentane Status des Listenelementes anhand des ‚+ -' Icons durch folgende Abfrage:

```
var img = obj.getElementsByTagName('IMG')[0];
if(img.src.indexOf(plusImage)>=0){
     ……show <UL>-ITEM ……
     img.src = img.src.replace(plusImage, minusImage);
}else{
     ……hide <UL>-ITEM……
     img.src = img.src.replace(minusImage, plusImage);
}
```

Listing 8: Statusabfrage des Listenelementes

Ergibt die Abfrage *FALSE*, wird das Listenelement zugeklappt:

```
var ul = obj.getElementsByTagName('UL')[0];
if (ul) {
     ul.style.display = 'none';
}
```

Listing 9: Zuklappen eines Knoten mit style Objekt

Durch die *style.display*-Eigenschaft können Elemente ausgeblendet werden, ohne einen Platzhalter zu erhalten. Das bedeutet, dass das Element zwar noch vorhanden ist, jedoch unsichtbar geschaltet wird und der Platz im Dokument nicht reserviert ist, sondern das folgende Objekt nachrückt, also genau der hier gewünschte Zustand. Das zu diesem Zeitpunkt angezeigte ‚-' Icon ist dann noch mit dem ‚+' Icon zu ersetzen.

Würden wir bei der oben formulierten Abfrage *TRUE* erhalten, könnte das Element bei einem statischen Baum durch `ul.style.display = 'block'` wieder eingeblendet werden. Die Kind-Knoten sollen bei jedem Erweitern jedoch neu vom Server geladen werden, um auf dem aktuellen Stand zu sein. Hier kommt nun AJAX ins Spiel:

```
ajaxIndex = ajaxIndex+1;
ajaxArray[ajaxIndex] = new sack();
ajaxArray[ajaxIndex].requestFile = RequestFile+
     '&parentId='+parentId+'&atts='+atts+'&what=1');
ajaxArray[ajaxIndex].onCompletion = function(){
     getNodeDataFromServer(ajaxIndex,ul.id,parentId);};
ajaxArray[ajaxIndex].runAJAX();
```

Listing 10: Beispiel eines XMLHTTP-Requests mittels AJAX zur Generierung der Kindknoten einer Kategorie

Für alle Ajax-spezifischen Funktionen ist die Klasse `sack` (js) zuständig. Um sich den Diensten dieser Klasse zu bedienen, wird ein Objekt dieser Klasse initialisiert. Der Variable `requestFile` wird der Link zur weiterverarbeitenden JSP-Klasse übergeben. In diesem Fall ist dies die Klasse `treehelper` (jsp). Diese Klasse verteilt alle *Requests* des Clients und benötigt dazu die Variable `what`, welche im Link, also in der Variabale

requestFile mitgegeben wird und durch den Befehl request.getParameter am Ziel erfragt werden kann. Dem Objekt onCompletion wird dann noch die Funktion übergeben, welche die Antwort des Servers verarbeiten soll. Mit dem Funktionsaufruf runAjax() wird die Abfrage schlussendlich gestartet. Das Objekt wird, wie in Listing 10 ersichtlich, in einem Array mit dem Index *ajaxIndex* gespeichert. Dieser Index und das Array sind globale Variablen. Der Index wird bei jedem Aufruf um eins inkrementiert. Durch die asynchrone Arbeitsweise ist dies notwendig, da es durchaus vorkommen kann, dass vor Beendigung eines Auftrages ein weiterer Auftrag ansteht, der das vorhandene Objekt überschreiben würde und so falsche Ergebnisse geliefert werden. Auch hier wird dann noch das ‚+' Icon mit dem ‚-' Icon ersetzt.

Weiter geht's am Server. Die Abfrage des Clients wird von der Klasse treehelper (jsp) zur Funktion responseXML() der Klasse TreePrepare (java) weitergleitet. Übergabeparameter sind die ID, also der Name der Kategorie und der Dateityp, da es je nach Typ Unterschiede bei der Erstellung des Rückgabestrings und den aufzurufenden Funktionen gibt. Um nun diesen String (in unserem Falls die <UL><LI>-Liste) erstellen zu können müssen die Kindknoten abgerufen werden.
Handelt es sich um Anhänge die angefordert werden, wird eine ArrayList durch die in der Klasse AustriaForumAttachmentProvider (java) bereits vorhandenen Methode listAttachments()erstellt. Diese wird dann durch die Methode createAttachesXML(), wieder eine Methode der Klasse TreePrepare (java), in die gewünschte Form gebracht und anschließend durch die Klasse treehelper (jsp) wieder dem Client zur weiteren Verarbeitung retourniert.

```
Element att = new Element("li");
att.setAttribute("id", "atts-"+dir);
att.setAttribute("parentId", dir);
att.setAttribute("atts", "true");
```

```
Element plusImg = new Element("img");
plusImg.setAttribute("src", imageFolder + plusImage);
plusImg.setAttribute("style","visibility: hidden");

Element loadImg = new Element("img");
loadImg.setAttribute("src", imageFolder + attImage);

Element a_att = new Element("a");
a_att.setText(attName[attName.length-1]);
att.addContent(plusImg);
att.addContent(loadImg);
att.addContent(a_att);
root.addContent(att);
```

Listing 11: Erstellung eines Listenelementes mit Hilfe der Klasse org.jdom am Server

Um die Knoten einer Kategorie oder eines Ordners abzurufen, wurde in der Klasse `AbstractFileProvider` (java) die Methode `getAllCategoryNodes()` implementiert. Diese Methode durchscannt die Kategorie bzw. den Ordner auf File System Ebene nach Sub-Ordnern oder Dateien bzw. Sub-Kategorien oder Seiten und erstellt zuerst drei Listen (ArrayList). Eine Liste enthält alle Sub-Kategorien, in der zweiten werden alle Sub-Ordner abgelegt und in der dritten alle Sub-Dateien. Diese werden noch alphabetisch geordnet und anschließend zusammengefasst in einer ArrayList retourniert. Durch die Aufteilung der Typen erst in eigenen Listen und das nachherige zusammenfassen ergibt eine getrennte Ansicht im Tree Browser. So werden alle Kategorien alphabetisch an der Spitze des Baumes angezeigt. Danach kommen alle Folder alphabetisch geordnet an die Reihe und am unteren Ende alle Dateien, natürlich auch alphabetisch geordnet. Diese ArrayList wird durch die Funktion `createXML_Col()` noch in die gewünschte `<UL><LI>`-Listenform gebracht und dem Client via `treehelper` (jsp) zurückgeschickt.

Der Response des Servers wird in der Funktion `getNodeDataFromServer()` verarbeitet. Dieser werden auch die ID des zu ersetzenden `<UL>`-Elementes und der Index des Speicherplatzes unseres AJAX-Objektes im Array übergeben um das richtige Element im Baum zu ersetzen. Durch `parseNewItems()`, dem ersten Schritt der clientseitigen Ge-

nerierung eines neuen Listenelemetes (siehe Kapitel 5.5.1.3), werden dem neuen Knoten dann noch die ersten Attribute hinzugefügt.

```
var response = ajaxArray[ajaxIndex].response;
var ul = document.getElementById(ulId);
ul.innerHTML = response;
ul.style.display = 'block'
parseNewItems(ul);
```

Listing 12: Einfügen des neuen Elementes

5.5.2.5 Interface zur Durchführung von Operationen

Um Operationen an Dateien, Kategorien und Ordnern durchführen zu können, war es notwendig, ein Interface zu schaffen, das dies dem User ermöglicht. Dieses System stellt zwei Arten zur Verfügung. Das Erste ist ein Drop Down Context Menü, das durch Klick der rechten Maustaste auf ein Listenelement erreicht wird. Als zweites besteht noch die Option, Elemente per Drag and Drop zu verschieben.

5.5.2.5.1 Drop Down Context Menü

In Kapitel 2.1.1.4 sowie 2.1.2.5 wurden bereits die wichtigsten Operationen aufgezählt, die in heutigen modernen Betriebssystemen auf Dateien und Ordnern durchführbar sein sollen. Laut Vorgabe soll es die Möglichkeit geben, Dateien und Ordner zu löschen, umzubenennen und zu verschieben. Des Weiteren soll es für verschiedene Usergruppen verschiedene Optionen geben. So soll ein User der Gruppe Administrator alle Optionen erhalten, die auf den jeweiligen Dateityp durchführbar sein sollen. Ein normaler User soll jedoch vorerst nur die Möglichkeit bekommen, eine neue Seite in einer bereits bestehenden Kategorie oder im Homeverzeichnis erstellen zu können. Die folgenden Tabellen geben Aufschluss, was welchem User ermöglicht werden soll.

Legende: A = Admin/Editor, U = User, nv = nicht verfügbar

Dateityp / Operation	Kategorie	Ordner	Page	Container Anhänge	Anhang
Erstellen	A	A	A/U	nv	nv
Umbenennen	A	A	A	nv	A
Verschieben	A	A	A	A	A
Kopieren	A	A	A	A	A
Öffnen*	A/U	A/U	A/U	A/U	A/U
Löschen	A	A	A	A	A

Tabelle 6: Mögliche Operationen auf Dateien oder Verzeichnisse in der Baumansicht für User

*Das Anzeigen von Pages ist grundsätzlich für jeden möglich. Sollte eine Seite nicht für jeden zugänglich sein wird dies vom JSPWiki-Security Modul intern geregelt und anstatt der Seite eine Warnung ausgegeben.

Die folgende Tabelle zeigt, welcher Dateityp in welcher und von welcher Usergruppe erstellt werden darf.

Legende: A = Admin/Editor, U = User, nv = nicht verfügbar

in welcher	Kategorie	Ordner	Page	Container Anhänge	Anhang
Ordner	A	A	A	nv	nv
Page	A/U	A/U	A	nv	nv

Tabelle 7: Welche Dateitypen können in welchem Dateitypen erstellt werden

Folgend noch die Tabelle, welche zeigt, welcher Dateityp von welcher U-sergruppe in welchem Dateityp verschoben oder kopiert werden darf.

Legende: A = Admin/Editor, U = User, nv = nicht verfügbar

von \ nach	Kategorie	Ordner	Page	Container Anhänge	Anhang
Kategorie	A	A	A	nv	nv
Ordner	A	A	A	nv	nv
Page	A	A	A	Nv	nv
Container Anhänge	A	nv	A	A	A
Anhang	A	nv	A	A	A

Tabelle 8: Welche Dateientypen können in welchen Dateityp kopiert oder verschoben werden

Anhand dieser Tabellen wurden die verschiedenen Drop Down Context Menüs mit den jeweiligen Optionen erstellt, die beim Start des Baumes in der Funktion `initTree()` der Klasse `JSDragDropTree` (js) initialisiert werden.

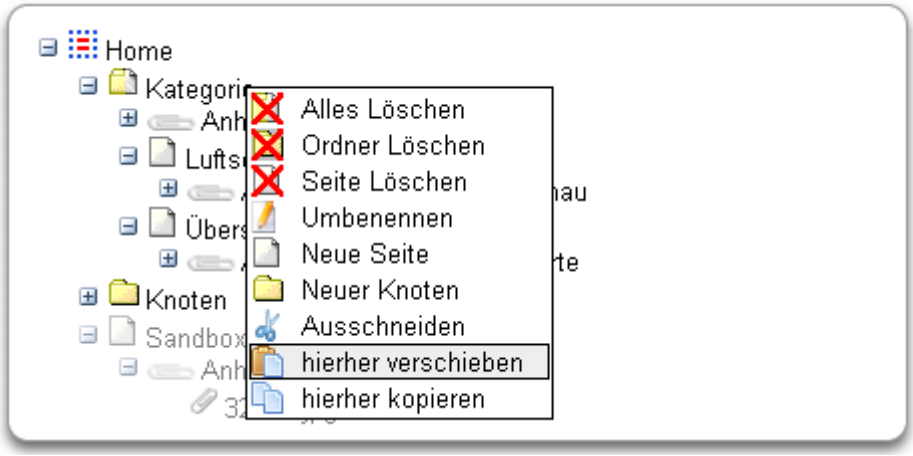

Abbildung 68: Das in Listing 13 erstellte Drop Down Menü nach Klick der rechten Maustaste auf eine Kategorie

```
menuModel = new dragdrop_menuModel();
menuModel.addItem (no++,'AllesLöschen',delbothImg,'',false, 'de-
lete_both');
menuModel.addItem(no++,'OrdnerLöschen', delfolderImg,'',false,
'delete_folder');
menuModel.addItem(no++,'SeiteLöschen', delpageImg,'',false, 'de-
lete_page');
```

```
menuModel.addItem(no++,'Umbenennen', renameImage,'',false,
'renameItem');
menuModel.addItem(no++,'NeueSeite', pageImage,'',false, 'new-
Page');
menuModel.addItem(no++,'NeuerKnoten', folderImage_close,'',false,
'newNode');
menuModel.addItem(no++,'Ausschneiden', cutImg,'',false,'initCut');
menuModel.addItem(no++,'hierherverschieben', pasteImg,'',false,
'initPaste');
menuModel.addItem(no++,'hierherkopieren', copyImg,'',false,
'initPaste_copy');
this.menuModel.init();
```

Listing 13: Initialisierung des Drop Down Menüs für Administratoren für den Typ Kategorie und befüllter Zwischenablage in der Funktion initTree() wie es in Abbildung 68 zu sehen ist

Bei der Initialisierung werden den einzelnen Optionselemente, die Parameter ID, Bezeichnung, Icon und die bei einem Aufruf auszuführende Funktion übergeben (siehe Listing 13).

In Abbildung 69 ist der prinzipielle sequenzielle Ablauf beim Klick der rechten Maustaste (in diesem Fall auschneiden und verschieben) auf ein Listenelement zu sehen.

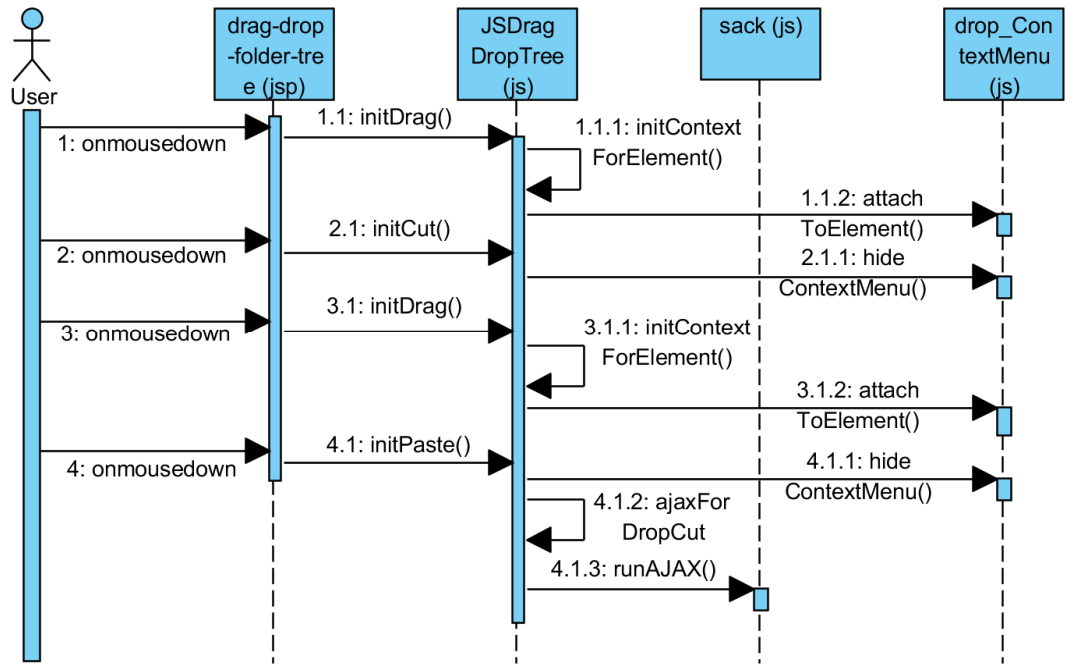

Abbildung 69: Sequenzieller Ablauf für Aufruf des Drop Down Menüs für die Option „Ausschneiden und hierher Verschieben“

Für die Anzeige und Funktion des Drop Down Menüs ist die JScript-Klasse `drag_drop_contextMenu` (js) zuständig. Diese enthält noch die Klasse `dragdrop_menuModel` (js), welche Methoden zur Initialisierung und Modellierung des Menüs zur Verfügung stellt (siehe Listing 13). Durch ein `onmousedown` Event wird die Funktion `initDrag()` aufgerufen. Nach Abfrage um welche Maustaste es sich handelt, wird im Falle der rechten Maustaste `initContextForElement()` aufgefordert dem gerade ausgewählten Element für den jeweiligen User das richtige Menü zur Verfügung zu stellen (siehe Listing 14). Dazu wird die Funktion `attachToElement()` in der Klasse `drag_drop_contextMenu` (js) aufgerufen, die das richtige Modell über die Funktion `_displayContextMenu()` in der Position des Mauszeigers zur Anzeige bringt (siehe Listing 15). Wird nun z.B. die Option Ausschneiden gewählt, wird das Menü wieder ausgeblendet (`hideContextMenu()`) und das Objekt in die Zwischenablage (`JSTreeObj.cutContainer`) kopiert. Bei wiederholtem Klick auf die rechte Maustaste (ab Schritt 3 im Sequenzdiagramm) beginnt das Ganze von vorne, nur werden aufgrund der gefüllten Zwischenablage im Menü zusätzlich die Optionen *hierher verschieben* und *hierher kopieren* angezeigt.

```
var imgsrc = obj.parentNode.getElementsByTagName('IMG')[1].src;
if (imgsrc.indexOf(JSTreeObj.both)>=0){
   if (JSTreeObj.cutContainer == null) {
      if(user.indexOf('admin')>=0){
       contextMenu.attachToElement(obj,false,menuModelBoth);
      }else{
       contextMenu.attachToElement(obj,false,menuModelBoth_user);
      }
  }else{
      if(user.indexOf('admin')>=0){
       contextMenu.attachToElement(obj,false,menuModelBoth_paste);
      }else{
```

```
  contextMenu.attachToElement(obj,false,menuModelBoth_user_paste);
}}}
```

Listing 14: Beispiel für die Auswahl des richtigen Drop Down Menüs für den Dateityp Kategorie in der Funktion initContextForElement().

```
if(document.all)e = event;
var ref = referenceTodragdropContextMenu;
ref.menuObject.style.left = e.clientX + 'px';
ref.menuObject.style.top = e.clientY + 'px';
ref.menuObject.style.display = 'block';
```

Listing 15: Codeschnipsel aus _displayContextMenu(), der die linke obere Ecke des Drop Down Menüs am Punkt des auslösenden Events (onmousedown) zur Anzeige bringt.

5.5.2.5.2 Drag and Drop

Durch Drag and Drop besteht die Möglichkeit ganze Kategorien, Dateien oder Anhänge zu verschieben. Zwar ist diese Operation auch über das Drop Down Menü möglich, jedoch mit vier Klicks, im Gegensatz zu einem `onmousedown` und einem `onmouseup` also einem Klick, über Drag and Drop. Die Ansicht während des Verschiebens ist in Abbildung 70 zu sehen.

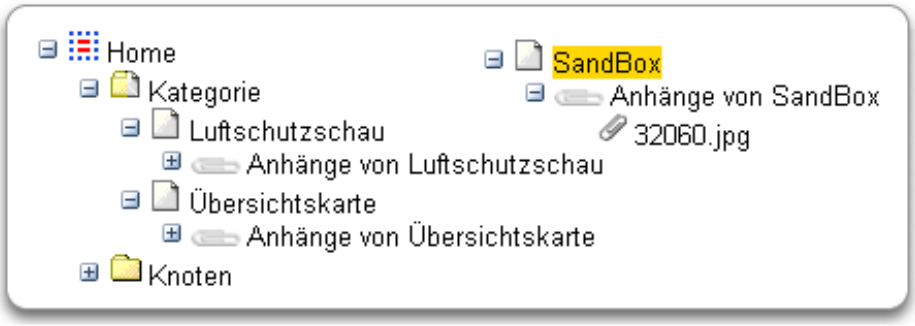

Abbildung 70: Ansicht während des Verschiebevorganges

Schon bei der Initialisierung des Baumes wird eine quasi Zwischenablage (`floatingContainer` als `<UL>`-Liste) erzeugt, welche die zu verschiebenden Elemente während des Vorganges beinhaltet. Sie wird dem Dokument angehängt und vorerst unsichtbar geschalten (siehe Listing 16).

```
this.floatingContainer = document.createElement('UL');
this.floatingContainer.style.position = 'absolute';
this.floatingContainer.style.display='none';
this.floatingContainer.id = 'floatingContainer';
document.body.appendChild(this.floatingContainer);
```

Listing 16: Initialisierung einer unsichtbaren leeren <UL>-Liste als Zwischenablage für die zu verschiebenden Elemente

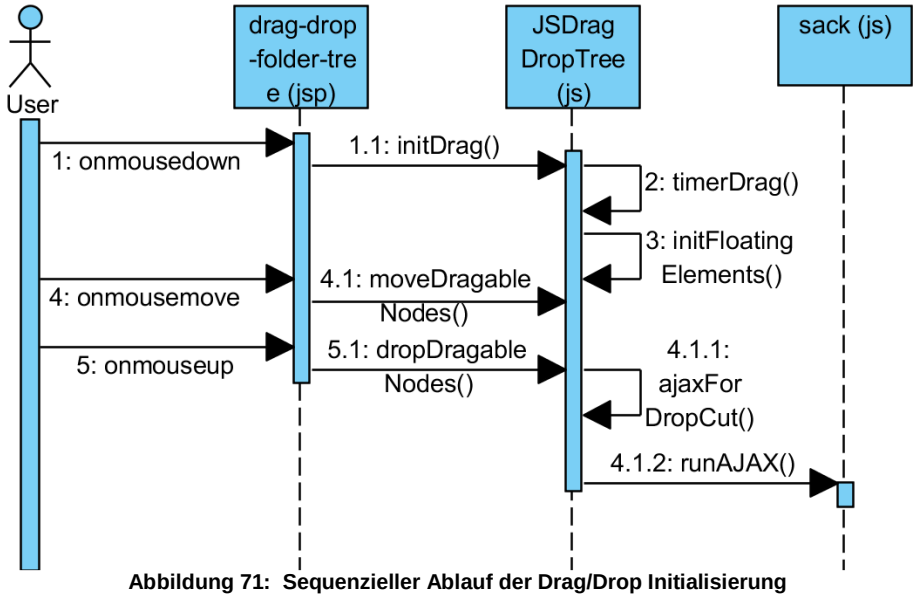

Abbildung 71: Sequenzieller Ablauf der Drag/Drop Initialisierung

In Abbildung 71 ist der sequenzielle Ablauf eines Drag and Drop-Vorganges zu sehen. Nachdem die Maus gedrückt gehalten wird, (`onmousedown`) wird der Drag-Vorgang initialisiert. Über die Funktion `timerDrag()` wird eine Verzögerung erzeugt, um bei einem einfachen Mausklick nicht gleich in einen Verschiebevorgang zu geraten. Nach Ablauf dieser Zeit wird das angeklickte Element in den `floatingContainer` kopiert (`initFloatingElements()` - Listing 17). Wird die Maus nun bewegt (`onmousemove`) wird das Element sichtbar geschalten und solange an den Mauszeiger gehängt (`moveDragableNodes()` - siehe Listing 18), bis die Maustaste wieder losgelassen wird. Sollte das Loslassen der Taste nicht auf einem Element des Tree Browsers passieren, wird es wieder an die ursprüngliche Position zurückgesetzt, andernfalls wird

der im nächsten Kapitel beschriebene Einfüge-Vorgang eingeleitet (`drop-DragableNodes()`).

```
this.floatingContainer.appendChild(dragNode_source);
```

Listing 17: Kopieren des Source Elementes in den „floatingContainer“ in der Funktion initFloatingElements()

```
if(document.all) e = event;
this.floatingContainer.style.left = e.clientX + 'px';
this.floatingContainer.style.top = e.clientY+ 'px';
this.floatingContainer.style.display = 'block';
```

Listing 18: Anhängen des „floatingContainer“ an Mauszeiger und sichtbar machen während die Maus gedrückt gehalten und bewegt wird in der Funktion moveDragableNodes()

5.5.2.6 Umbenennen, Verschieben und Kopieren

Für diese Operationen wurden die Klassen `PageCopier` (java), `Copy` (jsp) und `Rename_tree` (jsp) hinzugefügt sowie Änderungen in der Klasser `PageRenamer` (java) durchgeführt. Sie bilden sozusagen das Pendant zu den Klassen `treehelper` (jsp) und `TreePrepare` (java) nur übernehmen `Copy` (jsp) und `Rename_tree` (jsp) hier die Steuerung der Requests des Clients, welche zur Bearbeitung an die Klassen `PageCopier` (java) und `PageRenamer` (java) vermittelt werden.

5.5.2.6.1 Verschieben und Umbenennen

Das *Verschieben* ist im Grunde einfach ein Ändern bzw. Umbenennen des Pfades bei gleichbleibendem Dateinamen, demgegenüber ist das *Umbenennen* ein Ändern bzw. Umbenennen des Dateinamens bei gleichbleibendem Pfad. Sieht man den ganzen Pfad mit Dateiname als Namen einer Datei, kann beides unter den Begriff *Umbenennen* zusammenfasst werden. Deshalb hat es auch technisch größten Teils die gleiche Abhandlung, nur das visuelle Interface unterscheidet sich, also die Darstellung dieser beiden Operationen dem User gegenüber.

Das Verschieben ist wie in Kapitel 5.5.2.5 schon beschrieben über das Drop Down Menü und Auswahl der Option *hierher verschieben* oder über Drag and Drop erreichbar.
Umbenennen eines Dateinamens wird durch Auswahl der Option Umbenennen ebenfalls im Drop Down Menü aufgerufen. Danach wird das ausgewählte Element als Textfeld dargestellt, welches den aktuell bestehenden Dateinamen beinhaltet (siehe Abbildung 72). Nach Eingabe des gewünschten Namens wird durch Drücken der ENTER-Taste die Aktion gestartet oder durch Drücken der ESC-Taste abgebrochen (siehe Listing 19).

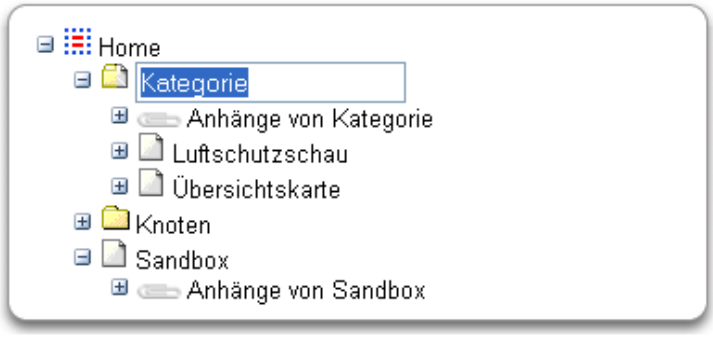

Abbildung 72: Darstellung des Textfeldes zum Ändern des Dateinamens

```
if(document.all)e = event;

if(e.keyCode==13||e.keyCode==9){    // Enter pressed
      JSTreeObj.__saveTextBoxChanges(false,this);
}
if(e.keyCode==27){      // ESC pressed
      JSTreeObj.__cancelRename(false,this);
}
```

Listing 19: Abfrage der gedrückten Taste nach der Eingabe im Textfeld zum Umbenennen von Dateien

Der weitere sequenzielle Ablauf des Verschiebens per Drop Down Menü oder Drag and Drop und des Umbenennens einer Datei ist in Abbildung 73 zu sehen.

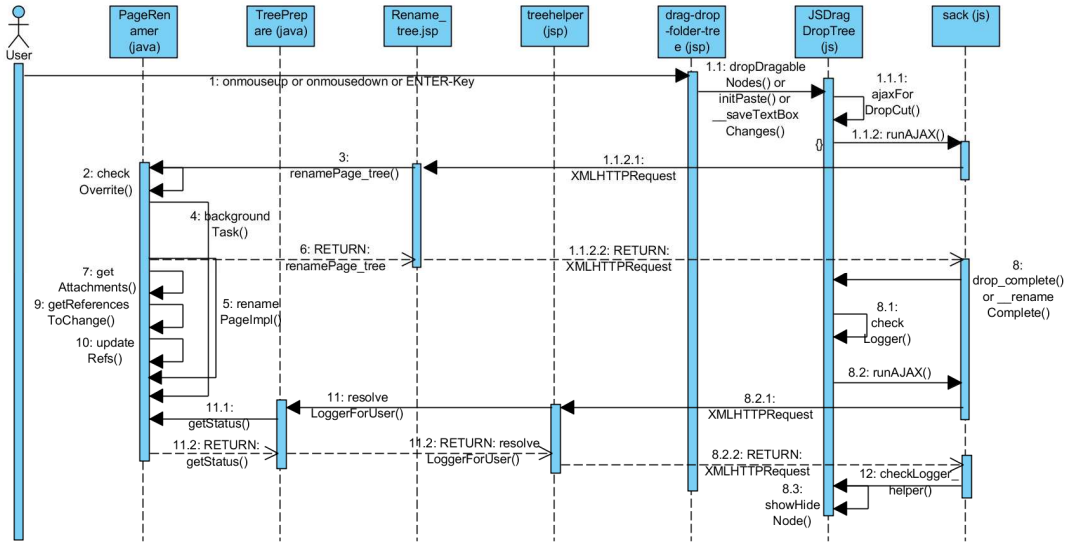

Abbildung 73: Sequenzieller Ablauf Verschieben per Drag/Drop oder Ausschneiden und Einfügen mit Drop Down Menu

Erstes Kriterium hier ist die eventuelle Namensgleichheit der zu verschiebenden Datei mit einer bereits im Ordner vorhandenen Datei. Zusätzlich war beim Verschiebevorgang zu achten, dass Dateien oder Ordner nicht in einen eigenen Unterordner und nur auf kompatible Dateien verschoben werden (keine Dateien in einen Anhangs-Container). Weiters war zu überlegen, was beim Verschieben einer ganzen Kategorie zu tun ist, also ob die ganze Kategorie oder nur der Ordner mit den Unterdateien und Unterordnern verschoben werden soll. Die Kontrolle dieser Überlegungen wird teilweise am Client und teilweise am Server durchgeführt.

Nachdem über eine *Confirm-Message* noch einmal nachgefragt wird, ob die Operation tatsächlich ausgeführt werden soll werden in der Funktion `ajaxForDropCut()` einige Kontrollen abgearbeitet, z.B. ob die Datei nicht in sich selbst oder in einen Unterordner oder in den Anhang einer anderen Datei verschoben wird. Sollte dies eintreffen, wird der Vorgang abgebrochen.

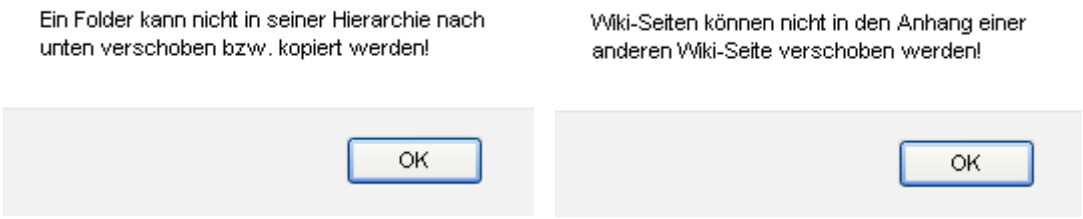

Abbildung 74: Mögliche Fehlerausgaben mit darauffolgendem Abbruch des Vorganges

Sollte das zu verschiebende Objekt eine Kategorie sein, wird noch abgefragt, ob die gesamte Kategorie verschoben wird oder nur der Ordner mit allen Unterseiten und Unterkategorien.

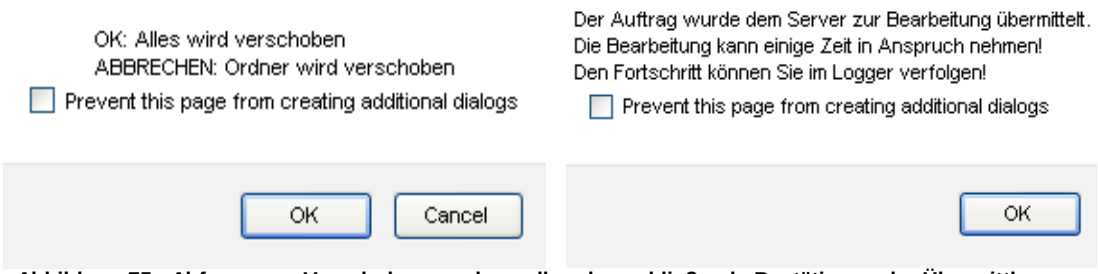

Abbildung 75: Abfrage was Verschoben werden soll und anschließende Bestätigung der Übermittlung an den Server

Der Auftrag wird anschließend an die JSP-Klasse `Rename_tree` (jsp) übermittelt wo er an die Klasse `PageRenamer` (java) geleitet wird. Diese Klasse besitzt drei innere Klassen.

Die Erste ist `RingList` (java), welche die Klasse `List` (java) implementiert. Diese Liste hat eine bei der Initialisierung zu bestimmende Kapazität. Wird beim Füllen diese erreicht, wird das erste eingefüllte Element gelöscht und dessen Platz kann wiederverwendet werden (FIFO).

Die Zweite ist `PageRenameLogger` (java). Diese benützt die vorhin beschriebene `RingList` (java) um mitgeloggte Statusmeldungen zu speichern, welche bei einem Umbenennen-Vorgang auftreten. Sie bietet verschiedene Funktionen an, um diese Ergebnisse abzurufen, sodass der Status über eine Anzeige mitverfolgt werden kann (siehe Abbildung 76).

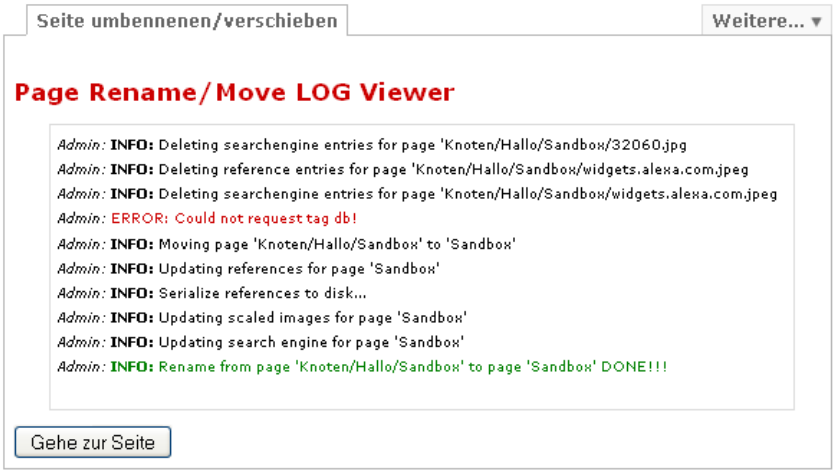

Abbildung 76: Statusanzeige beim Verschieben bzw. Umbenennen

Die dritte Klasse nennt sich `PageRenameUpdater` (java) und ist von `WikiBackgroundThread` (java) abgeleitet. Über sie wird im Hintergrund das Umbenennen durchgeführt.

Nochmal zurück, über `Rename_tree` (jsp) wird die Funktion `renamePage_tree()` in `PageRenamer` (java) aufgerufen. Hier wird kontrolliert ob der User das Recht hat, die Operation auf diese Datei bzw. in dem zu verschiebenden Verzeichnis durchzuführen. Des Weiteren wird kontrolliert, ob schon eine Datei mit gleichem Namen vorhanden ist. Der Funktion wird auch die Variable `force_overwrite` übergeben. Ist diese *true*, würde bei Namensgleichheit die vorhandene Datei überschrieben werden. Wird diese Operation vom Tree Browser ausgeführt, ist sie ständig *false*, somit wird der Vorgang bei Namensgleichheit immer abgebrochen. Sollten die Kontrollen in dieser Methode negativ ausfallen, wird dem Client eine Fehlermeldung geschickt (siehe Abbildung 77).

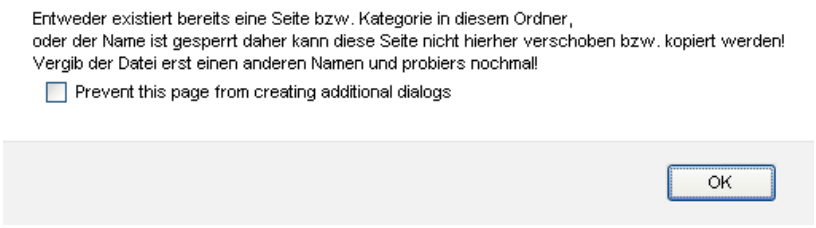

Abbildung 77: Fehlermeldung bei Namensgleichheit

Bei positivem Abschluss werden die Daten, der zu bearbeitenden Dateien, als `Vector` der Klasse `PageRenameUpdater` (java) übergeben, um diese so im Hintergrund zu bearbeiten. Der Client kontrolliert durch Abfragen des Loggers in einem bestimmten Intervall den Fortschritt des Vorganges. Das Beenden des Vorganges muss nicht abgewartet werden (kann bei Kategorien mit vielen Unterdateien lange dauern), sondern kann wieder weiterarbeiten, während am Server der Auftrag abgearbeitet wird. Dort wird über den *Background-Thread* die Funktion `renamePageImpl()` aufgerufen. In dieser Methode werden neben dem Namen der Datei auch die Referenzen (Verlinkung einer WikiSeite), alle Tags, alle Attribute sowie alle Anhänge der Seite bzw. aller Seiten der Kategorie geändert.

5.5.2.6.2 Kopieren

Der sequenzielle Ablauf eines prinzipiellen Kopiervorganges ist in Abbildung 78 ersichtlich.

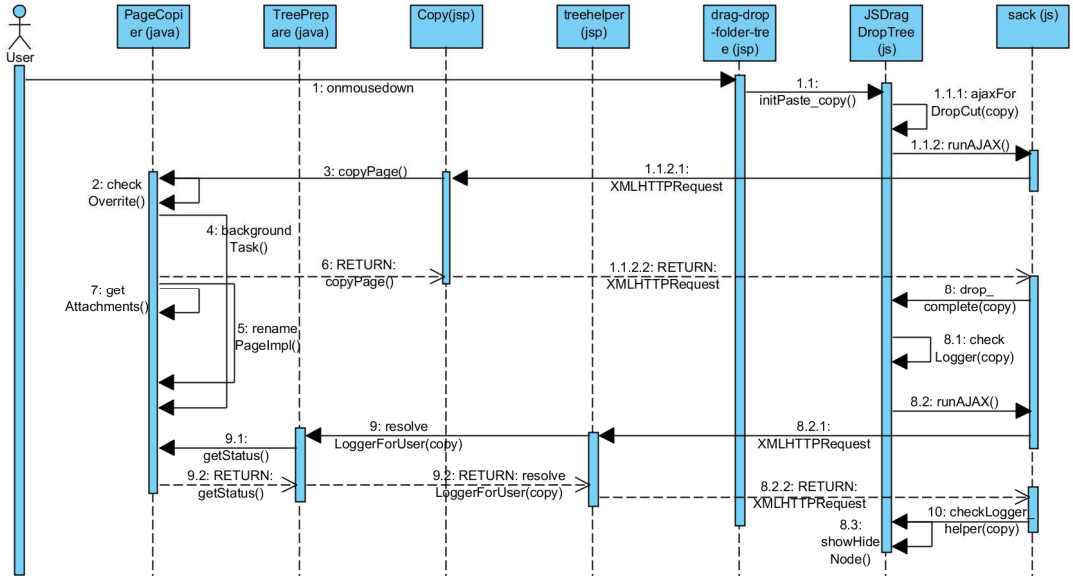

Abbildung 78: Sequenzieller Ablauf Kopieren und Einfügen mit Context Menu

Erreichbar ist diese Funktion über das Drop Down Menü. Kopieren ist technisch dasselbe wie das Verschieben, nur dass im letzten Schritt die Quell-Dateien nicht gelöscht werden, sowie die Referenzen, Attribute und

Tags der ursprünglichen Datei bzw. Dateien nicht geändert werden müssen. Der Ablauf wird serverseitig über die Klassen `Copy` (jsp) und `PageCopier` (java) geleitet.

Genauso wie Seiten, Kategorien oder Ordner, können auch Anhänge mittels Drop Down Menu oder Drag/Drop kopiert, umbenannt oder verschoben werden. Sie können alle gemeinsam durch Auswahl des Containers für Anhänge oder einzeln verschoben/kopiert werden.

5.5.2.7 Neue Dateien erstellen

Das System bietet über das Drop Down Menü auch die Möglichkeit neue Dateien zu erstellen. Dabei gibt es die Möglichkeit, eine neue Seite oder einen neuen Knoten (Ordner) zu erzeugen. Indirekt besteht auch die Möglichkeit neue Kategorien zu bilden.

5.5.2.7.1 Neue Seite

Der prinzipielle Ablauf zum Erstellen einer neuen Seite ist im Sequenzdiagramm in Abbildung 79 ersichtlich.

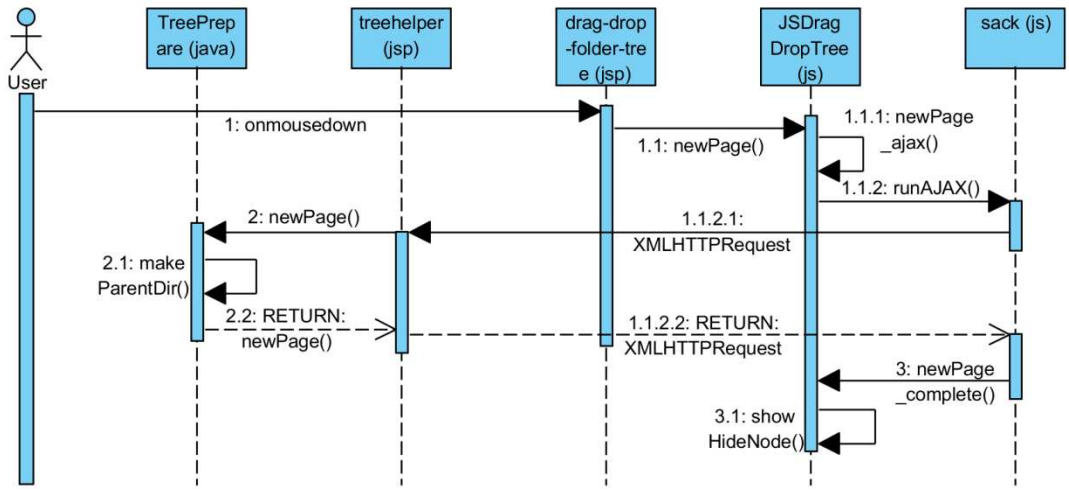

Abbildung 79: Sequenzieller Ablauf zum Erstellen einer neuen Seite

Nach Auswahl der Option *Neue Seite* im Drop Down Menü wird ein *Prompt* geöffnet, dass zur Eingabe eines Namens auffordert (Abbildung 80)

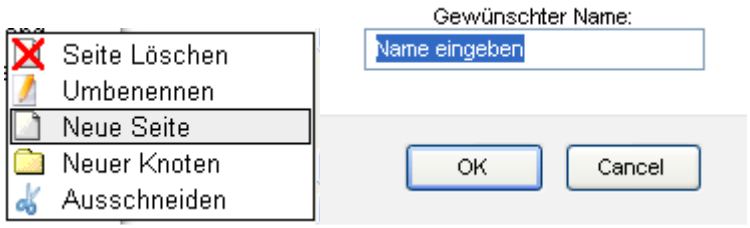

Abbildung 80: Drop Down und anschließende Aufforderung zur Namenseingabe

Nach Bestätigung wird in der Methode `newPage()` nach nicht erlaubten Sonderzeichen oder einem leeren Textfeld kontrolliert und anschließend die Daten für die Übermittlung an den Server via AJAX vorbereitet. In der Klasse `treehelper` (jsp) wird nach Überprüfung der Berechtigung des Users die Funktion `newPage()` in `TreePrepare` (java) aufgerufen, um eine neue Seite, also eine neue Datei auf Filesystemebene, zu erzeugen und anschließend eine Meldung dem Client retourniert. Im Ansichtsfenster öffnet sich bei positiv abgeschlossenem Vorgang das Fenster mit dem Editor, um die Texteingabe zu starten.

In der Funktion `newPage()` wird vor Erstellung des Files der Ordner nach bereits vorhandenen Dateien mit selben Namen und selben Typ kontrolliert. Sollte dies der Fall sein, wird der User aufgefordert, einen anderen Namen zu vergeben (Abbildung 81) und der Prozess von vorne gestartet.

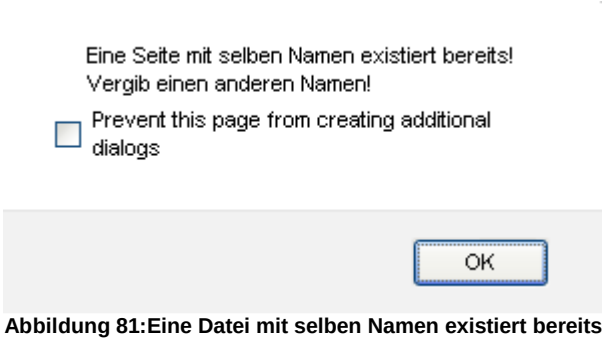

Abbildung 81:Eine Datei mit selben Namen existiert bereits

5.5.2.7.2 Neuer Knoten (Ordner)

Das Sequenzdiagramm des Ablaufes ist in Abbildung 82 dargestellt.

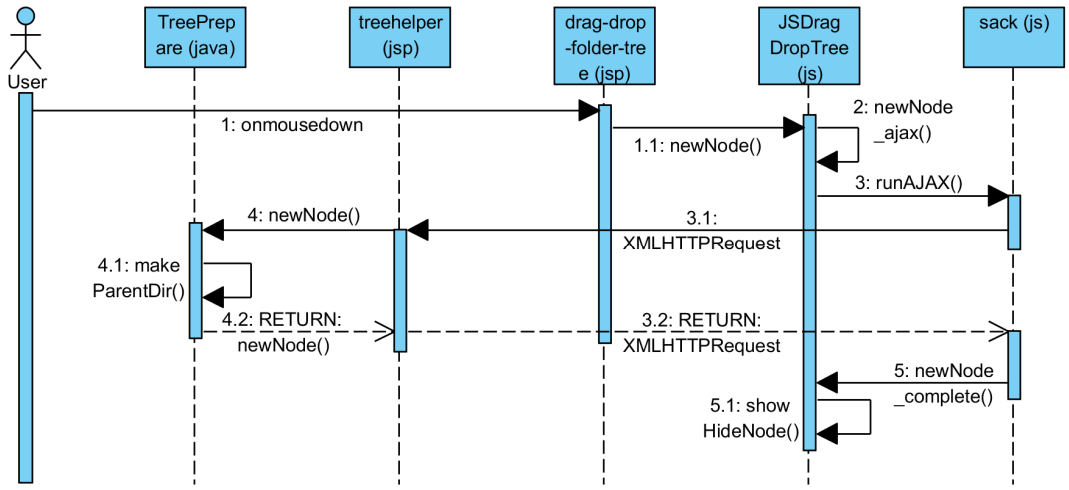

Abbildung 82: Sequenzieller Ablauf zum Erstellen eines neuen Ordners

Der Ablauf ist derselbe, wie vorhin bei der Erstellung einer neuen Seite bereits erklärt wurde, nur dass hier anstatt einer Seite bzw. einer Textdatei ein Ordner erstellt wird.

Auf Codeebene sind die Funktionen `newNode()`, `newNode_ajax()` clientseitig sowie `newNode()` und `makeParentDir()` in der Klasse `TreePrepare` (java) serverseitig für die ordnungsgemäße Durchführung der Aktionen zuständig.

5.5.2.7.3 Neue Kategorie

Um eine neue Kategorie zu erstellen gibt es mehrere Möglichkeiten:

- Neuerstellung einer Datei und eines Knotens (Ordner) mit selben Namen in einem Verzeichnis.
- Verschieben einer Seite auf eine andere Seite.
- Klicken auf einen Knoten (Ordner). Dadurch erscheint im Ansichtsframe die Meldung, dass eine Seite mit diesem Namen noch nicht existiert. Klickt man nun dort auf den Link *Erstellen* wird eine Seite mit selbem Namen erstellt und der Ordner wird somit eine Kategorie.
- Umbenennen einer Seite in den Namen eines Ordners im selben Verzeichnis oder umgekehrt

- Verschieben eines Ordners oder einer Seite in ein Verzeichnis, in welchem bereits ein Ordner oder eine Seite mit demselben Namen vorhanden ist.

5.5.2.8 Löschen von Dateien

Da das System die Option *Erstellen* zur Verfügung stellt, liegt es nahe, auch das *Löschen* bereit zu stellen. Erreichbar ist die Funktion wieder über das Drop Down Menü. Gelöscht werden können alle Dateitypen, vorausgesetzt man hat die nötigen Berechtigungen.

5.5.2.8.1 Löschen von Kategorien

Wie schon bekannt, ist der Typ Kategorie eine Mischung aus Datei und Knoten (Ordner). Daher kann man beim Löschen von Kategorien zwischen den drei Arten wählen:

- *Alles Löschen*
  Die komplette Kategorie wird gelöscht

- *Ordner (Knoten) Löschen*
  Nur der Ordner mit allen Unterkategorien und Unterseiten wird gelöscht

- *Seite Löschen*
  Nur die Seite wird gelöscht, der Ordner mit allen Unterkategorien und Unterseiten bleibt bestehen.

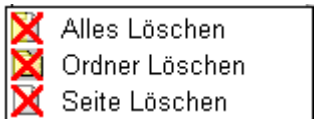

Abbildung 83: Mögliche Optionen zum Löschen einer Kategorie

Der sequenzielle Ablauf zum Löschen einer ganzen Kategorie ist aus dem Diagramm in Abbildung 84 ersichtlich.

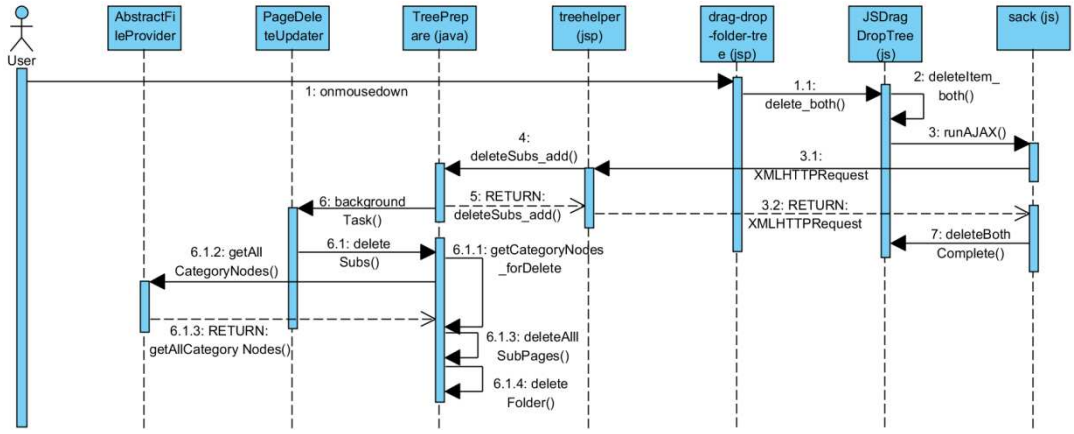

Abbildung 84: Sequenzieller Ablauf Löschen einer ganzen Kategorie

Initiiert wird der Vorgang durch Auswahl der Option im Drop Down Menü. Es erscheint ein Prompt, um nochmal zu vergewissern, ob wirklich alles gelöscht werden soll. Nach Bestätigung werden die Daten für die Erteilung des Auftrages in der Methode `deleteItem_both()` erstellt und mit AJAX dem Server gesandt. Nach Kontrolle der nötigen Berechtigung zum Löschen in der Klasse `treehelper` (js) wird der Auftrag der Funktion `deleteSubs_add()` in `TreePrepare` (java) weitergeleitet (Listing 20).

```
if(context.hasAdminPermissions()){
      log.info("Delete page and category "+parent+".
                User="+request.getRemoteUser()+",
                host="+request.getRemoteAddr());

      String result = tree.deleteSubs_add
                     (context,parent,true,false);

      if(!result.equals("OK")){
            String e =  result.split("Fehlermeldung:")
                        [result.split("Fehlermeldung:").length-1];
            log.error("ERROR while deleteing page and category
                      "+parent+" ERROR: "+e);
      }
}else{
      log.error("User: "+request.getRemoteUser()+
                " is not allowed to delete category "+parent);
      result = "You are not allowed to delete page and
                category "+parent;
}
```

Listing 20: Abfrage der Berechtigung, Start der Methode deleteSubs_add() und anschließender Response an den Client in der Klasse treehelper (jsp)

Der Funktion `deleteSubs_add()` wird der Context, der Namen der Kategorie, die Variable `parent` und die Variable `folder` übergeben. Die beiden letztgenannten sind vom Typ *boolean* und für die Festlegung des Löschtyps verantwortlich.

- `parent = true && folder = true`
  Die gesamte Kategorie wird gelöscht

- `parent = false && folder = true`
  Nur der Ordner mit allen Unterkategorien und Unterseiten wird gelöscht

- `parent = true && folder = false`
  Nur die Seite und darin eventuell vorhandene Anhänge werden gelöscht.

Die Variablen werden dort einzeln in einem `Object-Array` der Größe *4* gespeichert und dem `Vector m_updates` übergeben. Wie in Kapitel 5.5.2.3 erwähnt, besitzt `TreePrepare` (java) die innere Klasse `PageDeleteUpdater` (java), welche von der Klasse `WikiBackgroundThread` (java) abgleitet ist, sie läuft also im Hintergrund. Befinden sich nun ein oder mehrere Elemente in `m_updates`, wird über den BackgroundThread die Funktion `deleteSubs()` beauftragt, den Löschvorgang durchzuführen. Dafür werden über `getCategoryNodes_forDelete()` alle Unterdateien in einer `ArrayList` gespeichert und schließlich einzeln mit allen zugehörigen Referenzen und Anhängen in den Methoden `deleteAllSubPages()` sowie `deletePage()` gelöscht (siehe Listing 21). Abschließend wird noch der leere Ordner durch die Methode `deleteFolder()` gelöscht.

```
private void getCategoryNodes_forDelete(WikiContext context,String
page){

ArrayList set = null;
  try {
   set =
   (ArrayList)m_engine.getPageManager().getAllCategoryNodes(page);
   deleteAllSubPages(context,set);
  }catch (ProviderException e) {
         log.error("Can't get Nodes of Category: "+e.toString());

  }
}
```

Listing 21: Abfrage der Berechtigung, Start der Methode deleteSubs_add() und anschließender Response an den Client in der Klasse treehelper (jsp)

5.5.2.8.2 Löschen von Knoten (Ordnern)

Hier wird durch das Setzen der Schaltvariablen auf `- parent = false && folder = true -` veranlasst, dass nur das Löschen des Ordners durchgeführt wird. Die Abhandlung ist ansonsten gleich wie unter ‚Löschen von Kategorien' bereits erklärt.

5.5.2.8.3 Löschen von Seiten

Hier wird durch das Setzen der Schaltvariablen auf `- parent = true && folder = false -` veranlasst, dass nur die Seite gelöscht wird. Dadurch wird direkt die Funktion `deletePage()` vom Hintergrundprozess aufgerufen, welche für das Löschen einer Einzelseite zuständig ist.

5.5.2.8.4 Löschen von Anhängen

Das Löschen von Anhängen wird auch angeboten. Erreichbar ist diese Option bei Wahl eines Anhanges oder eines Containers für Anhänge und einem Klick auf die Option *Anhang Löschen* im Drop Down Menü.

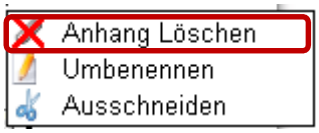

Abbildung 85: Mögliche Optionen zum Löschen einer Kategorie

Aus dem Sequenzdiagramm in Abbildung 86 sind die dafür zuständigen Klassen und Methoden ersichtlich.

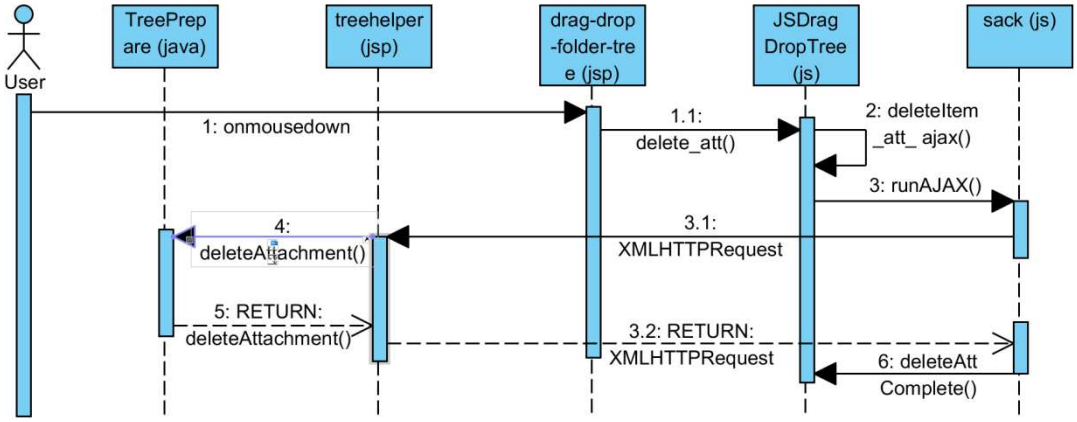

Abbildung 86: Sequenzieller Ablauf Löschen eines Anhangs

Die Anhänge werden nicht über den Hintergrund Thread gelöscht. Hier wird die dafür zuständige Funktion `deleteAttachment()` direkt aus der Klasse `treehelper` (js) aufgerufen, nach Abfrage der nötigen Berechtigungen.

5.5.2.9 Zusätzliche Features

5.5.2.9.1 Orientierung

In Kategorien mit vielen Artikeln ist es zur besseren Orientierung nützlich, die Position eines beliebigen Beitrages innerhalb der Hierarchie zu erfahren. Durch Auswahl der Option ‚Orientierung' im Drop Down Menü wird der ‚Ast' zur angewählten Datei visualisiert. Alle am Pfad liegenden Knoten sind verlinkt, um so auch direkt zu einer darüber liegenden Kategorie browsen zu können (siehe Abbildung 87).

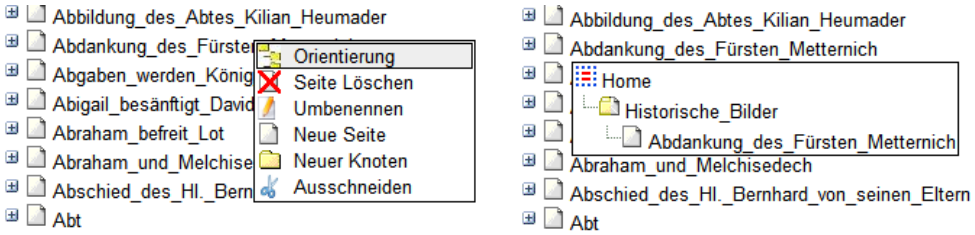

Abbildung 87: Rechts – Auswahl im Drop Down Menü, Links – Anzeige des Pfades bzw. Stellung in Hierarchie zur Orientierung

5.5.2.9.2 Tree-View Aus/Ein und Adressleiste

Wie bereits in Kapitel 5.5.1 beschrieben, bietet das System zur Orientierung eine Adressleiste im oberen Fenster an. Zusätzlich besteht die Möglichkeit, die Baumansicht auszublenden, komplett auszuschalten oder in einem eigenen Fenster darzustellen (siehe Abbildung 88).

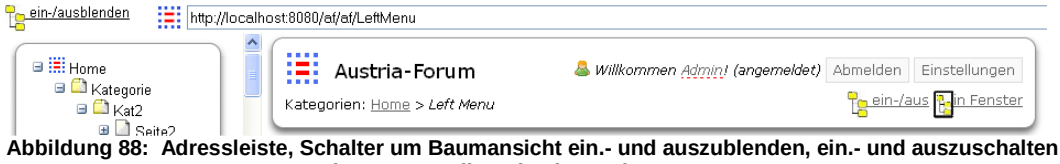

Abbildung 88: Adressleiste, Schalter um Baumansicht ein.- und auszublenden, ein.- und auszuschalten sowie zur Darstellung in einem eigenen Fenster

5.5.3 Cross Browser Kompatibilität

Die meisten Browser können mit JavaScript umgehen, nur sind die Implementierungen teilweise völlig unterschiedlich gelöst. Um dennoch auf den gängigsten Browsern wie FireFox, Internet Explorer oder Opera zu funktionieren, mussten bei manchen Funktionen die Browser erfragt werden um dafür eine funktionierende Lösung zu finden.

Folgend möchte ich einige Beispiele anführen:

```
if(document.all){
  this.indicator_offsetX = 2;//Offset where nodes would be dropped
  this.indicator_offsetY = 2;
if(navigator.userAgent.indexOf('Opera')>=0){
  this.indicator_offsetX = 2;//Offset where nodes would be dropped
  this.indicator_offsetY = -7;
}
```

Listing 22: Beaufschlagung eines „Offset“ um richtige Drop-Position zu erhalten

Für die korrekte Durchführung der Drop-Operation musste der ermittelten Position während des Loslassens der Maustaste ein gewisses Offset dazugegeben werden. Diese war, wie in Listing 22 zu sehen, bei Opera in der Y-Richtung anders zu setzen.

```
var screen_y = 0;
var isIe/*@cc_on = true@*/; //is ie browser
if (isIe && document.all) {
      screen_y = document.body.clientHeight;
else {
      screen_y = window.innerHeight;
}
```

Listing 23: Ermittlung der Höhe des Fensters

Ein weiteres Beispiel ist in Listing 23 zu sehen. Die Ermittlung der Fensterhöhe ist mit Internet Explorer anders zu gestalten als mit den restlichen Browsern.

```
var aTag = thisObj.getElementsByTagName('A')[0];
var span = document.createElement('span');
span.setAttribute("onmouseout","this.style.backgroundColor='transp
arent';");
span.setAttribute("onmouseover","this.style.backgroundColor='gold'
;");
span.innerHTML = aInner;
aTag.innerHTML = "";
aTag.appendChild(span);
var isIe/*@cc_on = true@*/;

if(isIe)
{ //ie Hack
var imgieinner = thisObj.innerHTML.split("<IMG");
var imgie = document.createElement("<IMG"+imgieinner[1]);
thisObj.removeChild(thisObj.getElementsByTagName('IMG')[0]);
thisObj.insertBefore(imgie,thisObj.getElementsByTagName('IMG')[0])
var ieInner = aTag.innerHTML.split(aInner);
var spanie = document.createElement(ieInner[0]);
spanie.innerHTML = aInner;
aTag.innerHTML = "";
aTag.appendChild(spanie);
}
```

Listing 24: Hinzufügen von Attributen

In Listing 24 wird einem <A>-Tag ein <SPAN>-Element mit Attributen für onmouseover und onmouseout-Events hinzugefügt. Die Attribute lassen sich bei allen Browsern mit der setAttribute-Anweisung durchführen. Wird dies auch für den Internetexplorer auf diese Weise durchgeführt, werden die Attribute nicht erkannt. Um die Funktion auch für IE zu gewährleisten, mussten die besagten Attribute aus dem vorher erstellten Element herausgescannt werden, um sie nachher wieder an der alten Stelle mit der insertBefore-Anweisung einzufügen.

5.5.4 Test des Tree Browsers

Durch einen Softwaretest wird geprüft und bewertet, ob die Anwendung die definierten Anforderungen erfüllt und ob die zur Verfügung stehenden Ressourcen für einen Betrieb des Programmes ausreichen. Die gewonnenen Erkenntnisse werden dazu genutzt, Fehler zu beheben und um softwarebedingte Performanceverluste zu erörtern und zu verbessern.[33]

Der Tree Browser wurde modulweise programmiert und das bestehende System so schrittweise erweitert. In unserem Fall gelten die einzelnen Funktionen als Module.

Um heraus zu finden, ob das Tool richtig arbeitet, wurde folgend getestet:

1. *Modultest:*
   Um frühzeitig Fehler zu entdecken, wurden für jedes neu hinzugefügte Modul Testfälle (Test Cases) definiert und entweder automatisch oder manuell durchgeführt (Funktionstest). Auch die Testfälle der bereits vorher integrierten Module wurden nochmals durchgeführt, um festzustellen ob die Integration des neuen Moduls die Funktionsweise der bisher vorhandenen Software nicht negativ be-

[33] (http://de.wikipedia.org/wiki/Test_(Informatik)) [Zugriff: 15.12.2011]

einflusst hat. Diese Vorgehensweise ist auch unter der Bezeichnung *Regressionstest* bekannt.

2. *Integrationstest:*

   Nachdem alle Komponenten zusammengefügt wurden, erfolgte ein Integrationstest, wodurch die Funktion der gesamten Software garantiert werden sollte. In unserem Fall wurde jedes neue Modul zur bereits bestehenden Software direkt integriert und getestet. Dies bedeutet, dass die Integration des letzten Moduls, dessen Testung sowie die Überprüfung aller vorher integrierten Komponenten gleichzeitig der Integrationstest war.

3. *Systemtest:*

   Der Systemtest wird entweder am System durchgeführt, welches zur Ausführung der Software herangezogen wird oder auf einer Testumgebung, welche die Ressourcen des ausführenden Systems bereitstellt. In dieser Phase soll neben der korrekten Funktion unter Belastung auch festgestellt werden, ob die Hardware (CPU, Speicher, Netzbandbreite) einen sicheren und effizienten Betrieb zulässt (Lasttest). Die Erkenntnisse daraus können ein Erweitern der Hardwarekomponenten oder Änderungen an der Software nach sich ziehen.

   Da das Tool leider nie am System des Austria-Forums integriert war, und ich auch bei weitem kein Testsystem mit ähnlichen Ressourcen zur Verfügung hatte, konnte kein ordentlicher Systemtest durchgeführt werden. Trotzdem wurde mit einem rechenintensiven Teil des Tools ein Lasttest durchgeführt.

5.5.4.1 Funktionstest

Funktionstests werden verwendet, um die korrekte Lösung eines funktionalen Problems zu überprüfen (anhand von formulierten Testfällen). Der Funktionstest ist Teil des Modul- und Integrationstests (Schimratzki, 2009).

Im Idealfall kann ein Softwaretest automatisch durchgeführt werden. Dadurch kann der Test nach jeder Programmänderung automatisch von praktisch jedem schnell und kostengünstig nochmals durchgeführt werden. Häufig gibt es Frameworks, die einem beim Schreiben von Testprogrammen unterstützen.

Auch für webbasierte Programme stehen Werkzeuge zur Verfügung, um automatische Tests durchzuführen. Einige *Open Source* Testfallgeneratoren sind z. B:

- Selenium
- TestGen4Web
- TCP Proxy
- Sahi

Leider musste ich feststellen, dass viele Generatoren zwar DHTML-Anwendungen laut Spezifikation unterstützen sollen, diese Anforderung in unserem Fall jedoch nur bedingt erfüllt haben. So war es z. B. nicht möglich, Drag/Drop zu simulieren. Daher konnten nicht alle Teile automatisch getestet werden, der Rest musste in *Handarbeit* durchgeführt werden.

Zur Generierung der automatisierten Testfälle wurde *PushToTest TestMaker* benutzt. Diese Plattform besitzt einen *Recorder*, welcher es ermöglicht, Testfälle für Selenium oder Sahi aufzuzeichnen und abzuspielen. Die erzeugten Skripte können dann auch noch von Hand angepasst werden.

Als Beispiel ist in Listing 25 ein Sahi-Skript zu sehen, welches das Erweitern von zwei Knoten simuliert. Die als XML gespeicherten Skripte sind einfach zu lesen und wenn benötigt, auch einfach zu ändern. Sie werden bei Ausführung vom Generator sequentiell abgearbeitet.
Im **ersten Schritt** mit der Bezeichnung *sequence=0* wird zur Seite des zu testenden Systems navigiert.

Im **zweiten Schritt** mit der Bezeichnung *sequence=1* wird auf ein „Plus-Icon“ geklickt. Hier wird also ein Knoten erweitert (expand).
Im **dritten Schritt** mit der Bezeichnung *sequence=2* wird ein weiterer Knoten erweitert.

```
<testcase name='exp_col' description='' author='' version='2.1'>
<dpl filename="" exectype="" errorhandling="stop"/>
<object-repository nodeJson=''>
</object-repository>

<step action="navigateTo" val-
ue="&quot;http://vali:8050/af/tree.jsp?redirect=Austria-
Forum&amp;settree=true&quot;" type="web" sequence="0">
<dialect type="sahi">
<accessor type="navigateTo" value='' selected="true"/>
</dialect>
</step>

<step action="click" value='' type="web" sequence="1">
<dialect type="sahi">
<accessor type="click" value="image(&quot;_plus.gif[2]&quot;)"
selected="true"/>
<accessor type="click" value="image(10)" selected="false"/>
</dialect>
</step>

<step action="click" value='' type="web" sequence="2">
<dialect type="sahi">
<accessor type="click" value="image(&quot;_plus.gif[1]&quot;)"
selected="true"/>
<accessor type="click" value="image(10)" selected="false"/>
</dialect>
</step>
</testcase>
```

Listing 25: Sahi-Skript: Simuliert das Erweitern und Zuklappen eines Knotens

Die Skripte können beliebig zusammengesetzt und somit die verschiedenen Testfälle erzeugt werden.

Jede einzelne Sequenz wird vom Generator auf dessen richtige Ausführung kontrolliert. Das Ergebnis wird in einem Fenster ausgegeben (siehe Abbildung 89). Darin ist ersichtlich, welche Schritte korrekt oder fehlerhaft abgeschlossen wurden und wie lange die einzelnen Schritte gebraucht haben (was bei Funktionstests jedoch nebensächlich ist). Ein genaueres Resultat ist anhand der mitgeloggten Ereignisse auf der Konsole ersichtlich.

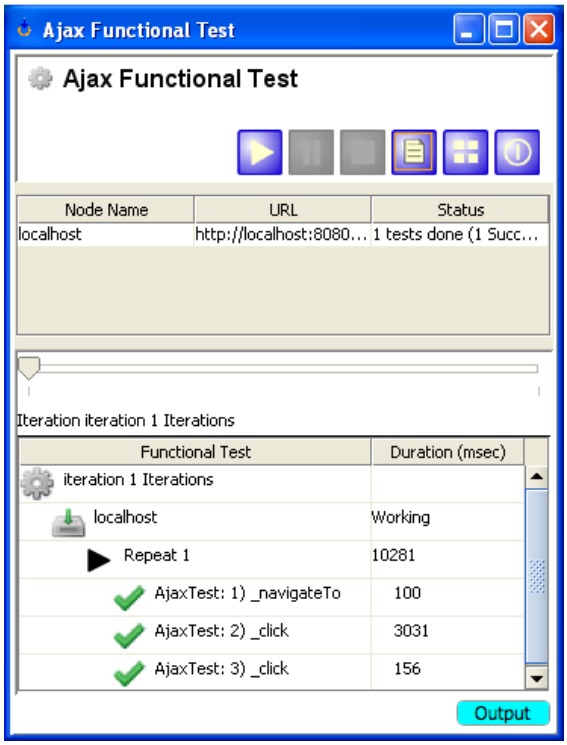

Abbildung 89: Ergebnis eines ohne Fehler ausgeführten Ajax-Funktions-Test

Die nicht automatisch durchführbaren Funktionstests wurden nach jedem Modul von mir selbst erledigt. Der endgültige Integrationstest wurde von zwei unabhängigen Personen durchgeführt. Sie wurden vorher in die Funktionsweise des Systems eingewiesen. Die definierten Testfälle für jedes Modul wurden durchprobiert und die aufgetretenen Fehler und Verbesserungsvorschläge für die folgende Korrektur mitgeschrieben. Es wurde auch auf verschiedenen Browsern getestet, um auch Cross-Browser-Kompatibilität zu gewähren. Beide Testpersonen waren versierte Internetnutzer und dadurch im Umgang mit verschiedenen Browsern und allfällig auftretenden Fehlern betraut.

5.5.4.2 Lasttest

Anders als bei Funktionstests, wo nur die korrekte Funktionsweise der Software bei einem User betrachtet wird, wird bei Lasttests die Leistung der Software und der zugrunde liegenden Hardware, also des Gesamtsystems, kontrolliert. Auf Basis der Skripte für den funktionalen Test können mit *PushToTest* auch Lasttests durchgeführt werden, indem mehrere virtuelle Benutzer simuliert werden, die diese Fälle sozusagen zur gleichen Zeit ausführen. Durch solche Leistungstests können Parameter wie die Antwortzeit (Zeit zwischen Anfrage und Antwort), der Durchsatz (Transaktionen je Zeiteinheit), die Skalierbarkeit aber auch die Fehleranfälligkeit bei Belastung durch unterschiedlich viele Benutzer getestet werden (Schimratzki, 2009).

Diese Tests sind eigentlich Teil des Systemtests und, wie oben schon beschrieben, nur am ausführenden Server oder einer diesem Server vergleichbaren Testumgebung richtig sinnvoll. Trotzdem sollte ein Lasttest auf einem zur Verfügung stehenden System durchgeführt werden, um Erkenntnisse über das Verhalten der Software bei Belastung zu erhalten.

5.5.4.2.1 Testumgebung für Lasttest

Die Testumgebung bestand aus einem Rechner mit einem Intel Core2 Duo Prozessor mit 3 GHz und 2 GB RAM, welcher den Tomcat Webserver und somit auch AFWiki inklusive des Tree Browsers ausführte.
Die Last wurde mit *PushToTest Testmaker* auf einem Rechner mit einem 2 GHz Intel Core2 Duo Prozessor und 2 GB Arbeitsspeicher erzeugt.
Mit *PushToTest* können bis 50 virtuelle Benutzer für die Lasttests erzeugt werden, die dann in einem variabel wählbaren Intervall eine definierte Funktion zur selben Zeit ausüben.

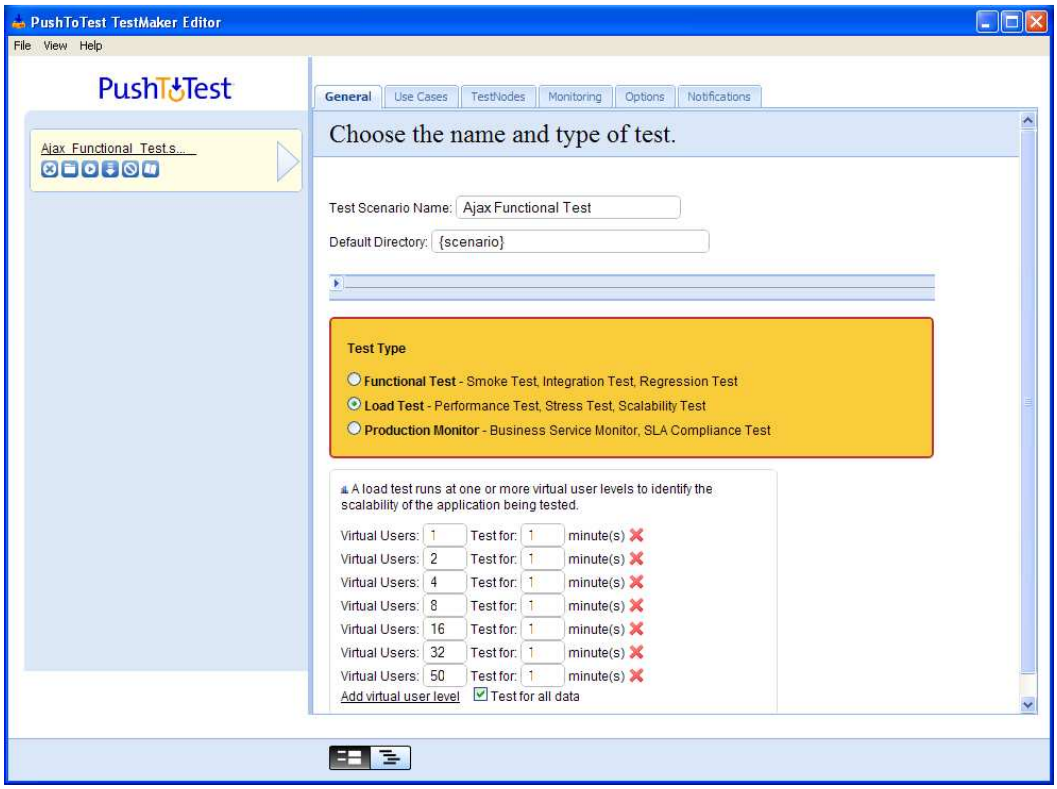

Abbildung 90: Einstellungen Lasttest in *PushToTalk*

In der Versuchsreihe wurden 1, 2, 4, 8, 16, 32 und 50 Benutzer vom Lastgenerator erzeugt, die gleichzeitig den Testfall ausführen. Die Testreihe wurde mit verschiedenen Übertragungsraten durchgeführt:

- 150 Kbit/s (unterer Standard für heutige Internetanbindungen),
- 700 Kbit/s (durchschnittlicher Standard für heutige Internetanbindungen) und
- 40 Mbit/s (durchschnittlicher Standard z.B. in Fimrennetzwerken für Intranet).

5.5.4.2.2 Aufgetretene Probleme und Fehler

Schon bei den ersten Durchläufen mit der niedrigsten Übertragungsrate stellte sich heraus, dass die Antwortzeit für das Erweitern (expand) von Knoten mit vielen Unterknoten (in diesem Fall über 2000) zu lang dauerte. Schon bei einem virtuellen User wurden Zeiten um 10 Sekunden ermittelt. Zwar sind Internetanbindungen mit 150 Kbit/s heutzutage nicht mehr Maß

der Dinge, trotzdem wäre auch hier zumindest bei einem User eine Zeit von maximal 6 Sekunden anstrebenswert.
So war es notwendig, zeitintensive oder häufig aufgerufene Programmteile im Code genauer zu analysieren. Als wesentliche Bremse stellte sich dabei die Abfrage, ob eine *Page* Anhänge besitzt heraus. Diese wird in unserem Testfall bei der Generierung der Liste für die Unterknoten über 2000 mal aufgerufen. In Listing 26 ist dieser Codeteil, wie er in der Funktion `createfileXML(String name, WikiPage page)` in der Klasse `TreePrepare` (java) vorhanden war, zu sehen.

```
if(m_engine.getAttachmentManager().hasAttachments(page)){
      .
      Java Code
      .
}
```

Listing 26: Abfrage nach Anhängen einer Page

Über den AttachmentManager wird die zur Verfügung stehende Funktion zur Abfrage nach Anhängen einer Seite verwendet. Dies wäre dem ersten Anschein nach auch die richtige Vorgehensweise, allerdings können durch folgende, etwas paradox wirkende Lösung wie sie in Listing 27 angeführt ist, 2 bis 3 ms pro Abfrage eingespart werden, abhängig vom ausführenden System. Dies ergibt z.B. bei einer Kategorie mit 2000 Beiträgen eine Zeiteinsparung von 4 bis 6 Sekunden. Die Überprüfung der Menge der *Attachments* einer Page *direkt* über die für die Verwaltung von Anhängen zuständige Klasse `AustriaForumAttachmentProvider` (java), anstatt über den Umweg der Superklasse, bringt hier eine wesentliche Verbesserung.

```
int size = 0;

try {
      AustriaForumAttachmentProvider austriaAttProv = null;
      size = austriaAttProv.listAttachments(pagew).size();

}
catch (ProviderException e1) {
      e1.printStackTrace();
}

if(size>0){

      .
      Java Code
      .
}
```

Listing 27: Verbesserte Abfrage nach Anhängen einer Page

Ein weiteres Problem war das ständige Überlaufen des zur Verfügung stehenden virtuellen Speichers (Java Heap Space) bei der Testreihe ab 50 Usern (siehe Abbildung 91).

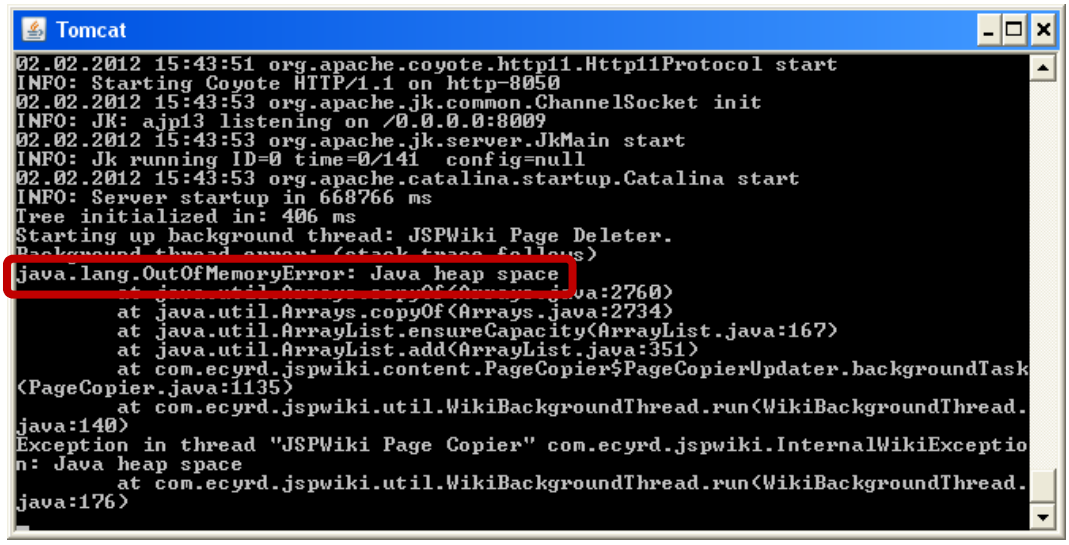

Abbildung 91: Java Heap Space wurde bei 50 Usern ständig überlaufen

Java benützt zur Verwaltung von Datenstrukturen einen Speicherbereich. Dieser wird *Java Heap* genannt und kann bis zu einer initial angegebenen Größe anwachsen. Wird diese Größe erreicht, wird die oben abgebildete *Exception* ausgegeben. Nun könnte man einfach die maximale Größe des *Heaps* vergrößern. Allerdings ist der Speicher eine endliche Größe eines

jeden Computers, weshalb man sich zuerst Gedanken um die Datenstruktur machen sollte.
In Java gibt es keine *Destruktoren*, wie z.B. in C++, um Objekte explizit aus dem Speicher zu löschen. Dafür besitzt Java einen Mechanismus (Garbage Collector), der automatisch im Hintergrund Objekte entfernt, auf die nicht mehr verwiesen wird. Deshalb sollte man nur die notwendigsten globalen Objekte initieren, da diese bis zum Beenden des Programmes referenziert sind und damit im Speicher liegen bleiben. Wichtig ist es auch, Referenzen auf Objekte, die nicht mehr benötigt werden, mit *null* explizit wieder zu dereferenzieren, damit der Speicher vom Garbage Collector wieder freigegeben werden kann. Dies ist vor allem in Methoden wichtig, die nicht gleich abgeschlossen werden.
In den Java Klassen des Tools waren einige Objekte, die nicht sofort dereferenziert wurden. Einige globale Variablen konnten zu lokalen umfunktioniert werden. Nach diesen Maßnahmen konnten die Tests ohne *Java Heap Space Exception* zu Ende geführt werden.

Auch die Fehlerhäufigkeit bei höherer Useranzahl war anfangs etwas groß. Bei 32 Usern wurde eine Fehlerwahrscheinlichkeit von 8% erreicht. Bei 50 Benutzern lag sie schon bei 23%. Dies war aber vermutlich Resultat der oben erwähnten Probleme, da eine zu lange Ladezeit bei vielen Benutzern auch als Fehler gewertet wurde. Außerdem gehen durch einen überlaufenden Speicher auch Anfragen verloren. Auf jeden Fall konnte die Fehlerwahrscheinlichkeit bei großem Benutzeraufkommen durch das Bereinigen der obigen Schwachstellen stark gesenkt werden. Fehler gab es nur noch bei Übertragungsraten von 150 Kbit/s.

5.5.4.2.3 Ergebnisse

Hier sollen noch Ergebnisse des Lasttests, für den unter Funktionstests vorgestellten Testfall gezeigt werden. Das Erweitern von Knoten stellt die rechenintensivste Aufgabe des Tools dar. Außerdem ist es die einzige Aufgabe, welche dem User sofort ein Ergebnis liefern muss (quasi in

Echtzeit). Alle anderen führen die Anforderungen des Users im Hintergrund aus, wodurch die Dauer der Ausführung eigentlich keine Rolle spielt, da zum Weiterarbeiten das Ergebnis nicht abgewartet werden muss. Deshalb sind die Ergebnisse dieses Lasttests interessant.
Für diesen Test wurde ein Knoten mit 2013 Unterknoten (ist ca. die Menge an Beiträgen der zurzeit größten Kategorie im Austria Forum) gewählt, die zusätzlich mit Anhängen versehen waren, wodurch die zu verrichtende Arbeit weiter gesteigert wurde. Der zweite Knoten verfügte über lediglich drei Unterknoten.

- **Lasttest bei einer Übertragungsrate von 150 Kbit/sec**

In Diagramm 1 ist die Skalierbarkeit anhand des Durchsatzes dargestellt. Durchsatz bedeutet die verrichtete Arbeit pro Zeit (Transaktionen pro Sekunde). Wünschenswert wäre es natürlich, dass sich der Durchsatz bei steigender Nutzerzahl derart ändert, dass jeder Benutzer die gleiche Antwortzeit für Anfragen erhält, wie sie ein einzelner User hat. Also dass das System z.B. bei vier gleichzeitig im System arbeitenden Benutzern den vierfachen Durchsatz von einem einzigen Benutzer bereitstellt. Dies ist je nach Hardware und Software nur bis zu einem gewissen Grad möglich. Die Skalierung gibt Auskunft darüber. In der Testreihe wurde die Anzahl der User immer verdoppelt um festzustellen, wie weit das System *skaliert*.

Wie in Diagramm 1 zu sehen, steigt der Durchsatz bis zu einem gewissen Optimum an und fällt dann wieder ab. Dies ist ein ganz normales Verhalten, da bei steigender Netzaktivität auch der Overhead zunimmt und damit zum Absinken des Durchsatzes beiträgt.
Bei einer Datenrate von 150 Kbit/s *skaliert* das System bis 4 User annähernd perfekt. Der Durchsatz erreicht bei 16 Nutzern das Maximum, dann sinkt er wieder ab.

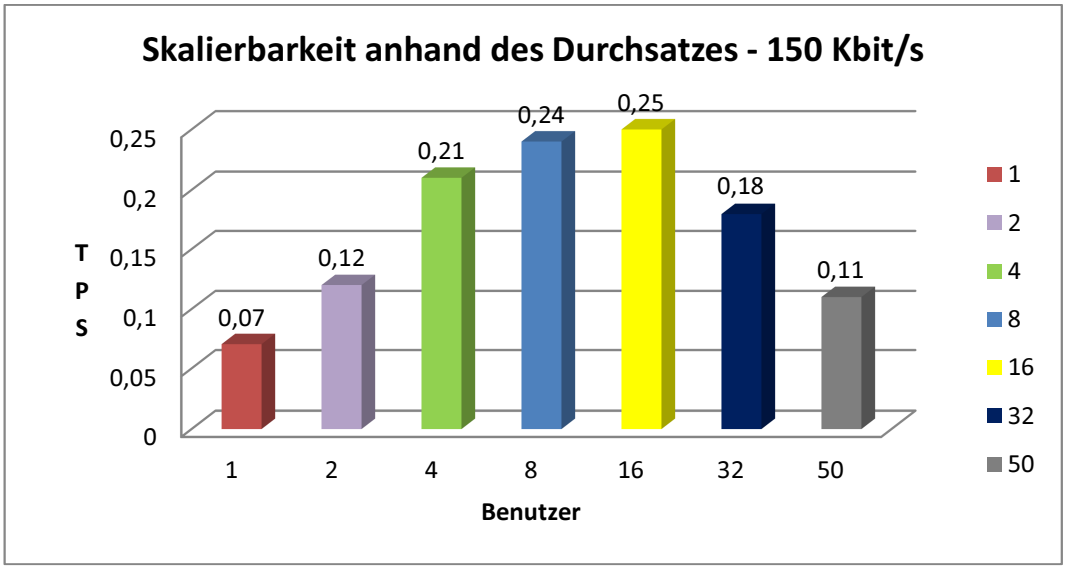

Diagramm 1: Skalierung anhand des Durchsatzes bei 150 Kbit/s

In Diagramm 2 sind die Antwortzeiten nur für die Sequenz des Erweiterns des Knotens mit 2013 Unterknoten für verschieden viele Benutzer abgebildet, also die Zeit zwischen dem Klicken auf das *Plus-Icon* bis zur Anzeige des Ergebnisses am Bildschirm. Ins Diagramm wurde die dabei schlechteste erfasste Zeit eingetragen. Hier ist ersichtlich, dass die Antwortzeit bis 4 Teilnehmer nur gering zunimmt. Bei 8 Teilnehmern ist sie noch akzeptabel. Ab 32 User sinkt der Durchsatz wieder, wodurch die Wartezeit stark zunimmt. Auch Fehler traten ab diesem Punkt auf (siehe Diagramm 3), allerdings wurde die Fehlerhäufigkeit durch die in Kapitel 5.5.4.2.2. beschriebenen Maßnahmen stark verbessert. Bei schnelleren Übertragungsraten traten keine Fehler mehr auf. Dies ließ darauf schließen, dass die Fehlerhäufigkeit neben der Nutzerzahl auch von der Übertragungsrate abhängt.

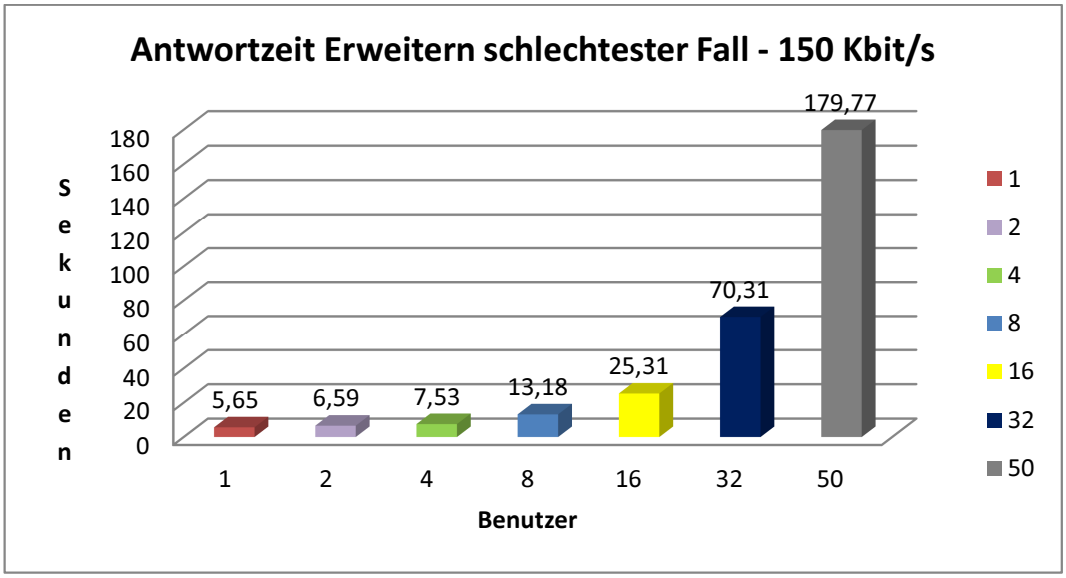

Diagramm 2: Antwortzeit für die Sequenz: „Erweitern eines Knotens mit ca. 2000 Unterknoten“ mit einer Datenanbindung von 150 Kbit/s

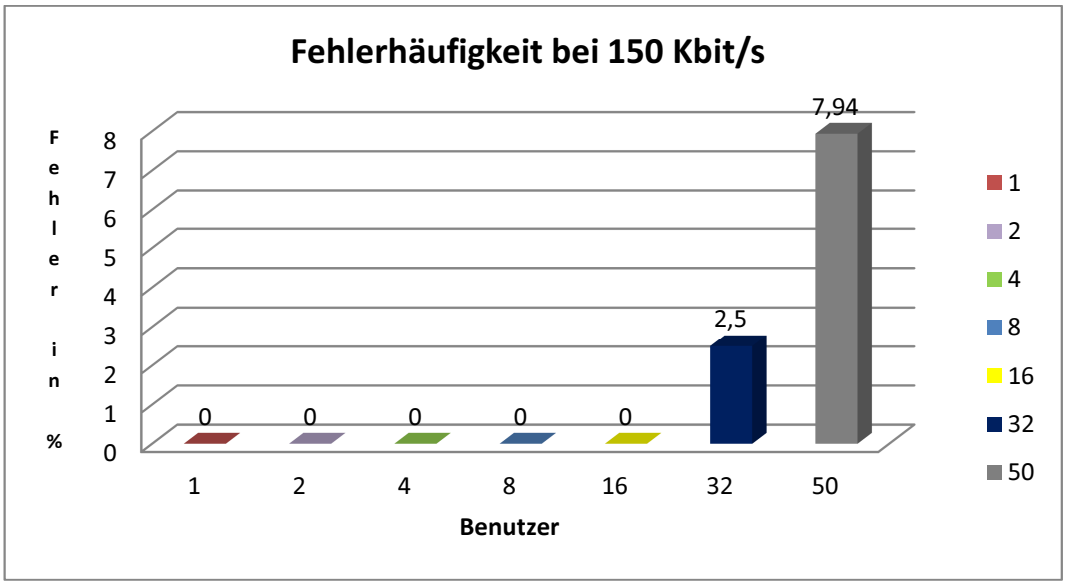

Diagramm 3: Fehleranfälligkeit bei 150 Kbit/s

- **Lasttest bei einer Übertragungsrate von 700 Kbit/s**

700 Kbit/s ist ungefähr die effektive Datenübertragungsrate in UMTS Netzen, also eine gängige Verbindung und somit die aussagekräftigste Testreihe. In der in Diagramm 1 abgebildeten Durchsatzstatistik ist das übliche Verhalten erkennbar. Das System skaliert bis 4 Nutzer perfekt. Dadurch ist festzustellen, dass eine gute Skalierung bei schnelleren Verbindungen

eher erreicht wird. Der höchste Durchsatz wird dann bei 8 Nutzern erreicht. Der Abfall setzt wieder ab 32 Usern ein.

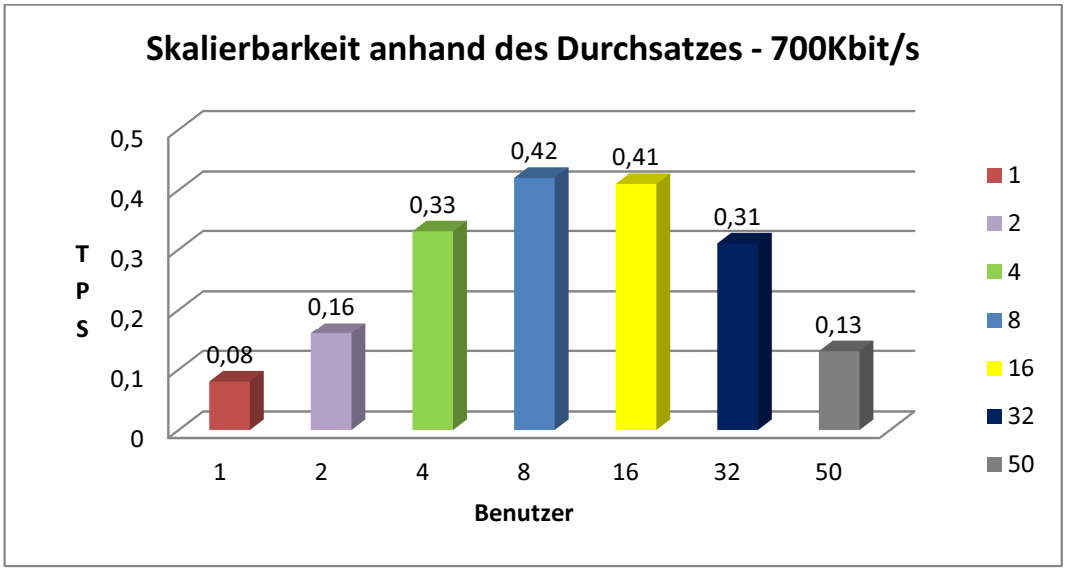

Diagramm 4: Skalierung anhand des Durchsatzes bei 700 Kbit/s

Die maximale Antwortzeit (siehe Diagramm 5) für das Erweitern steigt bis 8 User nur gering an, ist bis 16 User akzeptabel und bis 32 Benutzer vertretbar.

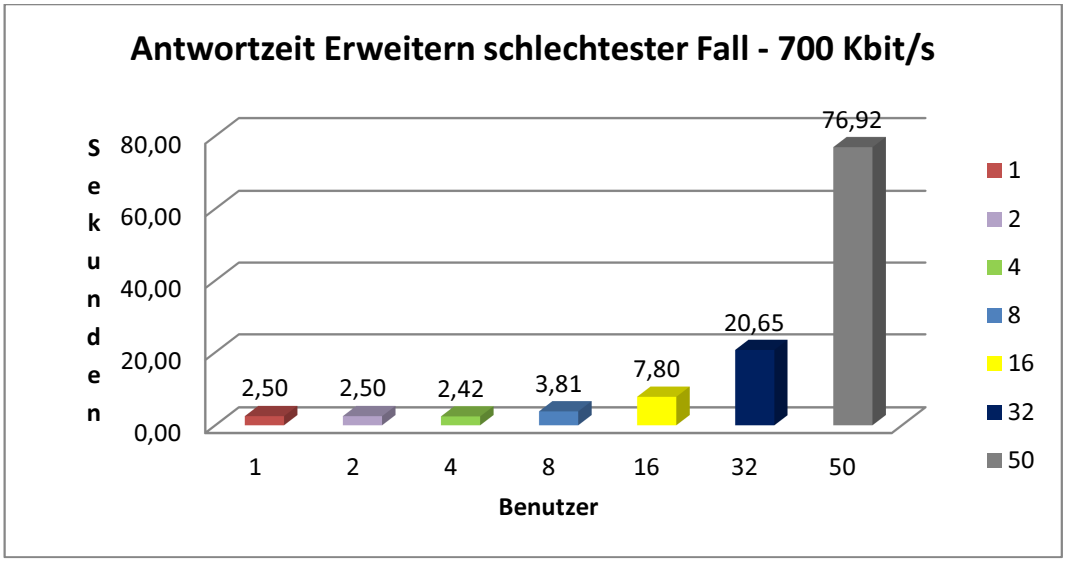

Diagramm 5: Antwortzeit für die Sequenz: „Erweitern eines Knotens mit ca. 2000 Unterknoten“ mit einer Datenanbindung von 700 Kbit/s

Diagramm 6 zeigt die mitgeloggten Antwortzeiten für das Erweitern des Knotens bei 50 Usern. Die Zeiten steigen mit Dauer des Tests von ca. 25 Sekunden bis zu einem gewissen Level stark an und flachen dann ab, bis sie bei ca. 76 Sekunden stagnieren. Dieses Verhalten ist bei hoher Nutzerzahl festzustellen. Bei wenigen Usern verläuft die Linie annähernd auf gleichbleibendem Level, wie in Diagramm 7 zu sehen ist. Grund dafür ist, dass anfangs viele Anfragen annähernd zugleich abgeschickt werden und am Server zur Bearbeitung einprasseln. Ab dem Knickpunkt verteilen sich die *Requests* besser, das System muss weniger Anfragen zur gleichen Zeit entgegennehmen und wird dadurch geringer belastet. Dieses Verhalten ist bei rechenintensiven Operationen stärker ausgeprägt, als bei weniger arbeitsintensiven Funktionen. Natürlich spielen die zur Verfügung stehenden Hardware Ressourcen auch eine Rolle. Je mehr zur Verfügung stehen, desto tiefer der Knickpunkt und desto kürzer ist die *Response Time*. Obwohl sie bei einer üblichen Datenanbindung von 700 Kbit/s und üblich auftretenden Userzahlen schon am Testsystem akzeptabel ist, würde der Einsatz des Tools am Server des Austria-Forums in einer noch kürzeren Antwortzeit und einer weniger ausgeprägten Antwortkurve enden.

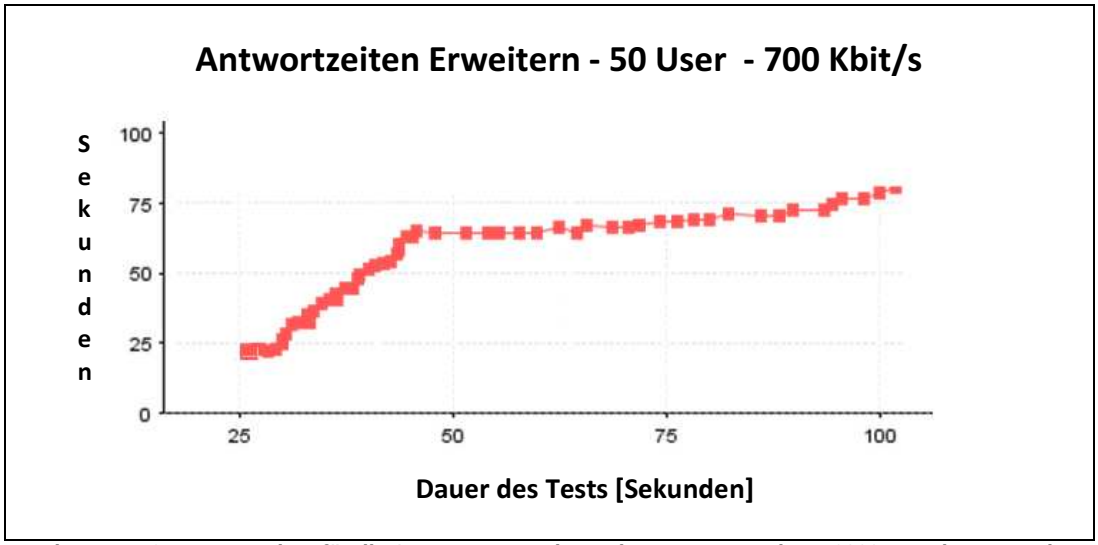

Diagramm 6: Antwortzeiten für die Sequenz: „Erweitern eines Knotens mit ca. 2000 Unterknoten“ mit einer Datenanbindung von 700 Kbit/s bei 50 Benutzern

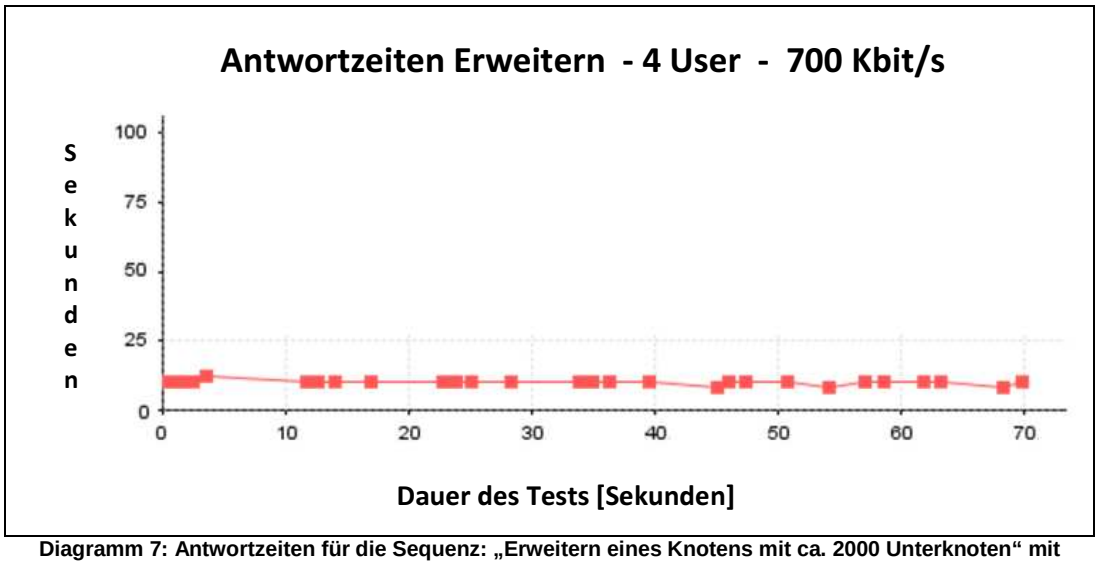

Diagramm 7: Antwortzeiten für die Sequenz: „Erweitern eines Knotens mit ca. 2000 Unterknoten“ mit einer Datenanbindung von 700 Kbit/s bei 4 Benutzern

- **Lasttest bei einer Übertragungsrate von 40 Mbit/sec**

Wie in den vorherigen Testfolgen festgestellt wurde, hängt die Skalierbarkeit stark von der Datenrate ab. Deshalb sollte mit dieser Versuchsreihe getestet werden, ob die Skalierung bei hoher Übertragungsrate weiter verbessert werden kann. Die in Diagramm 8 dargestellte Durchsatzstatistik sowie die in Diagramm 9 dargelegte Antwortzeitenstatistik zeigt, dass auch hier bis 4 Nutzern eine perfekte Skalierung erreicht wird. Danach stagniert der Durchsatz und nimmt bei 32 Usern wieder ab. Das bedeutet, dass das System bereits bei normalen Internetanbindungen ein übliches Skalierungsverhalten aufweist. Die Antwortzeiten können mit schnelleren Anbindungen jedoch weiter reduziert werden.

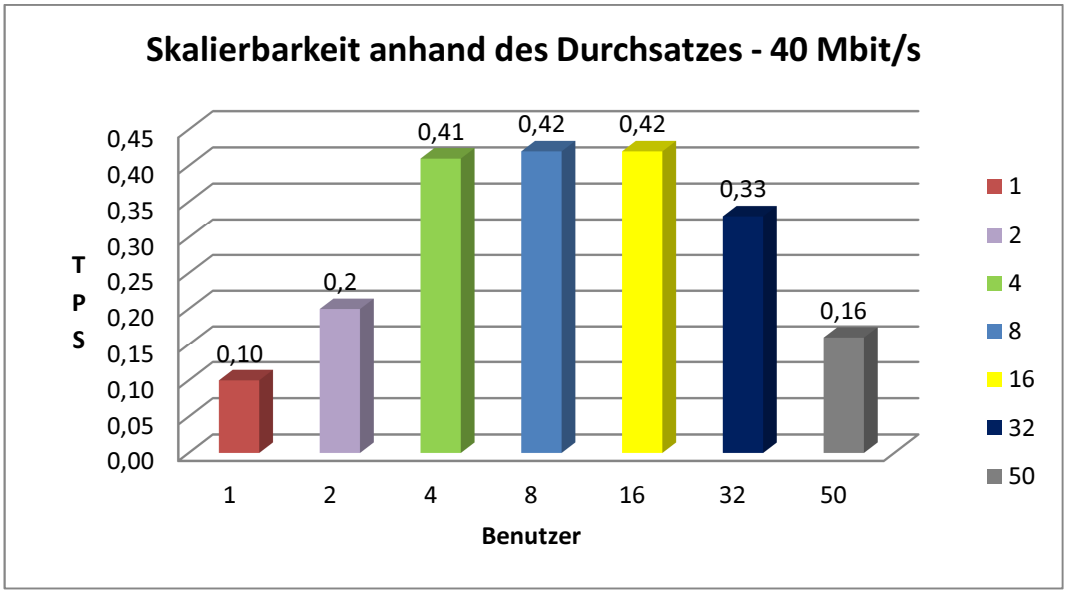

Diagramm 8: Skalierung anhand des Durchsatzes bei 40 Mbit/s

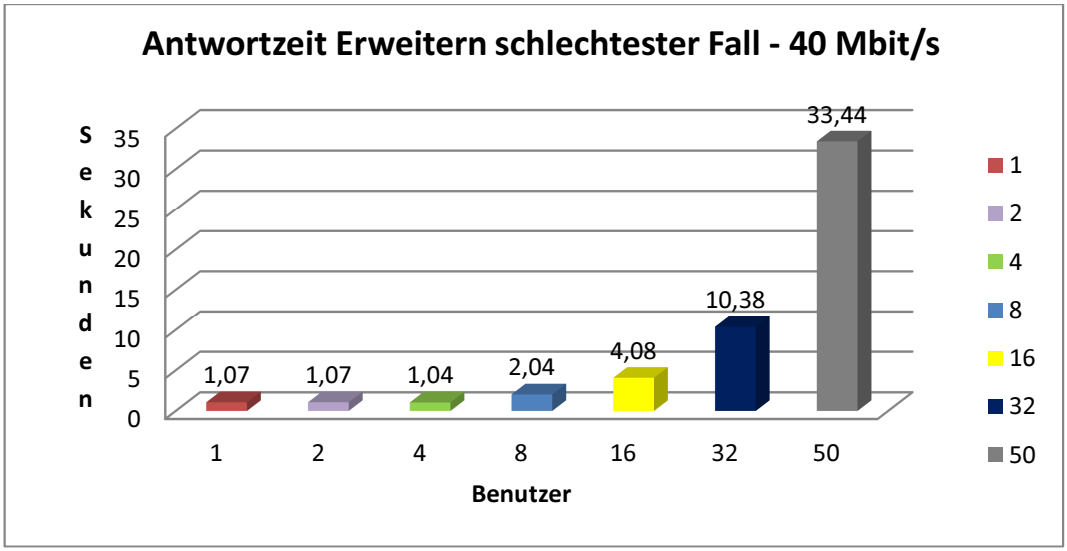

Diagramm 9: Antwortzeit für die Sequenz: „Erweitern eines Knotens mit ca. 2000 Unterknoten“ mit einer Datenanbindung von 700 Kbit/s

- **Vergleich der Skalierung mit bestehenden Systemen und Erkenntnisse**

Um einen Vergleich mit bestehenden Systemen zu haben, wurde ein Testfall für die Suche nach Artikeln mit dem Begriff *Graz* im Austria-Forum und in Wikipedia generiert. Die Versuchsreihe wurde mit 700 Kbit/s durchgeführt. Die Durchsatzstatistiken sind in Diagramm 10 und Diagramm 11 abgebildet. Ein Vergleich mit Diagramm 4 (Skalierung anhand des

Durchsatzes bei 700 Kbit/s) zeigt, dass alle drei Systeme einen ähnlichen Skalierungsverlauf aufweisen. Bis 4 User skalieren alle perfekt. Sie erreichen ein Durchsatzmaximum, welches bei allen ca. beim fünffachen Durchsatz in Bezug auf den bei einem User liegt. Ab 32 User sinkt er dann bei allen wieder.

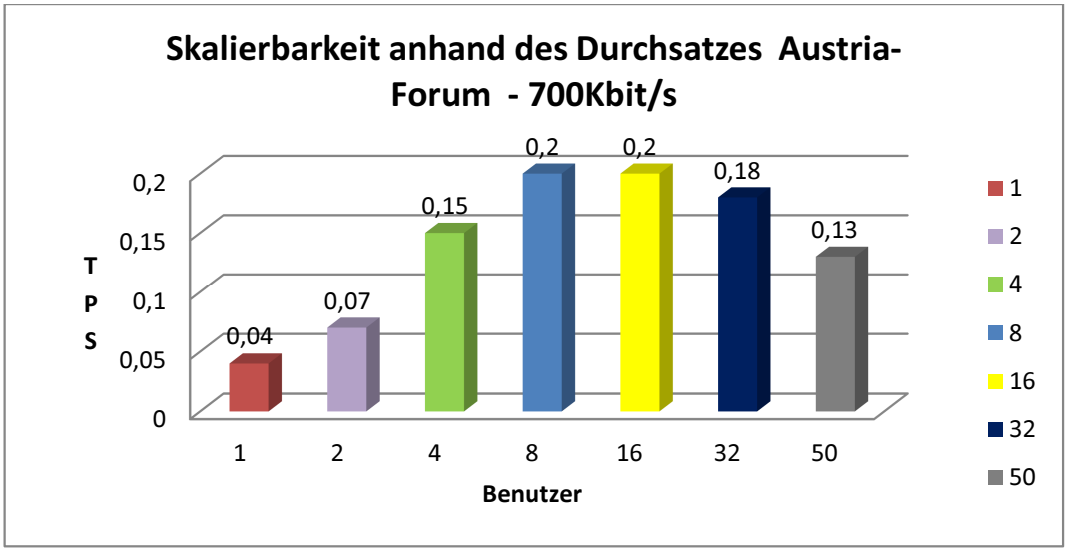

Diagramm 10: Skalierung anhand des Durchsatzes für die Suche im Austria-Forum

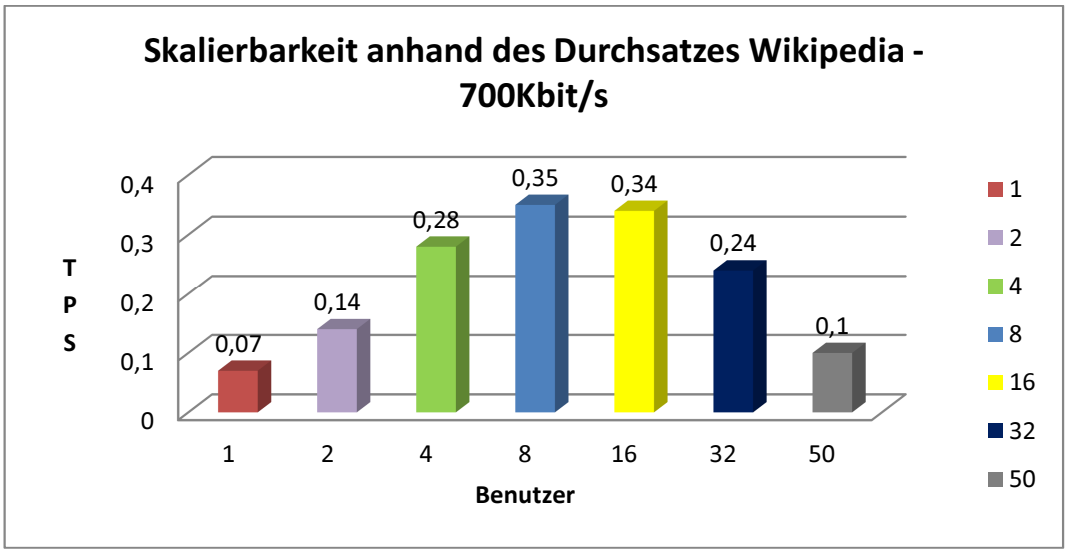

Diagramm 11: Skalierung anhand des Durchsatzes für die Suche in Wikipedia

Wie schon angemerkt wäre es sinnvoll, die Belastungstests am ausführenden Server durchzuführen. Trotzdem konnten durch die Versuchsreihen einige Probleme in der Software erkannt und ausgemerzt werden. Außerdem wurde festgestellt, dass die Software in Sachen Skalierbarkeit

ein durchaus übliches Verhalten aufweist. Auch die Antwortzeiten bei gängigen Datenraten waren akzeptabel, obwohl die zur Verfügung stehende Hardware am Server bei weitem nicht die Ressourcen eines üblichen Webservers erreichte.

5.5.5 Weiterentwicklung

Das Tool unterstützt zurzeit nur die Auswahl eines Elementes um darauf Operationen durchzuführen. Praktisch wäre auch die Auswahl mehrerer Elemente um diese z.B. zu kopieren, zu verschieben oder zu löschen. Momentan können mehrere Elemente nur dann bearbeitet werden, wenn sie in einer Kategorie vorhanden sind. Die Umsetzung könnte so gestaltet werden, dass bei Drücken einer bestimmten Taste (z.B. [STRG]) mehrere Elemente per Mausklick markiert werden können. Diese werden in einem Array gespeichert und können so einzeln zur Durchführung den Methoden übergeben werden.
Neben den üblichen Aufgaben zur Administration von Dateien könnten diesem Tool noch viele andere Funktionen hinzugefügt werden, die das Arbeiten mit Dateien unterstützen wie z.B. Option zur Erstellung von PDF-Dateien aus einem Artikel, Option zum Suchen eines Artikels innerhalb einer Kategorie, Option zum Versenden eines Artikels, Option zum Auslesen der Eigenschaften eines Artikels oder einer Datei, Option zum Packen oder Komprimieren von allen Artikeln einer Kategorie oder einer Ausgewählten Anzahl an Dateien usw.
Durch die Durchführung eines zeitlich definierten Testbetriebes oder eines Beta-Tests könnte anhand von User-Feedbacks die Benutzbarkeit und Performance des Systems verbessert sowie weitere Fehler ausfindig gemacht werden.

6. Zusammenfassung und Fazit

Die Gestaltung der Dateienstruktur bzw. Verzeichnisstruktur ist seit jeher Thema der Informationstechnologie. Wurden anfangs die Daten einfach *flach* ohne Verzeichnis am Datenträger abgelegt, änderte sich dies mit Aufkommen von Mehrbenutzersystemen, immer größer werdenden Datenmengen und Speichermedien allmählich in Richtung hierarchischer Systeme. So konnte die Namensvergabe, besonders in Zeiten als die Länge des Dateinamens noch auf wenige Zeichen begrenzt war, ungemein vereinfacht werden. Aber auch die Übersicht und Organisation der Daten konnte dadurch entscheidend verbessert werden. Durch die Bereitstellung der wesentlichen Dateioperationen konnte sich nun jeder Nutzer seinen eigenen Ordner, Unterordner, Unterunterordner usw. erstellen und so nach Belieben und eigenem Ermessen seine Dateien verwalten.

Durch die Entwicklung der Visualisierung entstanden im Laufe der Zeit auch geeignete graphische Oberflächen zur Erleichterung der Bedienung von Computersystemen. Die Visualisierung von Hierarchischen Systemen ist dabei ein wesentliches Thema der Informationsvisualisierung. Ständig wurden neue Systeme zur Darstellung von Hierarchien bzw. hierarchischen Verzeichnissystemen entwickelt. Eine davon ist der Tree Browser, der wohl bekannteste und beliebteste Vertreter auf Computersystemen dieser Zunft. Die Gründe dafür sind wohl die unkomplizierte Bedienung, das gute *Platzverbrauch/Übersicht-Verhältnis*, die Ähnlichkeit zu bereits vorhandenen Darstellungen von Hierarchien (Inhaltsverzeichnis eines Buches) und die große Verbreitung aufgrund des Einsatzes in den meist benutzten Dateimanagern (Explorer, KDE Konqueror usw.). Hat sich der normale Computerbenutzer einmal auf ein System gewöhnt, ist es natürlich nicht so einfach zu verdrängen. Nichts desto trotz stellt ein Tree Browser, ausgestattet mit wichtigen Dateioperationen ein mächtiges Tool zur Dateienverwaltung dar.

Wiki-Systeme sind ein wesentlicher Bestandteil von Web 2.0. Aufgrund ihrer Philosophie unterstützen sie *Kollaboratives Schreiben* im Netz und dadurch die Verbreitung kollektiven Wissens (siehe Wikipedia). Viele Wiki-Systeme speichern die erstellten Texte in Textdateien, welche flach am Datenträger abgelegt werden. Auch JSPWiki, auf dessen Code das System des Austria-Forums basiert, legt seine Dateien auf diese Weise ab. Auch hier stellte sich mit dem Aufkommen immer größer werdender Dateimengen das gleiche Problem ein, wie es in der Informationstechnologie schon vor Jahren vorhanden war. Eine Hierarchisierung der Dateienverwaltung ist somit auch hier eine geeignete Methode zur Verbesserung der Situation und außerdem der Grundstock zur Strukturierung und Schaffung einer besseren Übersicht über das Angebot im System.

Ein hierarchisches Dateiensystem lässt sich gut in *Baumform* visuell darstellen. Wie bereits aus der Entwicklung der nicht webbasierten Informationstechnologie bekannt, ist der Tree Browser ein gut einsetzbares Tool um auf wenig Platz einen guten Überblick über die vorhandenen Dateien zu schaffen. Auch das Auffinden von Beiträgen wird durch die Möglichkeit der hierarchischen Suche und somit auch die Navigation durch das Angebot gut unterstützt. Versehen mit den gängigsten Dateioperationen kann dieses System obendrein gute Dienste zur Datenadministration bieten.

Der Tree Browser stellt also zwei wesentliche Funktionen zur Verfügung:
Zum Ersten bietet er durch die visuelle, hierarchische Darstellung der Kategorien, Unterkategorien und Beiträge die Möglichkeit zur Navigation durch das System. Im Paper „*The Effects of Navigation Tools on the Navigability of Web-Based Information Systems*“ (Helic, Hasani-Mavriqi, Wilhelm, & Strohmaier, 2011) wird dem Austria-Forum alleine durch den Einsatz von *Breadcrumbs* eine effiziente Navigierbarkeit bescheinigt. Ein ähnliches Ergebnis könnte auch durch den Einsatz des Tree Browsers erwartet werden. Ein Webportal stellt sein Angebot allerdings besser dar, je mehr Platz zur Verfügung steht. Obwohl der Tree Browser, im Gegen-

satz zu anderen Visualisierungen für Hierarchien, sehr raumsparend zu Tage tritt, würde dessen Einsatz das Platzangebot für die Darstellung des eigentlich wichtigen Angebotes des Forums stark beschneiden, besonders auf kleinen Bildschirmen. Das Austria-Forum besitzt neben *Breadcrumbs* auch noch andere Werkzeuge zur Navigation und Orientierung im System. Somit ist der Tree Browser für diese Funktion obsolet.

Zum Zweiten bietet er Möglichkeiten zur Administration der vorhandenen Dateien. Im Paper „*The Singularity is Not Near: Slowing Growth of Wikipedia*“ (Suh, Convertino, Chi, & Pirolli, 2011) wird spekuliert, dass das verlangsamte Wachstum von Wikipedia in den letzten Jahren neben der erhöhten Bürokratie auch auf die Qualität der Werkzeuge zur Administration der Beiträge zurückzuführen ist. Der Tree Browser könnte damit durchaus ein Tool sein, um Wikipedia in dieser Misere behilflich zu sein. Das Austria-Forum verfolgt allerdings eine andere Philosophie als Wikipedia. Hier wird das Wachstum und die Administration der Beiträge durch das *Editorial Board* bestimmt. Also stünden die dateiadministrativen Fähigkeiten des Tree Browser nur einer geringen Anzahl von Usern zur Verfügung, die obendrein auch andere Möglichkeiten dafür besitzen. Daher würde der Einsatz des Tools für diese Funktion auch nicht unbedingt von Vorteil sein.

Diese Gründe führten schlussendlich zur Endscheidung, das Tool im Austria-Forum nicht einzusetzen.

Literaturverzeichnis

Hawaiian Dictionaries. (2003). Abgerufen am 19. Dezember 2011 von http://wehewehe.org/gsdl2.5/cgi-bin/hdict?e=q-0hdict--00-0-0--010---4----den--0-000lpm--1en-Zz-1---Zz-1-home-wiki--00031-0000escapewin-00&a=q&d=D21021

Vor-&Nachteile: Lineare Listen, Bäume. (13. März 2005). Abgerufen am 19. 12 2011 von http://www.delphi-forum.de/viewtopic.php?sid=7b964534a078bb1e7309fd2714cde60c&t=38272&start=0

GNU GENERAL PUBLIC LICENSE. (29. Juni 2007). Abgerufen am 29. November 2011 von http://www.gnu.org/licenses/gpl.html

Charles Minard - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Charles_Joseph_Minard

Comparison File Systems - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://en.wikipedia.org/wiki/Comparison_of_file_systems

Dateiformat. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Dateiformat

Dateisystem. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Dateisystem

ENZYKLO online Enzyklopädie. (2011). Von http://www.enzyklo.de/Begriff/hierarchische%20Suche abgerufen

Figurative human knowledge - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://en.wikipedia.org/wiki/Figurative_system_of_human_knowledge

Florence Nightingale - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Florence_Nightingale

Hierarchie. (November 2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Hierarchie

Höhle Lascaux - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/H%C3%B6hle_von_Lascaux

Jacques Bertin - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Jacques_Bertin

Red Hat. (2011). Abgerufen am 19. Dezember 2011 von http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-isa-de-4/s1-storage-usable.html

Scientific_visualization - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://en.wikipedia.org/wiki/Scientific_visualization

Tree_structure - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://en.wikipedia.org/wiki/Tree_structure

Verzeichnisstruktur - Wikipedia. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Verzeichnisstruktur

Visualisierung. (2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Visualisierung

Wiki - Wikipedia. (19. November 2011). Abgerufen am 19. Dezember 2011 von http://de.wikipedia.org/wiki/Wiki

Andrews , A., & Kasanicka, J. (2007). A Comparative Study of Four Hierarchy Browsers using the Hierarchical Visualisation Testing Environment (HVTE). Institut für Infoormationssysteme und Computer Medien Technische Universität Graz.

Andrews, K. (April 2008). Information Visualisation – Course Notes. S 7, S 19. Institut für Informationssysteme und Computer Medien, Technische Universität, Graz.

Baumgartner, U., & Siegert, H. J. (2007). Betriebssysteme. S 183 ff. Oldenburg: Wissenschaftsverlag GmbH.

Beaudoin, L., Parent, M. A., & Vroomen, L. C. (1996). *Cheops: A Compact Explorer For Complex Hierarchies*. Abgerufen am 19. Dezember 2011 von http://maparent.ca/~maparent/paper.html

Busskamp, D. (kein Datum). *Lizenzen - GPL - LGPL*. Abgerufen am 29. November 2011 von http://www.dbus.de/eip/kapitel04b.html

Egger, T. (22. Juni 2004). Visualisierung hierarchischer Daten. S 1, S 4 - 5, S 10 ff. Technische Universität Wien.

Garret, J. J. (2005). *Ajax: A New Approach to Web Applications*. Abgerufen am 1. Dezember 2011 von http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

Geuter, J. (22. Januar 2009). Konzeption und Entwicklung einer semantikbasierten Managementschicht für Dateisysteme. S 23 ff. Universität Oldenburg.

Helic a, D., Saeed, A. U., & Trattner, C. (2009). Creating Dynamic Wiki Pages with Section-Tagging. S 18 - 22. ACM.

Helic, D., Hasani-Mavriqi, I., Wilhelm, S., & Strohmaier, M. (2011). The Effects of Navigation Tools on the Navigability of Web-Based Information Systems. IICM und KMI der TU-Graz.

http://c2.com/cgi/wiki?MediaWiki. (kein Datum). Abgerufen am 7. Oktober 2011 von http://c2.com/cgi/wiki?MediaWiki

http://de.wikipedia.org/wiki/Test_(Informatik). (kein Datum). Von http://de.wikipedia.org/wiki/Test_(Informatik) abgerufen

http://de.wikipedia.org/wiki/Wikipedia:Hauptseite. (kein Datum). Abgerufen am 6. Oktober 2011 von http://de.wikipedia.org/wiki/Wikipedia:Hauptseite

http://doc.jspwiki.org/2.4/wiki/FileSystemProvider. (kein Datum). Abgerufen am 8. November 2011 von http://doc.jspwiki.org/2.4/wiki/FileSystemProvider

http://doc.jspwiki.org/2.4/wiki/PageFilters. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/PageFilters

http://doc.jspwiki.org/2.4/wiki/PageStorage. (kein Datum). Abgerufen am 8. November 2011 von http://doc.jspwiki.org/2.4/wiki/PageStorage

http://doc.jspwiki.org/2.4/wiki/PingWeblogsComFilter. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/PingWeblogsComFilter

http://doc.jspwiki.org/2.4/wiki/ProfanityFilter. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/ProfanityFilter
http://doc.jspwiki.org/2.4/wiki/RCSFileProvider. (kein Datum). Abgerufen am 8. November 2011 von http://doc.jspwiki.org/2.4/wiki/RCSFileProvider
http://doc.jspwiki.org/2.4/wiki/SpamFilter. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/SpamFilter
http://doc.jspwiki.org/2.4/wiki/VersioningFileProvider. (kein Datum). Abgerufen am 8. November 2011 von http://doc.jspwiki.org/2.4/wiki/VersioningFileProvider
http://doc.jspwiki.org/2.4/wiki/Wiki.Admin.Security. (kein Datum). Abgerufen am 7. November 2011 von http://doc.jspwiki.org/2.4/wiki/Wiki.Admin.Security
http://doc.jspwiki.org/2.4/wiki/Wiki.Forms. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/Wiki.Forms
http://doc.jspwiki.org/2.4/wiki/Wiki.How%20To.Use%20Variables. (kein Datum). Abgerufen am 4. November 2011 von http://doc.jspwiki.org/2.4/wiki/Wiki.How%20To.Use%20Variables
http://drupal-wiki.com. (kein Datum). Abgerufen am 13. Oktober 2011 von http://drupal-wiki.com
http://hsqldb.org/doc/guide/ch01.html#N101A8. (kein Datum). Abgerufen am 29. November 2011 von http://hsqldb.org/doc/guide/ch01.html#N101A8
http://info.tiki.org/display71. (kein Datum). Abgerufen am 13. Oktober 2011 von http://info.tiki.org/display71
http://twiki.org/cgi-bin/view/TWiki/WebHome. (kein Datum). Abgerufen am 13. Oktober 2011 von http://twiki.org/cgi-bin/view/TWiki/WebHome
http://www.dhtmlgoodies.com/scripts/drag-drop-folder-tree/drag-drop-folder-tree.html. (kein Datum). Abgerufen am 29. November 2011 von http://www.dhtmlgoodies.com/scripts/drag-drop-folder-tree/drag-drop-folder-tree.html
http://www.dhtmlx.com/docs/products/dhtmlxTree/index.shtml?pl. (kein Datum). Abgerufen am 29. November 2011 von http://www.dhtmlx.com/docs/products/dhtmlxTree/index.shtml?pl
http://www.dokuwiki.org/dokuwiki. (kein Datum). Abgerufen am 10. Oktober 2011 von http://www.dokuwiki.org/dokuwiki
http://www.gnu.org/licenses/lgpl.html. (kein Datum). Abgerufen am 29. November 2011 von http://www.gnu.org/licenses/lgpl.html
http://www.jspwiki.org/wiki/JSPWikiPlugins. (kein Datum). Abgerufen am November 2011 von http://www.jspwiki.org/wiki/JSPWikiPlugins
http://www.jspwiki.org/wiki/JSPWikiTemplates. (kein Datum). Abgerufen am 4. November 2011 von http://www.jspwiki.org/wiki/JSPWikiTemplates
http://www.jspwiki.org/wiki/JSPWikiTree. (kein Datum). Abgerufen am 29. November 2011 von http://www.jspwiki.org/wiki/JSPWikiTree
http://www.mediawiki.org/wiki/MediaWiki/de. (kein Datum). Abgerufen am 13. Oktober 2011 von http://www.mediawiki.org/wiki/MediaWiki/de

http://www.wikimatrix.org/show/DokuWiki. (kein Datum). Abgerufen am 10. Oktober 2011 von http://www.wikimatrix.org/show/DokuWiki

http://www.wikimatrix.org/show/Drupal-Wiki. (kein Datum). Abgerufen am 2011. Oktober 2011 von http://www.wikimatrix.org/show/Drupal-Wiki

http://www.wikimatrix.org/show/MediaWiki. (kein Datum). Abgerufen am 10. Oktober 2011 von http://www.wikimatrix.org/show/MediaWiki

http://www.wikimatrix.org/show/Tiki-Wiki-CMS-Groupware. (kein Datum). Abgerufen am 10. Oktober 2011 von http://www.wikimatrix.org/show/Tiki-Wiki-CMS-Groupware

http://www.wikimatrix.org/show/TWiki. (kein Datum). Abgerufen am 10. Oktober 2011 von http://www.wikimatrix.org/show/TWiki

http://www.wikimatrix.org/statistic/Most+Views. (kein Datum). Abgerufen am 8. Oktober 2011 von http://www.wikimatrix.org/statistic/Most+Views

Jetter, H.-C. (2006). Hyperbolic Treebrowser – Der hyperbolische Browser für hierarchische Daten. S 5 ff. Uni Konstanz.

Kerren, A. (2011). *Visualisierung von Hierarchien*. Von http://www-hagen.informatik.uni-kl.de/~kerren/courses/projects/index.html abgerufen

Kleiberg, E., Wetering, H. v., & Wijk, J. v. (2001). Botanical Visualization of Huge Hierarchies. S 7. TU Eindhoven.

Klempt, A. (28. April 2008). Implizite 3D-Hierarchisierung - Studienarbeit. S 4, 5, 7. Universität Rostock.

Knecht, M., & Schwärzler, M. (2006). *Beamtrees*. Abgerufen am 19. Dezember 2011 von http://www.cg.tuwien.ac.at/courses/InfoVis/HallOfFame/2006/SchwaerzlerKnecht/beamtrees.htm

Koch, S. (2001). *Binärbäume und AVL-Bäume*. Abgerufen am 19. Dezember 2011 von http://www.ai.wu.ac.at/~koch/courses/wuw/archive/inf-sem-ss-01/pinterits/text.html

Konqueror. (kein Datum). Abgerufen am 19. Dezember 2011 von http://www.konqueror.org/features/browser.php

Kuska, J. P. (2000). Wissenschaftliche Visualisierung - Skripte. S 6 ff, S 43.

Ludwig, K. A. (April 2004). STAR – Visualisierung von Daten. S 14, S 15 ff, S 20, S 21, S 24 ff, S 26 ff. Universität Konstanz.

Lyckman, D. (2009). *The Pros & Cons of using Flash vs Javascript*. Abgerufen am 2011. November 2011 von http://logicpool.com/archives/30

Meyer, T. (kein Datum). *Klassen und Prototypen in JavaScript*. Abgerufen am 15. Dezember 2011 von http://www.all-community.de/art290.htm

Moskaliuk, J. (26. August 2008). *Effektiver Einsatz von Wikis: Die Struktur des Wikis fördert die Konstruktion von Wissen*. Abgerufen am 5. 10 2011 von http://blog.moskaliuk.com/effektiver-einsatz-von-wikis-die-struktur-des-wikis-foerdert-die-konstruktion-von-wissen/

Nussbaumer, A. (23. August 2005). Hierarchy Browsers – Integrating Four Graph-Based Hierarchy Browsers into the Hierarchical Visualisation System (HVS). S 15 - 16, S 20 ff, S 63 ff. Institut für Informationssysteme und Computer Medien Technische Universität Graz.

Obertaxer, G. (Februar 2009). Wiki. S 7. TU-Graz.

Pareigis, B., Kahlbrandt, C., & Stephan, K. (2009). Algorithmen und Datenstrukturen für Angewandte und Technische Informatiker. S 99. Department für Informatik Hochschule für Angewandte Wissenschaften Hamburg.

Putz, W. (11. März 2005). The Hierarchical Visualization System – A General Framework for Visualizing Information Hierarchies Using the Example of Information Pyramids. S 14 ff . Institut für Informationssysteme und Computer Medien.

Sayed, R. O.-E. (Juni 2006). Wiki-Systeme im eLearning. S 23 ff . Johann Wolfgang Goethe-Universität Frankfurt/Main.

Schimratzki, O. (16. April 2009). Leistungstests auf Basis von Open Source Werkzeugen für das Content Repository Framework MyCore. S 7. Jena: Heinz-Nixdorf-Stiftungsprofessur für Praktische Informatik in Kooperation mit Thüringische Universitäts- und Landesbibliothek.

Schumann, H. (April 2004). Informationsvisualisierung – Methoden und Perspektiven. S 1, S 2, S 5, S 6. Universität Rostock – Institut für Informatik.

Semmernegg, M. (Februar 2009). Wiki vs. HTML. S 11 ff. TU-Graz.

Sen, E., & Krömer, J. (2008). Hackerkultur und Raubkopiere. S 126 ff. VDM Müller.

Silberschatz, A., Peterso, J., & Galvin, P. (1991). Operating System Concepts - 3. Edition. S 377 - 378, S 381. Addison-Weseley Publishing Company.

Suh, B., Convertino, G., Chi, E. H., & Pirolli, P. (2011). The Singularity is Not Near: Slowing Growth of Wikipedia. Palo Alto Research Center.

Tanenbaum, A. S. (2002). Moderne Betriebssysteme - 2. überarbeitete Auflage. S 407 ff, S 409 ff, S 414 ff, S 422 ff, S 427. Pearson Studium.

Thomas, E. (22. Juni 2004). Visualisierung hierarchischer Daten. S 1. TU Wien.

Trattner a, C. (Jänner 2009). Vom Austria-Forum zum Wiki-Konzept. S 47 - 51, S 61 ff, S 87 ff. TU-Graz.

Trattner b, C., & Helic, D. (2009). EXTENDING THE BASIC TAGGING MODEL: CONTEXTAWARE TAGGING. Kap 2 . IADIS International Conference on WWW: IICM und KMI der TU-Graz.

Trattner c, C., Helic, D., & Strohmaier, M. (2011). On the Construction of Effciently Navigable Tag Clouds Using Knowledge from Structured Web Content. Kap 3.1.

Trattner, C., Helic, D., Hasani-Mavriqi, I., & Leitner, H. (7.-9.. Juli 2010). The Austrian Way of Wiki(pedia)! Development of a Structured

Wiki-based Encyclopedia within a Local Austrian Context. *Kap 4.4, Kap 4.1*. IICM und KMI der TU-Graz.
Tuschl, R. (8. April 2008). Ein Userspace-Dateisystem zur Verwaltung von Grid-Daten. S 5 ff, S 7. Humboldt-Universität Berlin.
Waloszek, G. (2008). *Cone Tree*. Abgerufen am 19. Dezember 2011 von http://www.sapdesignguild.org/community/book_people/visualization/controls/ConeTree.htm
Werle, R. (2011). *Handbuch Internet Recherche*. Von http://www.werle.com/intagent/k5_2.htm abgerufen
Winkler a, D. (14. März 2007). *Visualisierungs- und Interaktionskonzepte zur Integration hierarchischer Daten in die HyperGrid*, S 9 ff. Universität Konstanz.
Winkler, D. (Januar 2006). HyperGridXGL. *Ein Interaktions-Konzept zur Integration von hierarchischen Daten in die HyperGrid*, S 4-5, S 8 ff, S 11, S 13 ff, S 15 ff. Universität Konstanz.
Xerox, P., & Ramana, R. (1996). *Visualizing Large Trees Using the Hyperbolic Browser*. Abgerufen am 19. Dezember 2011 von http://www.sigchi.org/chi96/proceedings/videos/Lamping/hb-video.html
Zeiner, K. (2010). Informatik Grundwissen - Betriebssysteme - Dateien und Verzeichnisse Windows XP. S 4.
Zornow, M. (31. Jänner 2004). Entwicklung eines Konzeptes zur angemessenen Beschriftung von Informationsobjekten. S 12. Universität Rostock.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Diagrammverzeichnis